
Chapter 1:08
Core Worlds
Principia v3.0
Reference Series
(c) Western Star
Entertainment Ltd, 2003-2006

NOTE

- OBJECTPLACER_3A – control, document
- OBJECTPLACER_3A – new form of Obj tag, multilayer, compatiblity with 3A, new capabilities (in place scaling and rotation...)
- OBJECTTRAKER_3A

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Worlds & Engines

- Many leading gaming products are based upon a configurable engine that renders and manages multimedia content.
- Principia provides more than a single engine. You can create your own engine without coding because

ENGINE = WORLD + VIEWERS

- Each such engine is fully configurable and created by YOU with simple scripts!

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Core Worlds

- Worlds are quintessential components of Principia-based applications. They create the virtual environment for the scene.
- Worlds are made of layers, which encapsulate media content and perform many vital management tasks on it.
- Worlds are closely associated with other components such as viewers and maps, to enable easy interactive presentation.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Core Worlds

- **There are many types of world designs, based on the artistic theme of the application:**
  - Point of view: first person, third person, multiview ...etc.
  - Scene nature: natural outdoors, dungeon, room, city ...etc.
  - Scene flow: FPS, RTS, TBS, movie clip ...etc.
  - Contents: characters, npc objects, static objects ...etc.
  - Camera: overhead, following, free moving ...etc.
  - Discretization: none, tiles, grids, tiled grids ...etc.
  - Base topology: unstructured trigs, quads, hexagons ...etc.
  - Movement: unrestricted, map coded, inferential ...etc.
  - Multimedia content: visual SFX, audio effects ...etc.
- **Principia provides full support for implementing these varieties of world design, with great speed and flexibility.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Core Worlds

- **What does a Principia world contain?**
  - World geometry data (typically tile sets)
  - World objects (Principia or user objects...etc)
  - World rendering data (materials... etc)
  - Operations to be executed upon contents
  - Control and organization data
  - User functions
- **All worlds derive from the AX_World class and inherit its layer and operation capabilities.**
- **Chapter 8 covers the foundational, generalized world Principia components. Specialized worlds are covered in later chapters.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Layers

- **Principia world contents are organized in layers to enhance flexibility and power.**
  - Layers determine the order and logic of rendering.
  - Layers apply different operations to objects.
  - Layers organize the multitude of world objects.
  - Layers can re-render geometry using new materials.
  - Layers are key for rich and natural worlds.
  - Good design translates into good layer structure.
- **Principia offers several different layer types for different world designs and render tasks.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Layers

- **Generalized layer contents**
  - Layer ID
  - Layer name
  - Layer type binary code
  - Layer material pre-applied to all layer objects
  - Layer optional link to another layer contents
  - Layer operations desired by the user
  - Contents (Principia AX_Object* set)
  - User procedures bound in code (executed after operations)
- ***IMPORTANT:*** **Layers are rendered in the order of their declaration within the definition of their parent world !!!**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Layers

- **Layers need to be defined before use!**
- **Layer definition modes:**
  - Define layers as individual scripted objects, to be referenced in the world definition at a later stage. This opens the most design options. This is best practice for complex worlds.
  - Inline, within the world definition. Inline layers are of the basic Layer3R kind, and can have a limited range of types, commands and linkages. This is what is done in most demos.
- **Linked layers:**
  - The object stack of linked layer is a pointer to the stack of another layer (but the material and operations can be different).

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Layers

Layer declaration syntax:

- Tag <Layer_Decl> for inline definition: Layer ID, layer name, initial and terminal materials, initial size of objects array
- Tag <Layer_Decl> with layer object reference: Layer ID, three reference fields the use of which depends on world type.
- Tag <Layer_Attr>: LayerID, type, operation and linkage reference. Used to impart types, operations and linkage for inline-defined layers. This tag can be repeated for the same LayerID to accumulate types and operations.
- Tag <Layer_LoD>: LayerID, LoDVar, LoDVal, a logic test which determines whether the layer is rendered based on external control data.
- Tag <Layer_Item>: LayerID, ObjectID adds an object.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Layers

Layer type codes (binary superposable)

- NONE: Ordinary object layer.
- LWT_TILE: The object stack of this layer is a tile geometry layer. Automatically set by generator. Used as hint for rendering processes.
- LWT_PARTED: The contents of this layer are to be rendered considering a partitioning schema. Automatically set by the runtime. Used as hint for rendering tiles and objects.
- LWT_NORENDER: The contents of this layer will not be rendered/updated, although materials and operations will be executed.
- LWT_PURENDER: The contents of this layer will be rendered using the generic ::Render() method instead of ::Update(). This will avoid duplication of things such as animation calculations in linked layers.
- LWT_NVRENDER: Execute layer contents regardless of visibility.
- LWT_ENDCLEAR: Reset the layer contents set after layer update.
- LWT_HIDE: The layer is hidden and not rendered.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Operations

- **Operations are essential Principia components:**
  - Execute operations on the collections of objects within the layer with which they are associated (prior to render)
  - Derive from the AX_Operator* Principia archetype
  - Defined in script and referenced by their ID
- **Typical Principia V3 operations used in world layers**
  - Visibility testing using generic object data
  - Space partitioning for render acceleration
  - Depth and material sort
  - Collision detection
  - Pointer overlap detection
  - And much more...

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Operations

- **Basic operations used in Chapter 8:**
  - OP_VISITEST_3R: Performs screen visibility and relation tests on object sets. Different techniques and different levels of performance and accuracy can be selected by the user. For objects re-rendered in multiple passes, carefully consider which layers should execute this op. In most worlds, non-visible objects will not be updated and rendered.
  - OP_PARTSECT_2R: Layer objects are subject to 2D rectangular partitioning for worlds that possess such partitioning schemas. Different techniques can be selected by the user to partition fixed or moving objects.
  - TBD
  - TBD
  - TBD
- **Multiple operations can be defined and run in the same layer. Their ordering is important.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Sectors

- **A production-scale world may contain hundreds of thousands of objects in its layers. It is not practical to process or update all of them in each frame.**
- **Principia features several generic partitioning algorithms to automatically divide world layers into sectors:**
  - WPART_NONE: No partitioning, process layers in their entirety
  - WPART_RECT2D: Regular rectangular 2D sectors. Useful for overhead 3rd-person worlds such as WorldRT3*
  - WPART_RECT3D: Regular rectangular 3D sectors. Useful for open 1st-person worlds such as WorldXXX*
  - WPART_BSP2D: Binary space partition based on a 2D criterion.
  - WPART_ROOM3D: Room visibility partition. Useful for closed 1st/3rd-person worlds such as WorldXXX*
- **With partitioning enabled, only visible sectors are processed per frame, for dramatic performance gains.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Sectors

- **Principia automatic partitioning is extremely powerful and valuable, but does not forgive small set-up errors:**
  - Forgotten or inappropriate layer partition types. Skipping the contents of a structural layer will typically garble the rendered image. Static layer objects that are moved without a refresh will render incorrectly too. Misspelled layer codes will wreak havoc.
  - Invalid OBJ->SectorCode. When created, objects have invalid sector codes. Normally, the runtime partitioning routine will map these codes. If this does not happen, invalid sector codes will cause objects to not render.
  - Invalid OBJ->XYZ. This is the primary reason for errors. If the user forgets to update the object XYZ data field, the inferred sector code will be invalid or incorrect.
  - Invalid OBJ->InteractionFlag. This flag is the final arbiter of objects renders. By default, it is set to VISIBLE, and may lead to unwanted results for specialized layers.
  - Modified OBJ->StatusFlags. These may be used internally by the engine to avoid duplicate visibility tests. If this flag is erroneously reset, tiles/objects may not render correctly. The same goes for OBJ->FeatureCode flags that control advanced behavior.
  - Modified OBJ->Com32A. These are populated with object grid references by the engine for the benefit of the user application, but not used by the engine. Be careful if the user application relies on these codes.
- **If the user application manipulates objects or layers directly, it must also manage the above codes appropriately. In DEV mode, Principia provides several layer/sector debug facilities.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World LoD

- In a typical game application, the world may have to be rendered across several different levels of magnification.
  - Layer contents may be organized according to level of detail required to render at given magnification.
  - The <Layer_LoD> tag can be used to specify the conditions that have to be met for a layer to render. Layers without LoD data will always render.
  - Typical LoD selection criteria are the overall world magnification, application preferences and GPU capability.
  - Typical LoD layer differences cover tile complexity and count, rendering materials, or object density.
- LoD control is effected by setting the values of the LoD variables. This is typically done by the viewer or by the user application.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Classes

- **WX_World3A:**
  - Simplest world, providing access to the common generic layer management functions.
  - For general purpose and custom design use.
- **WX_WorldRT3A:**
  - For third person overhead and “scrolling” views
  - Rectangular regular tiles.
  - Multilayer composition of tiles.
  - Relatively narrow zoom range.
  - Robust adaptive presentation management.
  - For strategy and board games.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – World Classes

- **Many other types of Principia components are closely associated with world objects:**
  - ***Viewers:*** **Control wrapper objects enable interactive presentation of world objects.**
  - ***Procedures:*** **Generate world components and assemble them into a working world based on procedural user input.**
  - ***Helpers:*** **Implement world content cursors, maps, object pickers and manipulators ...etc.**
- **Each world type is closely associated with specific procedures and viewers. These are essential for full use of the world capabilities.**
- **Each world type is presented here along with detail on its native procedures and viewers.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Core Concepts

- Engines = Worlds + Controls
- Worlds = Sequences of layers
- Layers = Contents (objects) + Operations
- Partitioning
- Tiles
- Generic object flags and functions
- Level-of-Detail controls
- Procedural content generation and DCTs
- Terrain and water data structures
- Terrain multi-pass imaging techniques

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Principia Worlds

- **Principia worlds provide power:**
  - Predefined configurable generic designs with integrated procedural and manual content management capabilities.
  - Layer-based contents organization with fast and automated LoD management.
  - Highly efficient built-in and configurable operations for space partition, visibility testing, interaction detection and many more capabilities.
- **Principia enables developers to create rich and complex worlds, rapidly and easily, across the full design spectrum.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Outline

- Demo 1.06.A – World3A
- Demo 1.06.B – WorldRT3A
- Demo 1.06.C – WorldFP1A

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Chapter 1:08
Core Worlds
Principia v3.0
Reference Series
(c) Western Star
Entertainment Ltd, 2003-2006

D108A – World3A

- **World3A is a direct services interface to the generic AX_World() component. It is basically a container of layers that:**
  - **Holds structured collections of objects**
  - **Performs operations on these, such as visibility testing, partitioning or sorting**
  - **Presents the layers contents upon update**
- **Its simple and generic nature make World3A suitable for creating casual arcade games, or multimedia scenes.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005



This colored and textured polygon is rendered using the programmable GPU pipeline, and one of the simplest possible vertex shaders.

It just transforms the vertices to viewport coordinates and passes them along with per-vertex diffuse color and texture coords to the fixed function pixel pipeline for rasterization.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A – World3A

Objectives

- Show core aspects of world contents management

Requirements and Implementation

- A world of colliding balls that spawn and destruct, and that can be saved and restored at any time
- Visibility testing and partitioning for moving items with two-pass visual FX.

Usage Notes:

- The four tasks in Ex01: structuring, instancing, client code and serialization are common to ALL worlds and ALL applications you will build.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A:Ex01 – World3A

- **Four core tasks for all worlds**
  - ***Structuring:*** **D108 will show how to organize world contents in layers based on operations and type.**
  - ***Instancing:*** **The dynamic creation and destruction of objects with particular properties.**
  - ***Client code:*** **The ability to bind and execute snippets of user-designed functions as part of the world structure.**
  - ***Serialization:*** **Accurately saving and restoring the state and content of an application at any time.**
- **World3A offers a simple framework for presenting how these core tasks, common to all worlds.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Structuring

The process of organizing world contents in layers, and setting layer properties and operations to implement the rendering workflow.

Structuring is driven by the gameplay and graphic design of our application.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Instancing & Client Code

This is what makes gameplay or multimedia objects appear, do their bit, and disappear.

This is where you will have to do some actual development work, instead of letting Principia scripted components do it for you.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Serialization

The process of storing and restoring the complex set of components that make up games and other multimedia applications.

Principia provides a powerful and easy to use serialization framework. You can save and load complex worlds with few lines of code!

We will use D108A:Ex01 to introduce this framework, which is common across all of Principia demos, components and applications.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Saving+Loading Principia applications

- Principia makes it easy to save and restore the complex, cross-referenced data structures and states of full-feature entertainment products.
- There are two main strategies for this: native and serialized. Most applications will use a combination of native operations for their intrinsic data, and use Principia serialization for their game components.
- Principia components and applications are serialized in a PCD3 file format, which is a chunk-based binary format for streaming Principia entertainment application components.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Pure native save+load (any format)

- Client application stores only its own data state
- Upon loading, client instances all significant internal Principia components other than those defined by script configuration.
- Saved client data state must fully map to significant object space
- Lots of client code, small files, no other application can use data easily

Practical serialized save+load (PCD3 format)

- Client application stores data state and calls upon Principia to serialize all internal components involved in the application.
- Upon loading, Principia will automatically restore the internal components and resolve all cross-references.
- Save only high-level client data
- Minimal client code, large files, other applications can read data

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

PCD3 format structure (1/2)

- Standard Principia file header for PCD3 format
  - 4byte BINF marker
  - 4 byte PCD3 file type marker
  - Null-terminated descriptor string
- Sequence of data items saved by the client and Principia
  - Client application data is saved as raw binary form (for primitive data such as floats or for client-defined structures) or as Principia data chunks (for Principia structures such as sets or grids).
  - Principia high-level components saved in data chunks. A world for instance will save and load all its embedded data automatically.
  - The high-level sequence is determined by the save/load procedure design at the client level.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

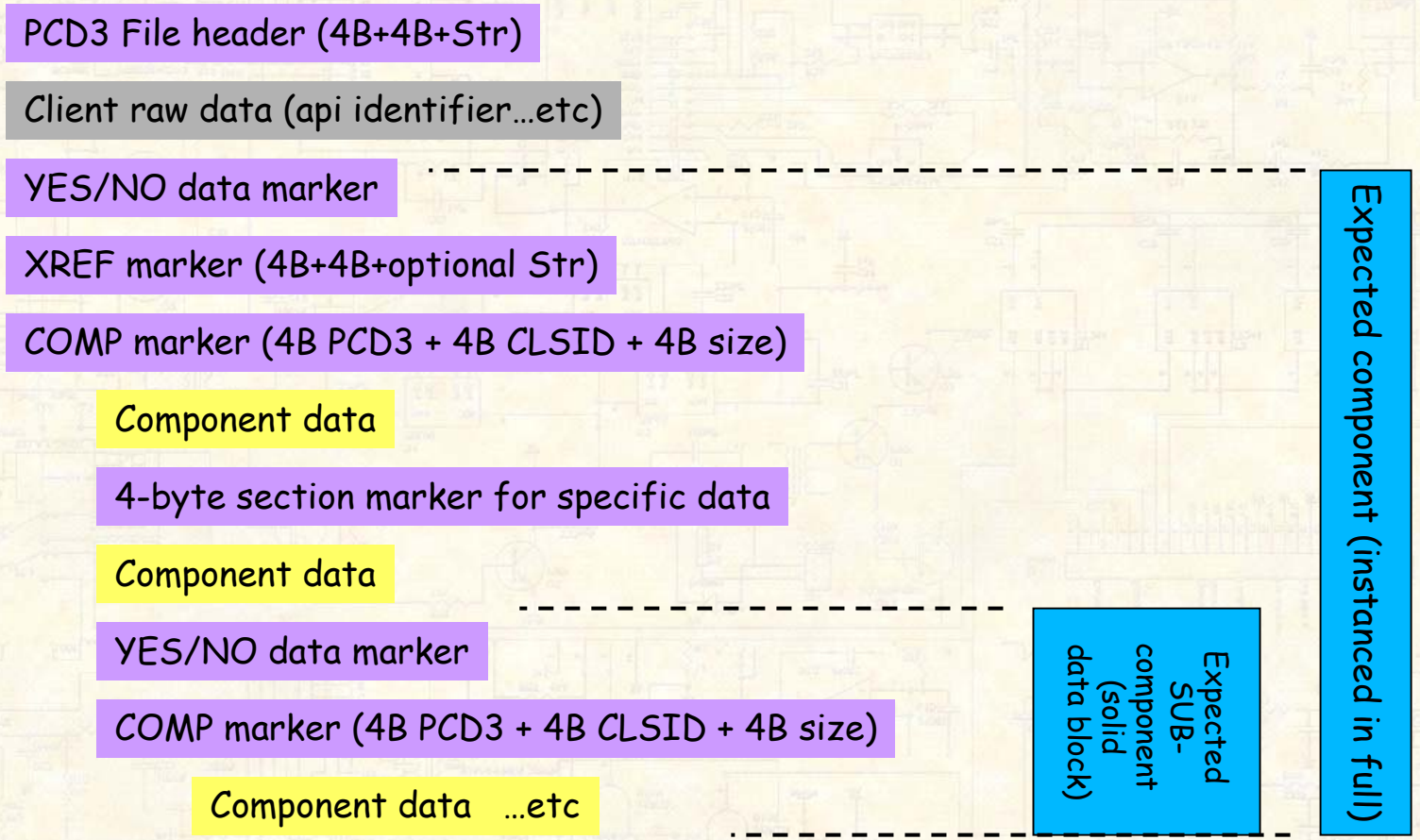
D108A - World Serialization

PCD3 format structure (2/2)

- "Solid block" data items without ID (e.g. arrays)
  - Data presence marker (4-byte YES/NO code). Used to indicate whether there is data, or whether the expected data item is NULL.
  - Component characterization marker (12-byte code made of 3x4-byte tags coding for a PCD3 chunk, the item CLSID and the item data size).
  - Binary data.
- Referenceable component data items with ID (e.g. objects)
  - Data presence marker (4-byte YES/NO code). Used to indicate whether there is data, or whether the expected data item is NULL.
  - Cross-reference marker (4-byte XREF/ENCL code, 4-byte component ID, resolution name for defined Ids).
  - Component characterization marker (12-byte code made of 3x4-byte tags coding for a PCD3 chunk, the item CLSID and the item data size).
  - Component data, which can in turn comprise simple data, component data chunks, and 4-byte section markers for validation and searching.

D108A - World Serialization

PCD3 format structure (example)



D108A - World Serialization

Two types of Principia components

- Defined components
  - Created while parsing the CFG file shipped with product.
  - Unique “individuals”, persistent, serve as templates (e.g. objects)
  - Have a positive ID that usually maps to a name-table name.
  - Managed via the Principia Api*Set collections.
  - Generally do not need to be serialized
- Instanced (or internal) components
  - Instanced internally by Principia or by the client
  - Usually make the most of the gameplay
  - Have a negative ID that is persistent
  - Managed via the Principia Usr*Set collections
  - Generally need to be serialized.
- Principia automatically serializes the sub-components and resolves cross-references processing high-level items.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Application names and namespaces

- Every Principia application has its own namespace where components are consistently referenced with a 4-byte ID. There are some basic rules to follow when saving, loading and moving data files to ensure proper reference resolution.
- Defined IDs are determined by the NAMES used in the configuration script, e.g. a texture is called TEX_RED or a character is called ROBO_BAD. If the same name is used to define these objects, they will be correctly loaded and resolved by any Principia application parsing PCD3 streams.
- Instanced IDs are internally generated. However, a Principia application using the PCD3 format stores all necessary info to resolve these at load time by following the cross-reference chain all the way up to a named ID. This ensures for compact, efficient and accurate storage of data.
- Many objects such as interactive characters have complex internal cross-referenced structures and methods. Saving and precisely loading these with Principia is a SNAP! The only thing you need to make sure is that you have used the same component NAMES in the configuration script between saving and loading applications (e.g. TEX_RED or ROBO_BAD). The same constancy requirement applies to state, signal, keyframe names ...etc.
- You do not need to keep file names or path locations the same, which is always a pain when working with other production environments.

D108A - World Serialization

Component serialization logic (loading)

- Has this component ID already been loaded?
  - Read cross-reference indicator tag and see if this is a cross-reference or an original component that is following.
- If Yes:
  - Resolve the ID read. For defined IDs, Principia uses the tag full string name to map to the application current name table. For IIDs, Principia uses a translation table linking file-space to api-space IIDs.
  - Find the component pointer in the Principia Usr/Api* collections.
  - Do not instance this component, just make a cross-reference.
- If Not:
  - Instance load of the component (creator from file, adds to Usr*), assign a new IID thereof and add it to the translation table.
- Unless the user wants to manually manage low-level component serialization, these tasks are done automatically.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Component serialization logic (saving)

- Has this component ID already been saved?
  - Test ID against Principia's internal database of already saved IDs
- If Yes:
  - Write yes-cross-reference 4-byte tag
  - Write ID and optional resolution name for defined IDs.
- If Not:
  - Write not-cross-reference 4-byte tag
  - Write ID and optional resolution name for defined IDs.
  - Save component into stream (using the ::Save) method
  - Update Principia database of already saved internal IDs
- All IDs are written in their 4-byte integer form. In addition, defined IDs are followed by their full script name.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Serialized sub-component instancing

- Most objects feature complex control data structures that may contain cross-references to other Principia components (e.g. state-animation controller maps of Biped1A). To help resolve some of these references, every object holds and serializes a copy of the prototype ID from which it derives.
- General rule A: Object sub-components that are or may be individually instanced are serialized in full. These include bounding boxes and kinexes, which are usually instanced, and principal geometry, materials and controllers which may be individually instanced.
- General rule B: Object sub-components that are not individually instanced are not serlialized. These are resolved by tracking the PID and copying its current associations. Examples include signal sets, behavior sets, state-component maps...etc.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Saving or loading world layers:

- Most of a Principia application contents are held in world layers. However, not all layer data needs saving: linked layers, rendering layers ...etc have no intrinsic data.
- Only layers defined with the LWT_SAVEABLE type tag will have their contents serialized by Principia.
- The layer properties and world layer structure defined in script are not saved by Principia (although these can be saved by the client application if necessary).
- Some components need property persistence rather than actual serialization (e.g. camera settings). These should be serialized by the client or saved as profile files.

D108A - World Serialization

Saving or loading separate components:

- When developing content, one usually wants to save individual components in a way that is detached from the application environment so that they can be read by any other application (e.g. when procedurally generating geometry, textures or animation tracks).
- Non-PCD3 targets: In the above case, the output file target should be a non-PCD3 target so that the content is saved in full and independent of cross-referencing. Most Principia procedures and ##SAVE commands assume non-PDC3 target by default (the user can over-ride this with flags).
- PCD3 targets: When developing game levels and save games, the natural output target should be a PCD3 file. This should be the task of the client master load/save procedure.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

- **Example: This is how Principia serializes the contents of any layer for you.**

```
void WX_Layer3R::Save (PX_File* pFile, int Flags)
BGN_CLASS_CODE{

   /* Auxiliary local variables */
   DECLARE_LOCAL  locale int         RefID = 0x00;

   /* Operation validation and nulls purge */
   if (!WX_Layer3R::TestType(LWT_SAVEDATA)) return;
   DEVMODE_TRAPERR( (Link) , Principia->TrapError("This layer should not be saved") );
   Contents->RemoveAny(NULL, false);

   /* PCD Save: Standard component prefix */
   Principia->PcdFileCompMark (pFile, CLSID_WLAYER3R, 0x00);

   /* PCD Save: Layer structural data and idents */
   ObjSet = Contents->ePtr;
   ObjNum = Contents->N;
   pFile->WrteLn(Contents->N);
   pFile->WrteLn(Contents->Max);

   /* PCD Save: Layer contents serialization */
   for (k=0; k<ObjNum; k++) begin{
       STDSERIALIZE_SAVEXREF_COMPONENT (pFile, ObjSet[k], RefID, Flags);
   endfor}

END_CLASS_CODE}
```

- **Simple… and the key is the use of Principia's serialization methods.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Example:

- This is how one would access a low-level component, such a animation keyframe #0 of the door sub-object of two buildings.
- Here, we make sure that these keyframes are not individually instanced (wasting memory and degrading performance).

```
+     ((MX_KinexF*) (((BX_Generic3C*) Layer[3]->Contents->ePtr[2])->CompOB[2]->Kinex))->KF->ePtr[0]         0x02868840
+     ((MX_KinexF*) (((BX_Generic3C*) Layer[3]->Contents->ePtr[0])->CompOB[2]->Kinex))->KF->ePtr[0]         0x02868840


And this is the 3-rd absolute bones keyframe for two different robot characters ...

+     ((MX_KinexF*) ((BX_Biped1A*) Layer[4]->Contents->ePtr[0])->KxRelBones)->KF->ePtr[3]  0x0286d280
+     ((MX_KinexF*) ((BX_Biped1A*) Layer[4]->Contents->ePtr[2])->KxRelBones)->KF->ePtr[3]  0x0286d280
```

Principia takes care of properly cross-referencing and instancing all sub-components of our world !

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Principia client serialization methods (1)

- Methods to be used by client for initiating/terminating serialization.
- Principia::PcdFileNew(char* fName, SX_String* fDesc) opens a new file for PCD3 format writing.
- Principia:: PcdFileOld(char* fName, SX_String* &fDesc) opens an existing file for PCD3 format loading and validates its contents.
- Principia::PcdSer_SaveInit() must be called by the client before starting to write PCD3 data.
- Principia::PcdSer_LoadInit() must be called by the client before starting to read PCD3 data.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Principia client serialization methods (2)

- Methods to be used by client code for saving components. Note that once a component is serialized with a *XREF* method, it must be loaded with the *XREF* loading method.
- STDSERIALIZE_SAVEXREF_COMPONENT(PF,ITM,SID,FLAGS) saves Principia ID-referenceable registered components items in PCD3 format. Most components (graphics, objects...etc) will be saved using this method.
- STDSERIALIZE_SAVEDATA_COMPONENT(PF,ITM,FLAGS) saves Principia components as non-referenceable, “monoblock” data items. Use this for simple components such as sets, strings, boxes... that occur once and locally.
- STDSERIALIZE_SAVEDATA_FULLARRAY(PF,TYPE,SZ,DAT,N,FLAGS) saves allocated arrays in PCD3 format with proper re-creation information.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Principia client serialization methods (3)

- Methods to be used by client code for restoring components. The LOADXREF methods are used for restoring items saved with the SAVEXREF method. The LOADDATA method is used for items saved with the SAVEDATA method.
- STDSERIALIZE_LOADXREF_COMPONENT(PF,TYPE,CAST,ITM,SID,FL,NDC) loads a Principia ID-referenceable component from a PCD3 stream. If the component is present, Principia will properly instance or cross-reference it in the current application pointer space.
- STDSERIALIZE_LOADXREF_ATOM(PF,CONS,CAST,ITM,SID,FL,NDC) loads a Principia ID-referenceable atom component from a PCD3 stream using the supplied constructor. Atom components are sets, strings...etc. Cross-referenced atoms cannot be loaded using LOADXREF_COMPONENT.
- STDSERIALIZE_LOADDATA_COMPONENT(PF,CONS,ITM,FL,NDC) loads a ID-less component from a PCD3 stream using the supplied CONS constructor, and assigns it an ID based on the current application ID-space.
- STDSERIALIZE_LOADDATA_FULLARRAY(PF,TYPE,SZ,DAT,N,FL,NDC) loads an array from a PCD3 stream, with proper reallocation.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Principia client serialization methods (4)

- These methods are useful for serializing components instanced in the client.
- STDSERIALIZE_LOADXREF_REFERENCE(PF,TYPE,CAST,ITM,SID,FL,NDC) loads any ID-referenceable component from a PCD3 stream without invoking standard creator instancing. This method is mostly used for serializing client-defined custom components (e.g. client-defined behaviors). It is essential that such custom components replicate the Principia collection management macros, and instance them with string names.
- STDSERIALIZE_WRTEDATA_DECL(PF,ITM,FL) saves any declared code variable that has a PX_File::WrteLn/ReadLn method. The corresponding loading method is *READDATA*.
- STDSERIALIZE_WRTEDATA_STRING(PF,STR,FL) performs an equivalent task for Principia SX_String. The corresponding loading method is *READDATA*.
- SX_Set::SaveComp() save-serializes sets the components of which are pointers to objects. This method is very useful for serializing collections of client objects. The corresponding loading data is SX_Set::LoadComp***().

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Principia client serialization methods (5)

- Several flags control the behavior of the serialization methods.
- <NDC> flag: If the component is not present, the NDC flag will determine how residual receiving pointer will be handled. If its value is <IFNONE_DEL> the item holder will be deleted. If the value is <IFNONE_NUL>, the item holder will be set to NULL. This flag is useful for determining how to manage memory with multiple re-loads.
- <FL> flag: A useful flag setting is FILEOPT_EXACTSIZE, which allocates the array or component to hold exactly the data present in the stream.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A - World Serialization

Principia low-level serialization methods

- It is very unlikely that you will need to use these methods yourself in the client application. You may use the PX_File:: ReadLn() and WrteLn() methods to access simple data.

```
int         PcdFileCompMark    (PX_File* pFile, int MarkCLSID, int ComSize);
int         PcdFileCompTest    (PX_File* pFile, int TestCLSID, bool ErrFail);
bool        PcdFileCompQuery   (PX_File* pFile, int &GetCLSID, int& GetSize);
void        PcdFileDataMark    (PX_File* pFile, bool IsData);
bool        PcdFileDataTest    (PX_File* pFile);
void        PcdFileCodeMark    (PX_File* pFile, int  Code);
bool        PcdFileCodeTest    (PX_File* pFile, int &Code, int RefCode, bool ErrFail);
void        PcdFileXrefMark    (PX_File* pFile, int  RefID, bool IsXref);
bool        PcdFileXrefTest    (PX_File* pFile, int &RefID);
void        PcdFileSaveID      (PX_File* pFile, int  RefID);
bool        PcdFileLoadID      (PX_File* pFile, int &RefID);
void        PcdFileSaveCHN     (PX_File* pFile, int  RefIX);
void        PcdFileLoadCHN     (PX_File* pFile, int &RefIX);

void        PcdSer_SaveInit    ();
void        PcdSer_LoadInit    ();

void        PcdSer_DBSavedMark (int RefID);
bool        PcdSer_DBSavedTest (int RefID);
int         PcdSer_DBTransFile (int RefID);

AX_Basic*   PcdGet_Math        (PX_File* pFile, int Flags, int SetID, void* Extended);
AX_Basic*   PcdGet_Basic       (PX_File* pFile, int Flags, int SetID, void* Extended);
AX_Kinex*   PcdGet_Kinex       (PX_File* pFile, int Flags, int SetID, void* Extended);
AX_Grafic*  PcdGet_Grafic      (PX_File* pFile, int Flags, int RefID, void* Extended);
AX_Object*  PcdGet_Object      (PX_File* pFile, int Flags, int RefID, void* Extended);
AX_Control* PcdGet_Control     (PX_File* pFile, int Flags, int RefID, void* Extended);
```

D108A - World Serialization

Principia PCD3 4-byte marker codes

- It is very unlikely that you will need to use these codes in the client application, but here they are for reference.

```
const int CHUNK_PCD3                = MAKEFOURCC('P','C','D','3')   ;//File marker: Principia V3 component prefix code
const int CHUNK_YDATA               = MAKEFOURCC('D','A','T','A')   ;//File marker: Data prsent tag (there is object)
const int CHUNK_NDATA               = MAKEFOURCC('N','U','L','L')   ;//File marker: Data absent tag (there is no object)
const int CHUNK_YXREF               = MAKEFOURCC('X','R','E','F')   ;//File marker: Xref prsent tag (object referenced)
const int CHUNK_NXREF               = MAKEFOURCC('E','N','C','L')   ;//File marker: Xref absent tag (object enclosed)
const int CHUNK_END                 = MAKEFOURCC('-','-','-','-')   ;//File marker: Data termination tag

const int CHUNK_GENARRAY            = MAKEFOURCC('L','I','S','T')   ;//File marker: Generic enumerated array
const int CHUNK_GENKINEX            = MAKEFOURCC('K','X','M','X')   ;//File marker: Generic kinex data
const int CHUNK_GENINTER            = MAKEFOURCC('O','I','N','T')   ;//File marker: Generic interactive object data
```

D108A - World Serialization

Optimal serialization sequence layout

- Defined by client:
- Open PCD3 file and write prefix data
- Call Principia::SerInit***() methods to reset the serialized ID databases and other internal management components.
- Call Principia::SerState***() methods to store clock, signal states and other general Principia states.
- Call Principia::SerUsr***() methods if desired to pre-store select groups of internally instanced components.
- Serialize desired client intrinsic variables and high-level Principia components (e.g. worlds...). The serialization of their sub-components will be done automatically.
- Principia provides a host of methods if you wish to deviate from this optimal sequence layout.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108A:Ex01 – World3A

File loader design

```
void WX_Layer3R::Save (PX_File* pFile, int Flags)
BGN_CLASS_CODE{

   /* Auxiliary local variables */
   DECLARE_LOCAL  locale int         RefID = 0x00;

   /* Operation validation */
   if (!WX_Layer3R::TestType(LWT_SAVEDATA)) return;
   DEVMODE_TRAPERR( (Link) , Principia->TrapError("This layer should not be saved") );

   /* PCD Save: Standard component prefix */
   Principia->PcdFileCompMark (pFile, CLSID_WLAYER3R, 0x00);

   /* PCD Save: Layer structural data and idents */
   ObjSet = Contents->ePtr;
   ObjNum = Contents->N;
   pFile->WrteLn(Contents->N);
   pFile->WrteLn(Contents->Max);

   /* PCD Save: Layer contents serialization */
   for (k=0; k<ObjNum; k++) begin{
       if (ObjSet[k]) then{
          RefID = ObjSet[k]->ID;
          if (Principia->PcdSer_DBTest(RefID)) then{
             Principia->PcdFileXrefMark (pFile, true);
             pFile->WrteLn(RefID);
          }else{
             Principia->PcdFileXrefMark (pFile, false);
             pFile->WrteLn(RefID);
             ObjSet[k]->Save(pFile, Flags);
             Principia->PcdSer_DBMark(RefID)
          endif}
       endif}
   endfor}

END_CLASS_CODE}
```

D108A:Ex01 – World3A

Implementation:

```
vs.1.1
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Chapter 1:08
Core Worlds
Principia v3.0
Reference Series
(c) Western Star
Entertainment Ltd, 2003-2006

D108B – WorldRT3A

Fasten your seatbelts! **World3A was basically an object container, suitable for arcade games, but little more. We will be now moving on to the "real" worlds.**

If you have tried to build your own game, you know of all the detail and process complexity in generating and rendering realistic worlds.

The coming Principia world objects enable a configuration and procedure driven approach to creating the incredibly complex structures behind a realistic application world.

You will need the full spectrum of your rendering and data structure knowledge to make the most of it.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

- **Basic WorldRT3A Geometry Structure**
  - Regular rectangular elevation grid (GridZ)
  - Regular rectangular node type code grid (GridT)
- **Basic WorldRT3A Tile Structure**
  - Multiple tile layers mapped to multiple world layers
  - Tiles encapsulated as AX_Object* / BX_Tile3A* items
  - Tiles typically arranged in rectangular array
  - Tiles typically regular rectangular meshes
  - Tile arrays and tiles themselves can have “voids”
- **Typical WorldRT3A Use**
  - Third person strategy games with overhead camera and relatively limited dynamic range.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

- **WorldRT3A runtime components:**
  - WX_WorldRT3A: world component
  - CX_WViewRT3A: native viewer
  - WX_Cursor3R: surface cell cursor
- **WorldRT3A generation procedures**
  - FX_WrldRT3A_Gen**: overall assembly procedures
  - FX_HField**: base heightfield generation procedures
  - FX_TerGen**: terrain elevation and DCT refining
  - FX_TexGen_DC**: DCT texture generation
- **Various procedures can be plugged in or out the generation process based on the desired results.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

- **D108B aims to show basic techniques for world creation (and not to make high-quality worlds)**
  - **Ex01: Basic world data organization and rendering, DCT map control and terrain coding.**
  - **Ex02: Procedural terrain generation workflow, natural UV coordinates, composite textures.**
  - **Ex03: Per pixel lighting, flowing water and moving water surfaces, tile material-palletization.**
  - **Ex04: Fast shadow maps, basic object management path-finding and collision resolution.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

Foundation concepts in Ex01:

- Generation procedure basic inputs: Z, UV and DCT maps (16-bit encoding), with simple procedural generation of Z and DCT data.
- Tile layer to world layer mapping with tile DCT masks: (a) carving mask, (b) content visible mask and (c) cover-overlaid mask.
- Terrain variety enhanced via multiple tile layer overlays
- Fixed world-space tiles, UV origination codes and UV sets
- Basic partitioning and visibility testing ops, LoD layer switching
- Static water with fine-scale animation and Z-depth encoding
- Type grid assembly from maps
- Basic world queries and cell cursor
- Wire-frame grid display
- Object distribution and standard footprint coding

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

Foundation concepts in Ex02:

- Terrain generation workflow with external
  - **geological feature simulation procedures**
  - **detail correction procedures**
  - **for Z, UV and DCT generation.**
- Tile rendering refinements
  - **UV mirroring alternation**
  - **Pure texture pre-mixing into composite base terrain textures to enable single-layer terrain rendering instead of multi-layer.**
- Natural UV data channel for steep slopes.
- Input (procedural raw) and output (Z,UV,DCT) maps.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

Foundation concepts in Ex03:

- Terrain enhancement techniques
  - **Bump-mapping**
  - **Scatter-based material paletization across multiple layers (each layer features different composite base textures).**
  - **Quad triangulation control**
  - **Texture optimal 256x256 sizing.**
- Dynamic rivers with programmable channels for water tiles:
  - **Depth**
  - **River flow natural UV**
  - **Riverinity factor**
  - **Procedural generation of water data from eroded terrain base**
- Per-vertex water wave animation.
- Basic rim visibility management

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

Foundation concepts in Ex04:

- Shadows! A fast binary-state PSM algorithm with smooth shadows.
- Mixed object type layer organization with extensive operation use and structuring. Managing billboard rendering in 3D worlds.
- Character selection and overall pointer interaction with world.
- Character path-finding and path display.
- Character path-modification collision resolution.
- Programmable placer and cursor.

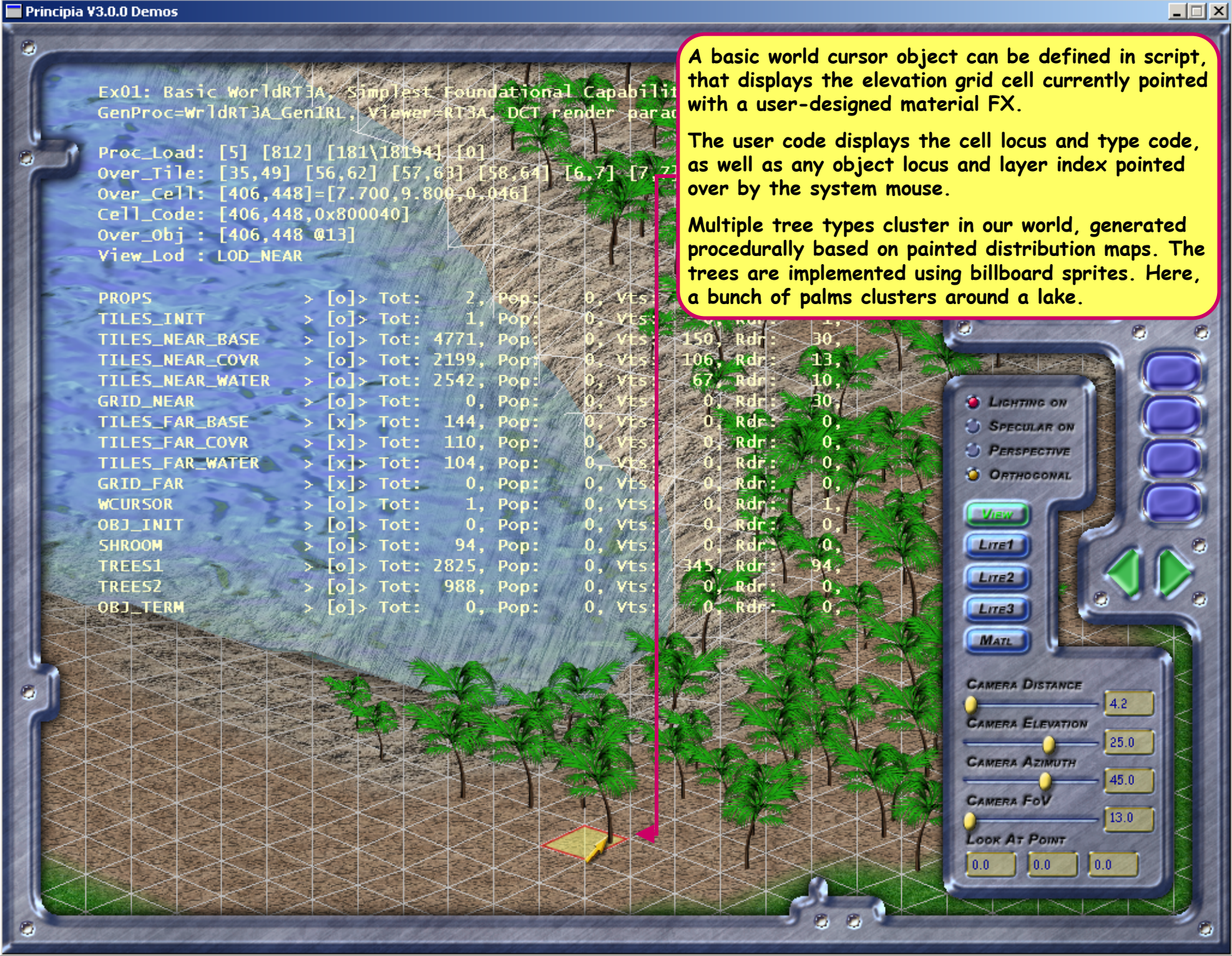
Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Proc_Load: [11] [1209] [331\13423] [0]
Over_Tile: [19,42] [20,43] [21,43]
Over_Cell: [149,304]=[-5.150,2.600,0.000]
Cell_Code:
Over_Obj :
View_Lod : LOD_NEAR
PROPS > [o]> Tot: 2, Pop: 0, Vts: 0, Rdr: 2,
TILES_INIT > [o]> Tot: 1, Pop: 0, Vts: 0, Rdr: 1,
TILES_NEAR_BASE > [o]> Tot: 4771, Pop: 0, Vts: 374, Rdr: 116,
TILES_NEAR_COVR > [o]> Tot: 2199, Pop: 0, Vts: 210, Rdr: 57,
TILES_NEAR_WATER > [o]> Tot: 2542, Pop: 0, Vts: 116, Rdr: 38,
GRID_NEAR > [x]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_BASE > [x]> Tot: 144, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_COVR > [x]> Tot: 110, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_WATER > [x]> Tot: 104, Pop: 0, Vts: 0, Rdr: 0,
GRID_FAR > [x]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
WCURSOR > [o]> Tot: 1, Pop: 0, Vts: 0, Rdr: 0,
OBJ_INIT > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
SHROOM > [o]> Tot: 94, Pop: 0, Vts: 34, Rdr: 10,
TREES1 > [o]> Tot: 2825, Pop: 0, Vts: 194, Rdr: 29,
TREES2 > [o]> Tot: 988, Pop: 0, Vts: 137, Rdr: 78,
OBJ_TERM > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
FPS: 21.3
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
0.0
0.0
0.0
This basic world has plains, grass, mountains, gently undulating shiny water, scattered trees, rocks and mushrooms … and a performance display showing the stats behind every layer of this world.


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Proc_Load: [50] [267] [124\361] [0]
Over_Tile: [24,49] [2,6]
Over_Cell: [120,259]=[-6.600,0.350,0.153]
Cell_Code: [120,259,0x40]
Over_Obj :
View_Lod : LOD_FAR
PROPS > [o]> Tot: 2, Pop: 0, Vts: 0, Rdr: 2,
TILES_INIT > [o]> Tot: 1, Pop: 0, Vts: 0, Rdr: 1,
TILES_NEAR_BASE > [x]> Tot: 4771, Pop: 0, Vts: 0, Rdr: 0,
TILES_NEAR_COVR > [x]> Tot: 2199, Pop: 0, Vts: 0, Rdr: 0,
TILES_NEAR_WATER > [x]> Tot: 2542, Pop: 0, Vts: 0, Rdr: 0,
GRID_NEAR > [x]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_BASE > [o]> Tot: 144, Pop: 0, Vts: 50, Rdr: 49,
TILES_FAR_COVR > [o]> Tot: 110, Pop: 0, Vts: 41, Rdr: 41,
TILES_FAR_WATER > [o]> Tot: 104, Pop: 0, Vts: 32, Rdr: 31,
GRID_FAR > [x]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
WCURSOR > [x]> Tot: 1, Pop: 0, Vts: 0, Rdr: 0,
OBJ_INIT > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
SHROOM > [x]> Tot: 94, Pop: 0, Vts: 0, Rdr: 0,
TREES1 > [x]> Tot: 2825, Pop: 0, Vts: 0, Rdr: 0,
TREES2 > [x]> Tot: 988, Pop: 0, Vts: 0, Rdr: 0,
OBJ_TERM > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
FPS: 21.3
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
26.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
0.0
0.0
0.0
At far zooms, a different set of layers is switched on, having fewer tiles and coarser geometry/materials. This enables fast renders over large world extents.


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Cap
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render
Proc_Load: [14] [1444] [861\18194] [0]
Over_Tile: [35,36] [35,37] [36,37] [37,38] [37,3
Over_Cell: [266,277]=[0.700,1.250,0.267]
Cell_Code: [266,277,0x40]
Over_Obj :
View_Lod : LOD_NEAR
PROPS > [o]> Tot: 2, Pop: 0, Vts: 0, Rdr: 2,
TILES_INIT > [o]> Tot: 1, Pop: 0, Vts: 0, Rdr: 1,
TILES_NEAR_BASE > [o]> Tot: 4771, Pop: 0, Vts: 454, Rdr: 228,
TILES_NEAR_COVR > [o]> Tot: 2199, Pop: 0, Vts: 289, Rdr: 138,
TILES_NEAR_WATER > [o]> Tot: 2542, Pop: 0, Vts: 73, Rdr: 50,
GRID_NEAR > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 228,
TILES_FAR_BASE > [x]> Tot: 144, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_COVR > [x]> Tot: 110, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_WATER > [x]> Tot: 104, Pop: 0, Vts: 0, Rdr: 0,
GRID_FAR > [x]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
WCURSOR > [o]> Tot: 1, Pop: 0, Vts: 0, Rdr: 1,
OBJ_INIT > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
SHROOM > [o]> Tot: 94, Pop: 0, Vts: 35, Rdr: 4,
TREES1 > [o]> Tot: 2825, Pop: 0, Vts: 347, Rdr: 154,
TREES2 > [o]> Tot: 988, Pop: 0, Vts: 102, Rdr: 55,
OBJ_TERM > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
At either LoD, the geometry can be displayed in overlaid wireframe mode by pressing the "G" key. This is extremely simple to do by simply re-rendering one of the layers with wireframe RS material.
A linked layer to the base terrain layer is used to do the re-render without any CPU overhead. This enables the use to also see where base terrain tiles have been "knocked out", based on overlay coverage assessment.
Lighting On
Specular On
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
10.8
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
0.0
0.0
0.0

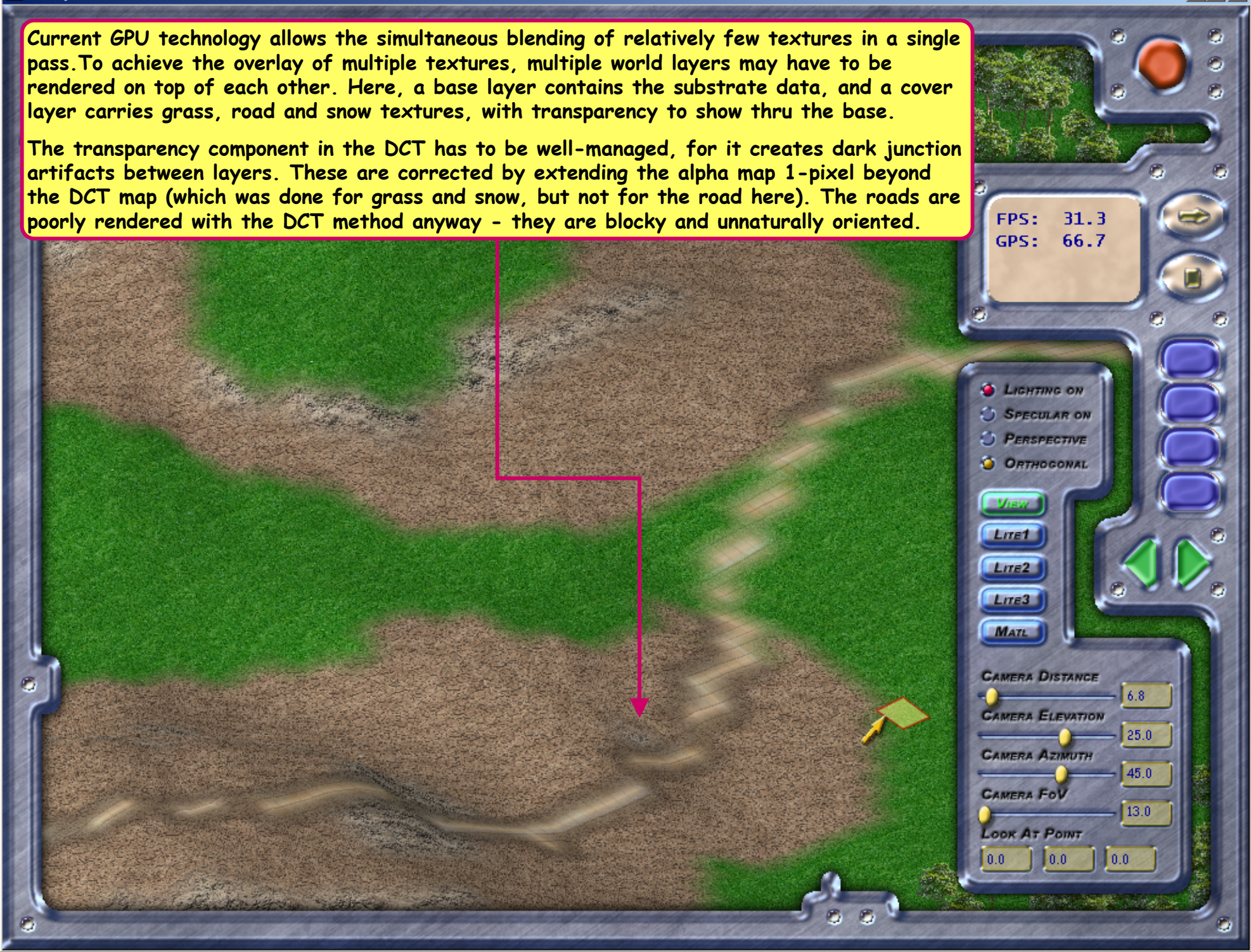

Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
FPS: 21.3
GPS: 21.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
34.9
Camera Elevation
The detailed layer statistic data is automatically generated by Principia. Its display and toggling via the "D" key is managed in the user code, which illustrates basic access to world layer data.
Because of the data linked, grid display does not carry over water. Note that some specular lighting usually enhances world atmosphere.


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilit
GenProc=WrldRT3A_GenIRL, Viewer=RT3A, DCT render parat
Proc_Load: [5] [812] [181\18194] [0]
Over_Tile: [35,49] [56,62] [57,63] [58,64] [6,7] [7,
Over_Cell: [406,448]=[7.700,9.800,0.046]
Cell_Code: [406,448,0x800040]
Over_Obj : [406,448 @13]
View_Lod : LOD_NEAR
PROPS > [o]> Tot: 2, Pop: 0, Vts:
TILES_INIT > [o]> Tot: 1, Pop: 0, Vts:
TILES_NEAR_BASE > [o]> Tot: 4771, Pop: 0, Vts: 150, Rdr: 30,
TILES_NEAR_COVR > [o]> Tot: 2199, Pop: 0, Vts: 106, Rdr: 13,
TILES_NEAR_WATER > [o]> Tot: 2542, Pop: 0, Vts: 67, Rdr: 10,
GRID_NEAR > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 30,
TILES_FAR_BASE > [x]> Tot: 144, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_COVR > [x]> Tot: 110, Pop: 0, Vts: 0, Rdr: 0,
TILES_FAR_WATER > [x]> Tot: 104, Pop: 0, Vts: 0, Rdr: 0,
GRID_FAR > [x]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
WCURSOR > [o]> Tot: 1, Pop: 0, Vts: 0, Rdr: 1,
OBJ_INIT > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
SHROOM > [o]> Tot: 94, Pop: 0, Vts: 0, Rdr: 0,
TREES1 > [o]> Tot: 2825, Pop: 0, Vts: 345, Rdr: 94,
TREES2 > [o]> Tot: 988, Pop: 0, Vts: 0, Rdr: 0,
OBJ_TERM > [o]> Tot: 0, Pop: 0, Vts: 0, Rdr: 0,
A basic world cursor object can be defined in script, that displays the elevation grid cell currently pointed with a user-designed material FX.
The user code displays the cell locus and type code, as well as any object locus and layer index pointed over by the system mouse.
Multiple tree types cluster in our world, generated procedurally based on painted distribution maps. The trees are implemented using billboard sprites. Here, a bunch of palms clusters around a lake.
Lighting On
Specular On
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
4.2
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
13.0
Look At Point
0.0
0.0
0.0

Principia ¥3.0.0 Demos

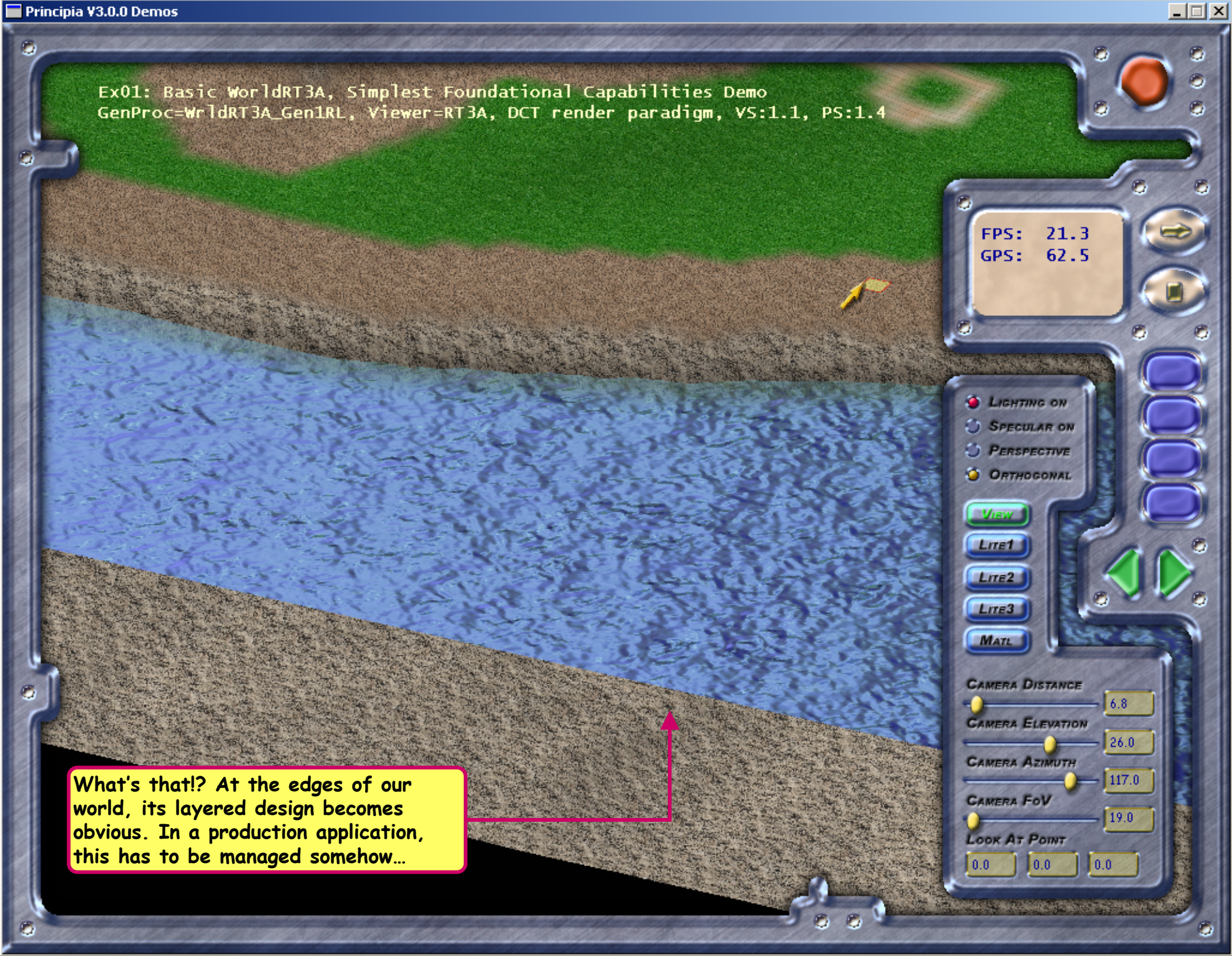
Current GPU technology allows the simultaneous blending of relatively few textures in a single pass.To achieve the overlay of multiple textures, multiple world layers may have to be rendered on top of each other. Here, a base layer contains the substrate data, and a cover layer carries grass, road and snow textures, with transparency to show thru the base.

The transparency component in the DCT has to be well-managed, for it creates dark junction artifacts between layers. These are corrected by extending the alpha map 1-pixel beyond the DCT map (which was done for grass and snow, but not for the road here). The roads are poorly rendered with the DCT method anyway - they are blocky and unnaturally oriented.




Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 21.3
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Many static objects are spread thru our world. While there is some structure to their spread, we have some way to go before there is a truly realistic-looking distribution.
Keep in mind, that a realistic distribution may not be in the best interest for strategy games, where concentration and some stylistic distribution can enhance gameplay,


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 21.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
In addition to the elevation grid, a binary type grid is overlaid over WorldRT3A. This type encodes up to 32 simultaneous conditions for any mesh cell, such as whether it is on a plain, or has a tree in it.The type grid is generated procedurally with the rest of the world
The type grid can be used by Principia (and the application) to control behavior. Here, the world cursor is hidden when the pointer is over a mountain. Many such interactive behaviors are easily configured in script, without coding.


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 21.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.8
Camera Elevation
26.0
Camera Azimuth
117.0
Camera FoV
19.0
Look At Point
0.0
0.0
0.0
What's that!? At the edges of our world, its layered design becomes obvious. In a production application, this has to be managed somehow…


Principia V3.0.0 Demos
Ex01: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WorldRT3A_GenIRL , Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 21.3
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
10.0
Shouldn't the trees be bigger than the mushrooms? …despite all the little shortcomings, keep in mind that we have created a rather large, reasonably rendered 3D game world VERY FAST and WITHOUT A SINGLE LINE OF CODE!
And in the demos to come, we will take care of all the remaining little details.

D108B:Ex01 – WorldRT3A

- **Objectives**
  - Show the basic concepts and techniques for layered worlds using WorldRT3A, its viewer WViewRT3A and generating procedure FX_WrldRT3A_Asm1RL.
- **Requirements and Implementation**
  - Basic world with variegated terrain types and scattered objects based on distribution control textures layering.
  - Showcase the essential design and configuration controls for the RT3A world component family group.
- **FX_WrldRT3A_Asm1RL is the simplest procedure for generating landscaped worlds with water, variegated land surfaces and object distributions.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **WorldRT3A basic structural features for Ex01:**
  - Multilayer tiled geometry using DCT and PS1* shaders to render variegated terrain with rock, grass ...etc, mapped to features such as roads, mountains ...etc.
  - Animated single-level pool-type water surfaces with animation, simplest reflectivity and transparency.
  - Terrain type coding based on procedural DCT variegation and on presence of material objects.
  - Scattering of basic sprite and 3D objects such as different tree types, rocks ...etc, sensitive to type.
- **Image maps drive much of the terrain and object placement logic in the RL1 procedure. Visualizing these is very helpful first design step.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

WorldRT3A basic interactive features for Ex01:

- Enable basic camera navigation controls, and toggle display of a tile mesh wireframe grid (shows the use of world layer property toggle method).
- Allocate true objects to a space partition array. This is used to accelerate object interaction management. We use a simple built-in supercell partition algorithm here.
- Display a grid cursor mini-tile as the pointer hovers above the surface (shows the use of built-in viewer tracking capabilities and of reference scalar fields).
- Display the index and SP cell ID of true objects over which the pointer is located (same as above).
- Display optional rendering statistics for each world layer for debugging and performance tuning.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

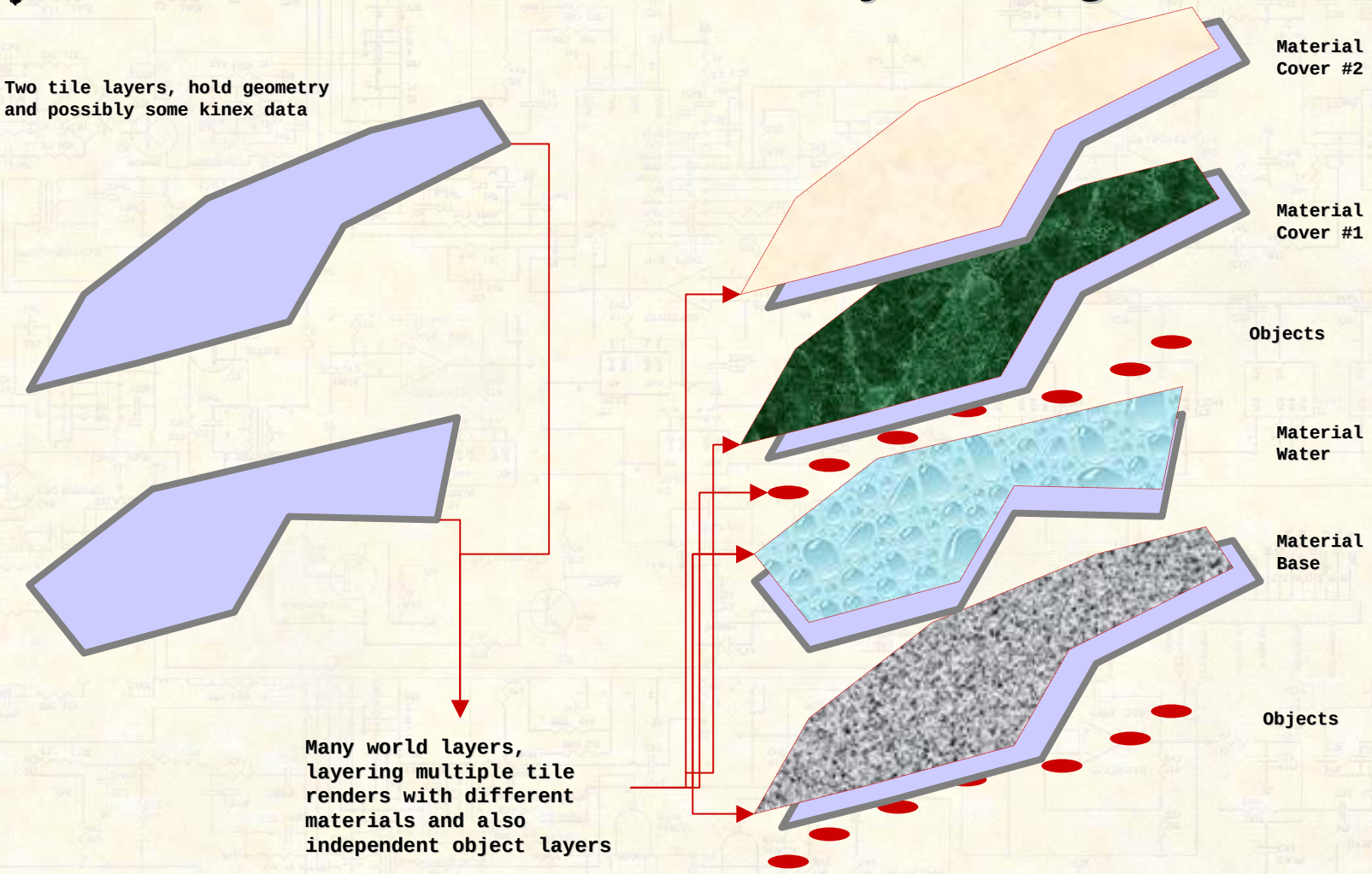
WorldRT3A basic acceleration features for Ex01:

- Tile level: skip generation of tile mesh points (or entire tiles) for non-displayable portions of the tile (based on an carving function test at generate time). Used for partial surfaces such as water cover.
- World layer level: skip incorporation of tiles to a given layer object queue if these tiles have no visible data (based on DCT transparency over tile). Used for overlay layers with sparse data (e.g. roads, snow layers...etc).
- World layer level: skip incorporation of tiles to a given layer if these tiles will be opaquely overlaid by other layers having the same tile set (based on DCT transparency over tile). Used for accelerating render of deep layers such as base rock ...etc.
- World layer level: sort the associated tile set from front to back to reduce the depth complexity of the scene.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Basic world tile and render layer design



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL Workflow, Ex Procedure

- Terrain type definitions (up to 32 type codes defined as binary masks for grid codes)
- ZF: Elevation field(s) synthesis. Basis for all subsequent generation processes.
- DCT: Generation of DCT for each world layer based on (a) procedural functions or (b) external data import. Even if DCTs may not be used in later rendering, they are employed by the procedure to help eliminate unused tiles.

FX_WrldRT3A_Asm1RL Workflow, In Procedure

- TILE: Tile layer generation based on procedural functions evaluated at regular XY positions, typically based on the elevation field.
- ZGRID: Elevation grid generation based on procedural function, typically clipped elevation field to represent terrain and water level.
- TGRID: Type grid generation, based on maps, procedural function evaluation and/or rule sequence evaluation.
- WLMAP: Tile layer mapping to world layers under DCT control.
- MTLMAP: Tile-material mapping and tile modification to enhance textural richness.
- OBJ: Procedural object distribution, considering TGRID and logical procedural function tests, typically based on ZF. In this stage, TGRID can be modified based on object presence and footprint. Multiple passes of this step are possible.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL, DCT Paradigm:

- How to render a patchwork of multiple texture types over tiled terrain without tiling artifacts? The industry standard approach uses DCT textures sampled over UV_W to blend several small terrain type textures repeatedly tiled and sampled over UV_T.
- The blend is carried in the pixel shader of the layer material, and is implementation-dependent. It typically blends three types of terrain textures based on the RGB values of the DCT. If the terrain has more than three types of textures, multiple layers with different DCT and base textures will be required.
- The DCT information is supplied to the generation procedure in order to (1) eliminate renders of empty tiles which do not involve any of the base textures of a given layer, and (2) generate a code field internal to WorldRT3A that contains information on the underlying terrain nature.
- All the information on terrain type nature is encoded in the DCT, which are generated externally to the procedure (e.g. hand painting or procedural texture generation).

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL, Structural Tiles:

- Multiple, ordinal-index tile layers. Each tile layer can be mapped several times to different world layers.
- Tiles form a regular rectangular NQX.NQY array, although some tiles in the array may be skipped from addition to the respective world layer if they carry no image content.
- Tiles are BX_Generic3A objects. Each tile geometry is a separate XYZ+T01+Optional vertex stream with structure that can be different for each tile layer.
- Each tile geometry is a basic rectangular NVX.NVY vertex array, although for some tiles, the array may have an incomplete irregular geometry if it has no image content.
- Each tile layer is controlled by three essential parameters: elevation function FZ, carving function FL and distribution control texture DCT.
- Nothing in the procedure assumes particular terrain types or rendering methods. The user has full flexibility to define tile properties and map them to world layers with different data interpretation and rendering methods.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL, Principal Tile Controls:

- Main coordinate systems for WrldRT3A are (1) real XY system, (2) individual tile unit UV_T system repeated over world plane, and (3) entire world plane unit UV_W system.
- Elevation function $FZ(UV_W)$: Generates the Z channel for tile geometry data. Vertex texture channels can take the UV_T and UV_W coordinates above for multi-textured terrain rendering.
- Carving function $FL(UV_W)$: Used to generate irregular tiles by carving away triangle geometry from the tile where not needed, such as water tiles that will never be rendered because they are over land, or road tiles that will never be rendered because they do not contain any road. The use of the carving function is controlled by the carving mask mode flags.
- Visibility distribution control texture $DCT(UV_W)$. Does not affect tile shape, but if the TileNUL parameter is set and the texture has no content over the entire tile, then this tile will not be added to the world layer which maps to this tile layer.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **FX_WrldRT3A_Asm1RL, Tile source data:**
  - Scripted mode: The tile data streams are defined in scrip – as functions or as external grid data files - and linked to the appropriate script tag sockets of the ASM1RL procedure. This mode is suitable for static worlds with offline designer workflows, such as those used in cinematic production.
  - Embedded mode: the tile data is generated by the client and passed as SX_Grid1R objects. This mode is used in real-time world generation client applications such as world editors.
- **FX_WrldRT3A_Asm1RL, Tile carving:**
  - Tile carving is controlled by a carving mask function (scripted mode) or a mask grid (embedded mode). In either case, a positive non-zero mask value signals that the vertex is to be retained.
  - Carving mask mode = EXACT_TRIG_MASK. The carved tile will be formed by triangles with all three vertices covered by the mask.
  - Carving mask mode = EXACT_QUAD_MASK. The carved tile will be formed by quads with all four vertices covered by the mask.
  - Carving mask mode = MAJOR_QUAD_MASK. The carved tile will be formed by quads with at least three vertices covered by the mask.
  - Carving mask mode = XTEND_TRIG_MASK. The carved tile will be formed by quads with at least one vertex covered by the mask.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL, General UV Systems

- UVSYS_TILE: Is the UV_T system, tiled repeatedly over the world surface, over which terrain textures are sampled. The tile coordinates are repeated in a [0..1][0..1][0..1]... sequence using simple linear periodicity.
- UVSYS_ALTTILE: A variant of the UV_T system, whereby the tile coordinate unit spans are repeated using the [0..1][1..0][0..1]... sequence. This allows to use textures that are not perfectly periodic.
- UVSYS_RNDTILE: A variant of the UV_T system, whereby the tile coordinate unit spans are repeated using a random [0..1][0..1][1..0]... sequence. With perfectly periodic textures, this minimizes repeat artifacts.
- UVSYS_WRLD: Is the UV_W system, which maps the world plane to the unit square. This is the native coordinate system for DCT texture design.
- UVSYS_IMPORT: Is a local curvilinear UV system that is based on the isocontours of a given elevation field. This system is used for texture mapping of waves, escarpments ...etc. This system typically has to be procedurally generated outside the world and imported via a map file, or via a grid object.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL, Terrain Scalar Fields:

- Z: Elevation scalar field over UV_W/XY. Used by the engines to navigate world cursors, position objects over the world ...etc. This field may or may not be related to the tile elevations.
- T: Type code field over UV_W/XY with binary combination of terrain types. Used by the engines to determine the position validity for various operations, based on what type of surface may be underneath. It is expected for most user applications to interact with the Z and T fields.
- FX_WrldRT3A_Asm1RL allows up to 32 different terrain types to be encoded in the code field. The procedure is agnostic as to the terrain physical type encoded by the DCT. It does not care that a DCT RGB pixel encodes roads, water, mud, rock, grass, talus...etc.
- Tile boxes: Each tile carries its own bounding box in world space coordinates, used for fast visibility testing. The <Tile_BoxOverride> tag enables to override the elevation limits of the box relative to those encoded in the tile geometry Z-channel. This tag is used for water tiles, where the Z-channel may carry depth data, but the tile is rendered at Z=0.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

FX_WrldRT3A_Asm1RL, Scalar Source Data:

- Scripted mode, Z field: Defined by evaluating the <GridElevationF>(u,v) function provided in script at a LoD set by the <GridTileSource> layer.
- Embedded mode, Z field: Copied from the external tile elevation field Aex_GridZ[GridTileSource] that must be connected to the procedure. This field can be optionally modulated by evaluating GridElevationF(u,v,AexZ).
- Scripted/embedded mode, T field, <GridTypeAsmMap> enables to map the RGBA channels of a series of DCT textures to set T flags. In scripted mode, <GridTypePreclear> clears the type field.
- Scripted/embedded mode, T field, <GridTypeRuleMap> generates or updates the T-field using a sequence of rules covered in D108C. The rules feature a series of functions F(u,v,z,g,q). In embedded mode, z (elevation) g (degree slope) and q (modifier) are taken from the external data field Aex_GridZ[GridTileSource], Aex_GridG and Aex_GridQ, which must be connected to the procedure. In scripted mode, z and g are taken from GridElevationF.
- Embedded mode, T field: The type field is copied from the external field Aex_GridT which must be connected to the procedure.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

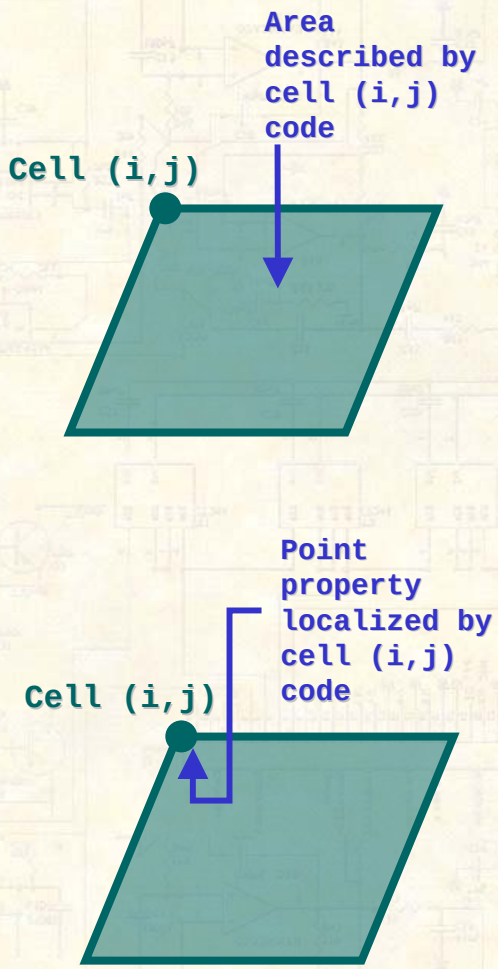
FX_WrldRT3A_Asm1RL, Tile Kinex Options:

- <Tile_Kinex>: The tag instructs the generator to create an individual kinex for every tile based on the prototype supplied. This tag has very important implications for the rendering design and type of tiles used.
- Non-null tile kinex: In this case, the kinex transform will contain the tile base position in the world, and will be applied before rendering the tile. The tile data will be generated in *model* space. Tile kinexes are used when separate XY and Z streams are used to save bandwidth.
- Null tile kinex: The tiles contain no kinexes and their vertices XYZ streams will be generated in full in the world coordinate system. Many techniques such as tile carving, tile triangle optimization ...etc. require tiles to be generated in full and not be stream-separated. The decision to use stream-separate tile design or not depends on the world design.
- Null tile kinex rendering initialization: In most cases (such as here) the world layering design must set the appropriate shader constants (usually via a prefix layer object) before rendering the tiles.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Type code field. Can encode up to 32 overlapping terrain content types encoded using name-mapped binary masks. There are two encoding modes:**
  - Cell-referencing. The code at node (i,j) describes the next cell contents. In this paradigm, objects are typically placed mid-cell, and elements such as roads run over the cell surface.
  - Node-referencing. The code describes the node point contents. Objects are typically placed at nodes and motion checks happen along edges.
- **WorldRT3A is encode mode-agnostic, as long as the mode is used consistently across the application. Node referencing is a bit faster.**



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **All this will be made tangible by examining the implementation of the world, section by section:**
  1) Terrain types definition
  2) World layer structure definition (pulled forward)
  3) Elevation definition
  4) DCT definition for render and tile generation
  5) Layer materials
  6) Object scattering procedures
  7) World generation procedure
  8) World viewer set-up
  9) User application code (informational display)

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 1: Terrain type codes used in our world. Note they must generally be binary superposable. Up to 32 codes are supported in the RT3A family.**

```
##define <NAME_VAL>       =  (  TC_WATER , 0x00000001 )  ' Water, elevation below zero
##define <NAME_VAL>       =  (  TC_MOUNT , 0x00000002 )  ' Mountain, tile elevation too high

##define <NAME_VAL>       =  (  TC_CLIFF , 0x00000010 )  ' High slope terrain at all but very low elevation
##define <NAME_VAL>       =  (  TC_ROCK  , 0x00000020 )  ' Moderate slope terrain at low elevation and low slope at high elevation
##define <NAME_VAL>       =  (  TC_SOIL  , 0x00000040 )  ' Low slope terrain at all except high elevations

##define <NAME_VAL>       =  (  TC_GRASS , 0x00000100 )  ' Overcover, low slope at low and moderate elevations over soil
##define <NAME_VAL>       =  (  TC_SNOW  , 0x00000200 )  ' Overcover, low slope at high elevations
##define <NAME_VAL>       =  (  TC_ROAD  , 0x00000400 )  ' DCT
##define <NAME_VAL>       =  (  TC_TREE  , 0x00800000 )  ' Procedural object placement
##define <NAME_VAL>       =  (  TC_LUMPS , 0x00008000 )  ' Procedural object placement
##define <NAME_VAL>       =  (  TC_NPC   , 0x00004000 )  ' Procedural object placement
##define <NAME_VAL>       =  (  TC_BUILT , 0x00002000 )  ' User
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 2: World layer design. The fundamental design decisions must be made beforehand:**
  - Two levels of detail based on rendered extent.
  - Built-in 12x12 block simple contents partitioning. This is determined by tile counts and object distributions. The tile count in each partition at any LoD must be an integer.
  - Tiles to have full XYZ streams. This means more vertex data but no individual tile kinexes. A structural dummy object is required to set tile transform shader constants prior to render.
  - Two land layers (substrate+cover) and one water layer.
  - Substrate layer to use 3channel non-alpha DCT.
  - Cover layer to use 3channel+alpha DCT.
  - Water layer to use animated normal and reflection mapping
  - Water layer to use tile elevation as depth FX key.
  - Separate tiles for land and water layers, with elimination of rendering and storage for tiles that have no visible content.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 2: World tile layers and operations.

- Simple visibility testing does not use state flag optimization.
- Partitioning done once, for all objects are static.
- Each layer has its own tiles, with partitioning and visibility testing operations.
- Note material setting and types for each layer.
- Grid layer is a link to a tile layer that is already partitioned and visi-tested at render time.
- Simple design has 10x the frame rate of World3A.

```
##define (OP_VISITEST) as <OP_VISITEST_3R>
   #tag Method    = (           NONE )
   #tag TstObjVS  = (             NO )
   #tag SetObjVS  = (             NO )

##define (OP_PARTFIX) as <OP_PARTSECT_2R>
   #tag Frequency = (           NONE )


##define (L_TILES_G1N) as <WORLDLAYER_3R>
   #tag Name      = "TILES_NEAR_GBASE"
   #tag Type      = (       LWT_TILE )
   #tag Type      = (     LWT_PARTED )
   #tag MatInit   = (  ML_GRND_EX01 )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_PARTFIX )
   #tag Operation = (    OP_VISITEST )
   #tag Count     = (           1024 )

##define (L_TILES_G2N) as <WORLDLAYER_3R>
   #tag Name      = "TILES_NEAR_GCOVR"
   #tag Type      = (       LWT_TILE )
   #tag Type      = (     LWT_PARTED )
   #tag MatInit   = (  ML_COVR_EX01 )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_PARTFIX )
   #tag Operation = (    OP_VISITEST )
   #tag Count     = (           1024 )

##define (L_TILES_W1N) as <WORLDLAYER_3R>
   #tag Name      = "TILES_NEAR_WATER"
   #tag Type      = (       LWT_TILE )
   #tag Type      = (     LWT_PARTED )
   #tag MatInit   = (  ML_WATR_EX01 )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_PARTFIX )
   #tag Operation = (    OP_VISITEST )
   #tag Count     = (           1024 )

##define (L_GRIDN) as <WORLDLAYER_3R>
   #tag Name      = "TILES_NEAR_GRID"
   #tag Type      = (       LWT_HIDE )
   #tag Type      = (       LWT_TILE )
   #tag Type      = (     LWT_PARTED )
   #tag Link      = (    L_TILES_G1N )
   #tag MatInit   = (  ML_GRID_EX01 )
   #tag MatTerm   = (           NONE )
   #tag Count     = (              8 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 2: World object layers definition.

- The materials here are carried within each object.
- While the Principia runtime optimizes redundant material calls automatically, a better approach (shown later) would be to sort by material.
- Object layers need only the LWT_PARTED type hint. Each layer contents are partitioned once, visibility tested at each frame, and rendered.

```
##define (L_OBJ_SOLIDS) as <WORLDLAYER_3R>
   #tag Name      = "OBJ_SOLIDS"
   #tag Type      = (    LWT_PARTED )
   #tag Operation = (    OP_PARTFIX )
   #tag Operation = (   OP_VISITEST )
   #tag Count     = (           512 )

##define (L_OBJ_TREES1) as <WORLDLAYER_3R>
   #tag Name      = "OBJ_TREES1"
   #tag Type      = (    LWT_PARTED )
   #tag Operation = (    OP_PARTFIX )
   #tag Operation = (   OP_VISITEST )
   #tag Count     = (          4096 )

##define (L_OBJ_TREES2) as <WORLDLAYER_3R>
   #tag Name      = "OBJ_TREES2"
   #tag Type      = (    LWT_PARTED )
   #tag Operation = (    OP_PARTFIX )
   #tag Operation = (   OP_VISITEST )
   #tag Count     = (          1024 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 2: World properties and layer assembly:

- Note the dummy object setting shader registers prior to tile rendering
- Note the tile set property tags. Only tile set #0 is used for cursor display. Tile sets are the only Principia component that are referred to by an ordinal number, and not via a named ID.

```
##define <SYS_VARIABLE>   =  (   V_LODCTL ,   VARTYPE_INT ,    LOD_NEAR )

##define (WRLD_EX01) as <WORLD_RT3A>
   #tag Sector_Mode   =  (          WPART_RECT2D )
   #tag Sector_Dim    =  (          12 ,  12 , 1 ) ' Should be tile multiple
   #tag Sector_Band   =  (                 0.01 ) ' Sector safety guardband
   #tag Tile_SetCodes =  (  0 ,             YES ) ' Use tile set #0 for cursor
   #tag Tile_SetCodes =  (  1 ,              NO ) ' Nus tile set #1 for cursor
   #tag Tile_SetCodes =  (  2 ,              NO ) ' Nus tile set #1 for cursor


   #tag Layer_Decl    =  (       L_PROPS ,          "PROPS" ,                - , - ,
   #tag Layer_Decl    =  (   L_TILES_INI ,      "TILES_INIT" ,                - , - ,
   #tag Layer_Decl    =  (   L_TILES_G1N ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_G2N ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_W1N ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (       L_GRIDN ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_G1F ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_G2F ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_W1F ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (       L_GRIDF ,              NONE ,   NONE ,     NONE )

   #tag Layer_Decl    =  (       L_WCURS ,         "WCURSOR" ,                - , - ,    4 )
   #tag Layer_Decl    =  (     L_OBJ_INI ,        "OBJ_INIT" ,        M_STDNULL , - ,    4 )
   #tag Layer_Decl    =  (  L_OBJ_SOLIDS ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (  L_OBJ_TREES1 ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (  L_OBJ_TREES2 ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (     L_OBJ_END ,        "OBJ_TERM" ,        M_STDNULL , - ,    4 )

   . . .
   #tag Layer_Item    =  (       L_PROPS ,      O_AXISYS )
   #tag Layer_Item    =  (       L_PROPS ,     O_STDNULL )
   #tag Layer_Item    =  (   L_TILES_INI ,     O_TILEKNX )
```

```
##define (MSHD_STDXFORMS) as <MATERIAL_3A>
   #tag VsMat     =  (           0 ,       MAT_WT )
        'Model-to-world space transform in shader
        register 0
   #tag VsMat     =  (           8 ,      MAT_WVT )
        'Model-to-camera space transform in shader
        register 8

##define (KX_TILES_EX01) as <KINEX_S>
   #tag Script    =  "Decl(M);"
   #tag Material  =  ( MSHD_STDXFORMS )

##define (O_TILEKNX) as <GENOBJECT_3A>
   #tag Instance  =  (      INST_REF )
   #tag Construct =  ( KX_TILES_EX01 ,   R_NONRDR ,
        NONE )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 2: World LoD control conditions used to switch between near and far views.

```
. . .

   #tag Layer_LoD     =  (   L_TILES_G1N ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (   L_TILES_G2N ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (   L_TILES_W1N ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (       L_GRIDN ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (       L_WCURS ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (  L_OBJ_SOLIDS ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (  L_OBJ_TREES1 ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (  L_OBJ_TREES2 ,      V_LODCTL ,      LOD_NEAR ,     TEST_AND )
   #tag Layer_LoD     =  (   L_TILES_G1F ,      V_LODCTL ,      LOD_FAR  ,     TEST_AND )
   #tag Layer_LoD     =  (   L_TILES_G2F ,      V_LODCTL ,      LOD_FAR  ,     TEST_AND )
   #tag Layer_LoD     =  (   L_TILES_W1F ,      V_LODCTL ,      LOD_FAR  ,     TEST_AND )
   #tag Layer_LoD     =  (       L_GRIDF ,      V_LODCTL ,      LOD_FAR  ,     TEST_AND )

. . .
```

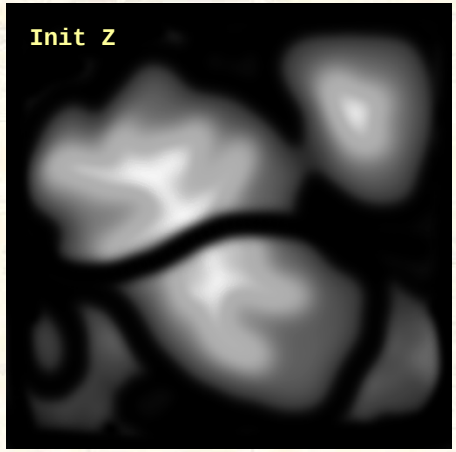
Important usage note:

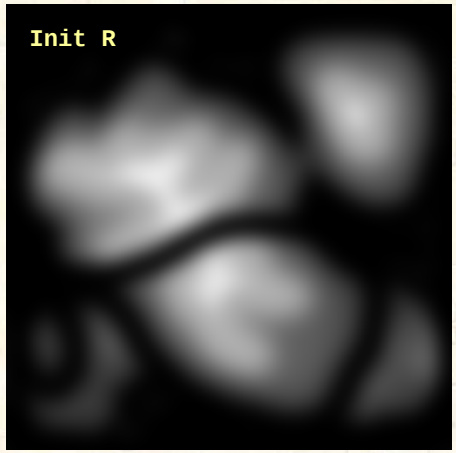
- **Because most world assembly procedures (including *ASM1L here) pre-clear all target tile layers, do not seed these layers with objects in the world definition.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 3: Elevation field generation. Most RT3A family components are elevation-field driven.**
  - An initial, simple image kernel, that hints at the overall terrain shape is fractally convolved using a HField2A procedure to auto generate the world elevation field.
  - The fractal roughness distribution is controlled by a similar map. This makes mountains sharper and flat areas smoother.
  - The fractal convolution hides the height steps that would be visible an 8-bit grayscale elevation map was directly used for fZ(UV).





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 3: Elevation generation. Much procedural activity is easily and simply controlled by maps!**

```
##define (S_ZINI_EX01) as <SURFACE>
   #tag File      =  "Files_Media/Map_Elevation_Input_Ex01.bmp"
   #tag Definer   =  (       DEF_S2D )
   #tag LoadNow   =  (             1 )

##define (S_RINI_EX01) as <SURFACE>
   #tag File      =  "Files_Media/Map_Roughness_Input_Ex01.bmp"
   #tag Definer   =  (       DEF_S2D )
   #tag LoadNow   =  (             1 )

##define (KF_ELEV_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(u,v); u=frac(u); v=frac(v); z=LMap2(u,v); Ret(z); "
   #tag ZMap      =  (   S_ZINI_EX01 )
   #tag ZMin      =  (         0.000 )
   #tag ZMax      =  (         1.000 )

##define (KF_RUGH_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(u,v); u=frac(u); v=frac(v); z=LMap2(u,v); Ret(z); "
   #tag ZMap      =  (   S_RINI_EX01 )
   #tag ZMin      =  (        -0.500 )
   #tag ZMax      =  (         0.400 )
```

- **Note the importance of wrapping the image map sampling arguments before fractal sampling!**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 3: Elevation generation. Note the height scale of the final terrain, used for feature ID.**

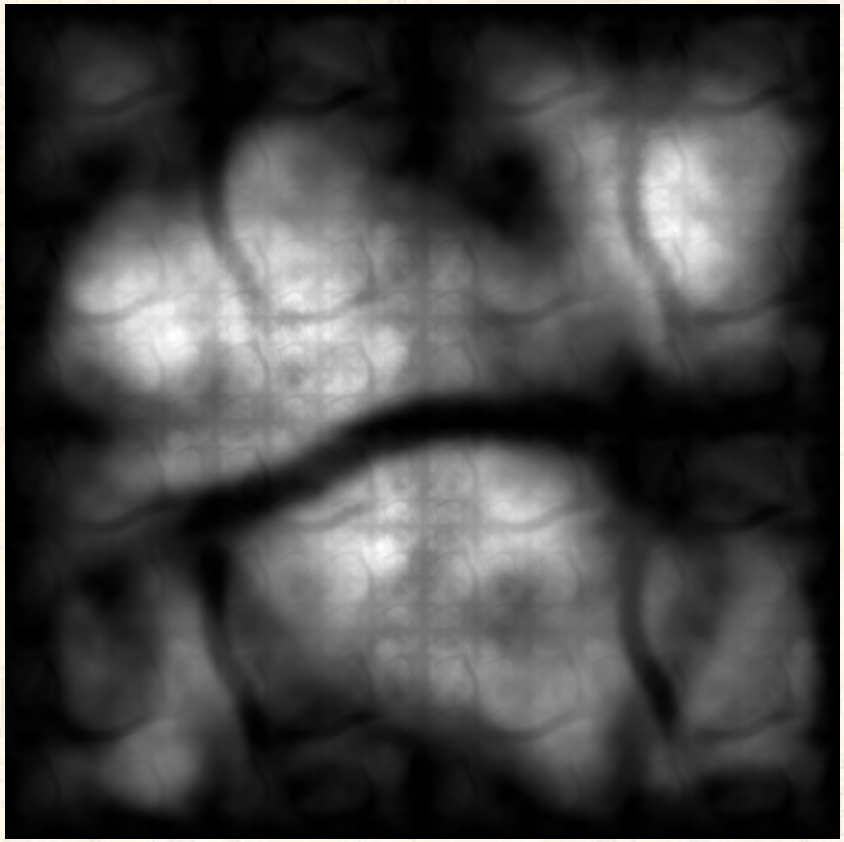
```
##define (HF_ELEV_EX01) as <PROCDAT_HFIELD_2A>
   #tag ProceduralModel        =  (         PF_FBM )
   #tag MappingMethod          =  (        PM_ROTS )
   #tag CoreKernel             =  (   KF_ELEV_EX01 )
   #tag XTransF                =  (              - )
   #tag YTransF                =  (              - )
   #tag ZTransF                =  (              - )
   #tag KernelR                =  (   KF_RUGH_EX01 )
   #tag KernelL                =  (              - )
   #tag KernelG                =  (              - )
   #tag Roughness              =  (           0.70 )
   #tag Offsets                =  (   0.00 ,  0.00 )
   #tag Gain                   =  (           2.00 )
   #tag Lacunarity             =  (           2.00 )
   #tag Octaves                =  (              7 )
   #tag DoScale                =  (            YES )
   #tag NbSamples              =  (             64 )
   #tag UserMin                =  (            0.0 )
   #tag UserMax                =  (            1.0 )
   #tag BlendFunction          =  (              - )
   #tag HFieldA                =  (              - )
   #tag HFieldB                =  (              - )
   #tag HFieldC                =  (              - )
   #tag HFieldD                =  (              - )

##define (F_ELEV_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); h=hf(x,y); h=Spl3(h); Ret(h); "
   #tag HField    =  (  HF_ELEV_EX01 )
   #tag Data      =  (         -1.000 ,      -1.000 )
   #tag Data      =  (          0.000 ,      -0.400 )
   #tag Data      =  (          0.100 ,      -0.300 )
   #tag Data      =  (          0.150 ,       0.000 )
   #tag Data      =  (          0.300 ,       0.050 )
   #tag Data      =  (          0.400 ,       0.100 )
   #tag Data      =  (          0.600 ,       0.300 )
   #tag Data      =  (          0.750 ,       0.500 )
   #tag Data      =  (          0.800 ,       1.000 )
   #tag Data      =  (          1.000 ,       1.500 )
   #tag Data      =  (          1.200 ,       2.000 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 3: Elevation field generation. This is the resulting elevation map in 8-bit grayscale.**
  - In production, the elevation map would be generated in 16-bit grayscale
  - Then retouched with image editing software to refine or add special features
  - And re-loaded in 16-bit resolution to avoid step artifacts.



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT generation. We need several DCT for rendering, terrain typing and tile-layer mapping:**
  - DCT for substrate layer, describing the distribution of three different substrate types. Substrate tiles will be placed everywhere except where fully covered (tiles under water are not considered covered due to alpha).
  - DCT for the cover layer, describing the distribution of cover, with a separate alpha channel to fine-control how the cover blends with the underlying substrate.
  - DCT for the water layer, describing where geometry is needed. This DCT will be used to generate non-rect tiles with vertices only where water is needed.
- **The DCTs are independent of LoD, but interesting FX can be achieved if this was not the case.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT generation, substrate. The fastest way to do it is using channel procedures such as TexGen1DC. Here are the channel functions:**

```
##define (F_DCT_LAYER_GRNDSOIL_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"
   #tag Operation =  " o=Tab2(z,s); Ret(o);"
   #tag Cell2D    =  ( -9.99, 0.04 ,  0.00, 8.00,   0.0 ) ' Soil low-z, low-s
   #tag Cell2D    =  (  0.04, 0.50 ,  0.00, 8.00,   1.0 ) ' Soil mid-z, low-s
   #tag Cell2D    =  (  0.50, 9.99 ,  0.00, 8.00,   0.0 ) ' Soil hig-z, low-s
   #tag Cell2D    =  ( -9.99, 0.04 ,  8.00, 22.0,   0.0 ) ' Soil low-z, mid-s
   #tag Cell2D    =  (  0.04, 0.50 ,  8.00, 22.0,   1.0 ) ' Soil mid-z, mid-s
   #tag Cell2D    =  (  0.50, 9.99 ,  8.00, 22.0,   0.0 ) ' Soil hig-z, mid-s
   #tag Cell2D    =  ( -9.99, 0.04 ,  22.0, 99.9,   0.0 ) ' Soil low-z, hig-s
   #tag Cell2D    =  (  0.04, 0.50 ,  22.0, 99.9,   0.0 ) ' Soil mid-z, hig-s
   #tag Cell2D    =  (  0.50, 9.99 ,  22.0, 99.9,   0.0 ) ' Soil hig-z, hig-s

##define (F_DCT_LAYER_GRNDROCK_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"
   #tag Operation =  " o=Tab2(z,s); Ret(o);"
   #tag Cell2D    =  ( -9.99, 0.04 ,  0.00, 8.00,   0.5 ) ' Rock low-z, low-s
   #tag Cell2D    =  (  0.04, 0.50 ,  0.00, 8.00,   0.5 ) ' Rock mid-z, low-s
   #tag Cell2D    =  (  0.50, 9.99 ,  0.00, 8.00,   0.5 ) ' Rock hig-z, low-s
   #tag Cell2D    =  ( -9.99, 0.04 ,  7.50, 22.0,   0.5 ) ' Rock low-z, mid-s
   #tag Cell2D    =  (  0.04, 0.50 ,  5.00, 22.0,   0.5 ) ' Rock mid-z, mid-s
   #tag Cell2D    =  (  0.50, 9.99 ,  8.00, 22.0,   0.5 ) ' Rock hig-z, mid-s
   #tag Cell2D    =  ( -9.99, 0.04 ,  45.0, 99.9,   0.5 ) ' Rock low-z, hig-s
   #tag Cell2D    =  (  0.04, 0.50 ,  22.0, 99.9,   0.5 ) ' Rock mid-z, hig-s
   #tag Cell2D    =  (  0.50, 9.99 ,  22.0, 99.9,   0.5 ) ' Rock hig-z, hig-s

##define (F_DCT_LAYER_GRNDCLIF_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"
   #tag Operation =  " o=Tab2(z,s); Ret(o);"
   #tag Cell2D    =  ( -9.99, 0.04 ,  0.00, 8.00,   0.0 ) ' Cliff low-z, low-s
   #tag Cell2D    =  (  0.04, 0.50 ,  0.00, 8.00,   0.0 ) ' Cliff mid-z, low-s
   #tag Cell2D    =  (  0.50, 9.99 ,  0.00, 8.00,   0.7 ) ' Cliff hig-z, low-s
   #tag Cell2D    =  ( -9.99, 0.04 ,  8.00, 22.0,   0.0 ) ' Cliff low-z, mid-s
   #tag Cell2D    =  (  0.04, 0.50 ,  8.00, 22.0,   0.0 ) ' Cliff mid-z, mid-s
   #tag Cell2D    =  (  0.50, 9.99 ,  8.00, 22.0,   0.7 ) ' Cliff hig-z, mid-s
   #tag Cell2D    =  ( -9.99, 0.04 ,  22.0, 99.9,   0.0 ) ' Cliff low-z, hig-s
   #tag Cell2D    =  (  0.04, 0.50 ,  22.0, 99.9,   0.0 ) ' Cliff mid-z, hig-s
   #tag Cell2D    =  (  0.50, 9.99 ,  22.0, 99.9,   0.7 ) ' Cliff hig-z, hig-s
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT generation, substrate. Note that it is best to link DCT size to world grid resolution, and activate unit domination in the DCT procedure.**

```
##define (PROC_DCT_LAYER_GRND_EX01) as <PROCTEX_GENDCT_1A>
   #tag Method    =  ( PROC_DCGEN_HFRGBA )
   #tag Norm3     =  (                NO )
   #tag Norm4     =  (                NO )
   #tag Domine3   =  (               YES )
   #tag Domine4   =  (                NO )
   #tag DomineL   =  (                NO )
   #tag SecondDCT =  (                 - )
   #tag TexA      =  (                 - )
   #tag CodA      =  (        0x00000000 )
   #tag FieldZ    =  (       F_ELEV_EX01 )
   #tag FieldS    =  (       F_ELEV_EX01 )
   #tag FieldL    =  (       F_ELEV_EX01 )
   #tag FieldC    =  (                 - )
   #tag FieldN    =  (                 - )
   #tag Scale     =  (    0.05 ,    0.05 ) ' Discretization related
   #tag FL_Level  =  (                0.0 )
   #tag FC_Level  =  (                0.0 )
   #tag Channel1R =  ( F_DCT_LAYER_GRNDSOIL_EX01 )
   #tag Channel1G =  ( F_DCT_LAYER_GRNDROCK_EX01 )
   #tag Channel1B =  ( F_DCT_LAYER_GRNDCLIF_EX01 )
   #tag Channel1A =  (                     NONE )

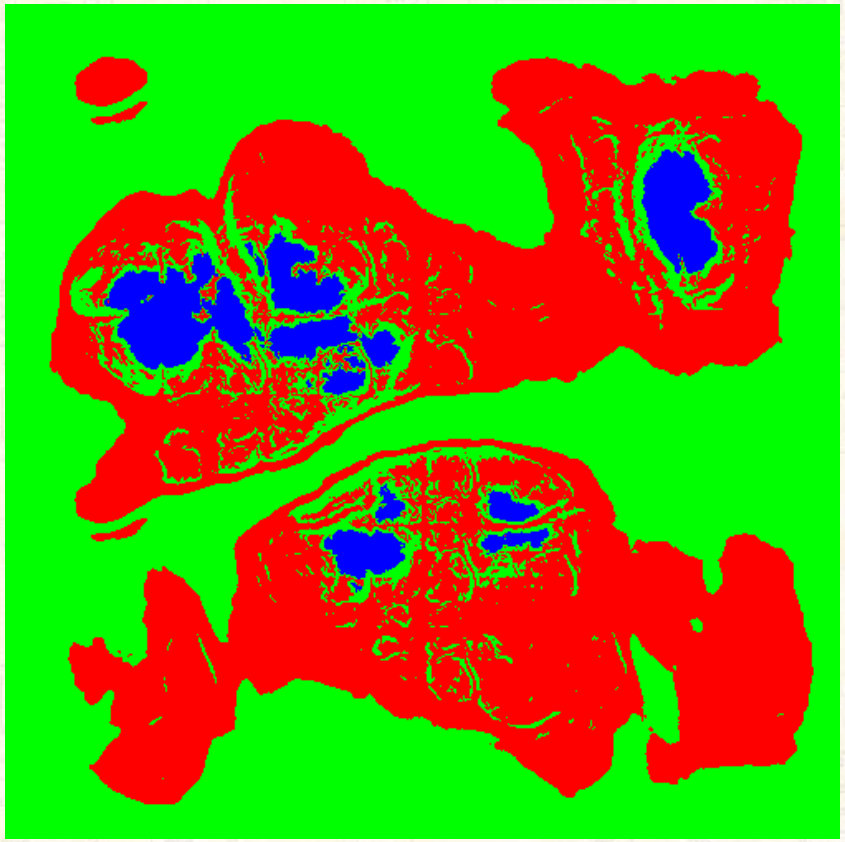
##define (S_DCT_LAYER_GRND_EX01) as <SURFACE>
   #tag Procedure =  ( PROC_DCT_LAYER_GRND_EX01 )
   #tag Definer   =  (          DEF_TEX )
   #tag Width     =  (              505 ) ' Discretization related
   #tag Heigt     =  (              505 ) ' Discretization related
   #tag LoadNow   =  (                1 )

##save (S_DCT_LAYER_GRND_EX01) in ("Files_Media/Map_DCT_LayerGround_Ex01_C.bmp") alpha in ("Files_Media/Map_DCT_LayerGround_Ex01_A.bmp")
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT generation, substrate. This is the resulting DCT, used by procedure and shaders.**
  - In this DCT, G=lime rock (by water), R=soil, B=dark serpentine rock.
  - The simple channel functions used un this TexGen1DC lead to unnatural texture distribution when carefully examined.
  - We will refine this in later examples.



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT generation, water. This texture is not used for rendering, but helps as procedural key for tile economy measures and terrain typing.**

```
##define (F_DCT_LAYER_WATRRED_EX01) as <FUNCTION_S> ' Encodes water. No cursor.
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"      ' Elevation field related
   #tag Operation =  " o=Gt(z,0.00,0.0,1.0); "       ' Elevation field related
   #tag Operation =  " Ret(o); "

##define (F_DCT_LAYER_WATRGRN_EX01) as <FUNCTION_S> ' Encodes nothing.
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"      ' Elevation field related
   #tag Operation =  " o=Set(0.0); Ret(o); "

##define (F_DCT_LAYER_WATRBLU_EX01) as <FUNCTION_S> ' Encodes nothing.
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"      ' Elevation field related
   #tag Operation =  " o=Set(0.0); Ret(o); "

##define (F_DCT_LAYER_WATRALF_EX01) as <FUNCTION_S> ' Encodes mountain. No cursor.
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"      ' Elevation field related
   #tag Operation =  " o=Gt(z,0.60,1.0,0.0); "       ' Elevation field related
   #tag Operation =  " Ret(o); "

##define (PROC_DCT_LAYER_WATR_EX01) as <PROCTEX_GENDCT_1A>
   #tag Method    =  ( PROC_DCGEN_HFRGBA )
   #tag Norm3     =  (                NO )
   #tag Norm4     =  (                NO )
   #tag Domine3   =  (                NO )
   #tag Domine4   =  (               YES )
   #tag DomineL   =  (                NO )
   #tag TexA      =  (                 - )
   #tag CodA      =  (        0x00000000 )
   #tag FieldZ    =  (       F_ELEV_EX01 )
   #tag FieldS    =  (                 - )
   #tag FieldL    =  (                 - )
   #tag FieldC    =  (                 - )
   #tag FieldN    =  (                 - )
   #tag Scale     =  (    0.05 ,    0.05 ) ' Discretization related
   #tag FL_Level  =  (               0.0 )
   #tag FC_Level  =  (               0.0 )
   #tag Channel1R =  ( F_DCT_LAYER_WATRRED_EX01 ) ' Used for water GridType marking
   #tag Channel1A =  ( F_DCT_LAYER_WATRALF_EX01 ) ' Used for moubtain GridType marking
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT generation, water. These are the resulting DCT maps for water and mountains…**





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT, cover. This texture is manually painted to place some roads, grass and snow.**

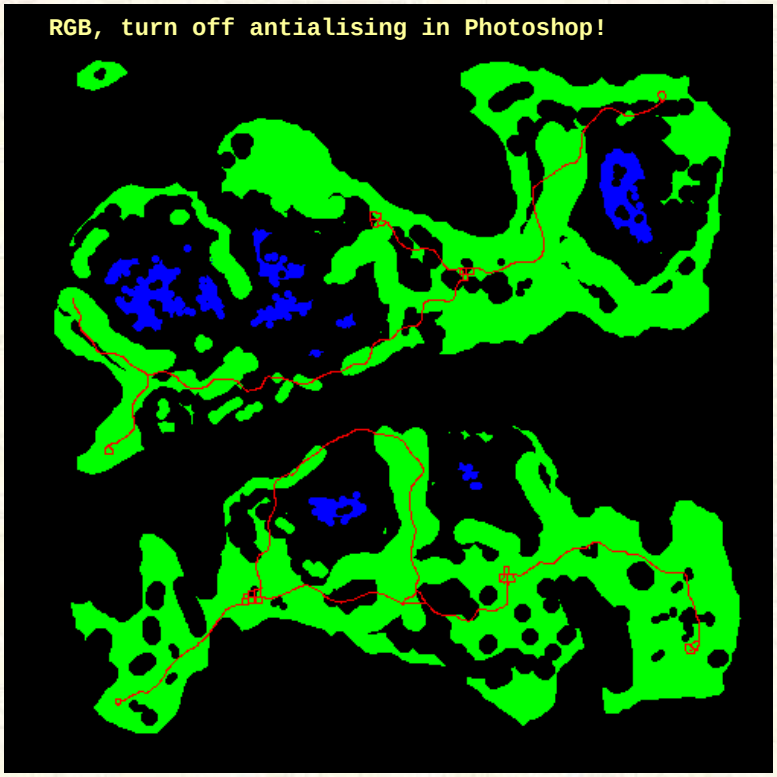
```
##define (S_DCT_LAYER_COVR_EX01) as <SURFACE>
   #tag File      =  "Files_Media/Map_DCT_LayerCover_Ex01_C.bmp"
   #tag Alpha     =  "Files_Media/Map_DCT_LayerCover_Ex01_A(MinusPixel).bmp"
   #tag Definer   =  (       DEF_TEX )
   #tag LoadNow   =  (             1 )
```

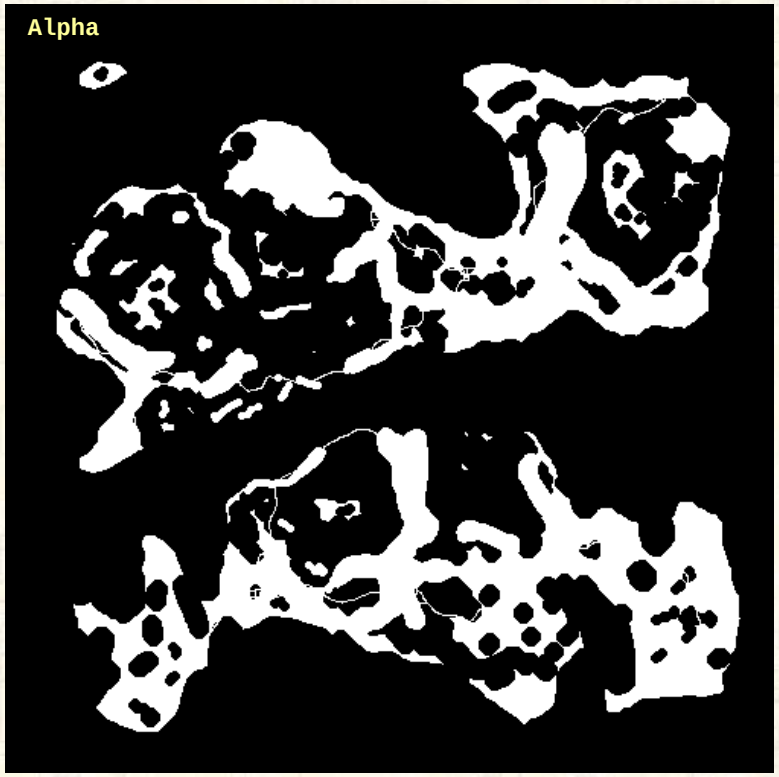
- **The cover is blended with the substrate in the frame buffer. This requires careful alpha design:**
  - DCT RGB data can have gaps
  - If the alpha channel maps exactly to RGB, an INVSRC frame buffer blend will have dark edges due to black interpolated from DCT data (visible on roads in Ex01).
  - One solution is to paint the alpha channel to be -1 pixel smaller than RGB contents.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 4: DCT for cover layer. Painted in Photoshop using the elevation map as paint template. DCTs design is critical for realistic world appearance.**





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 5: Layer materials design considerations:

- To minimize the performance impact of state changes, sort layer objects by material. Best practice is to do major GPU state changes at layer initialization.
- To minimize CPU overhead, Place same-material objects on the same layer (and remove the material reference and place it in the layer definition itself).
- Avoid unintentionally inherited GPU states such as filtering that may be left over from previous layers. Strategically insert “reset” material layers.
- Many SFX will require multiple pass rendering. Use linked layers for this purpose and think of how to sequence operations for linked, multi-pass layers.
- Think carefully of shader ops, shader constant states, and DCT texture states, and frame buffer propagation thru layers.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Layer material, substrate. Note carefully all the settings, for each one has a visible role!**

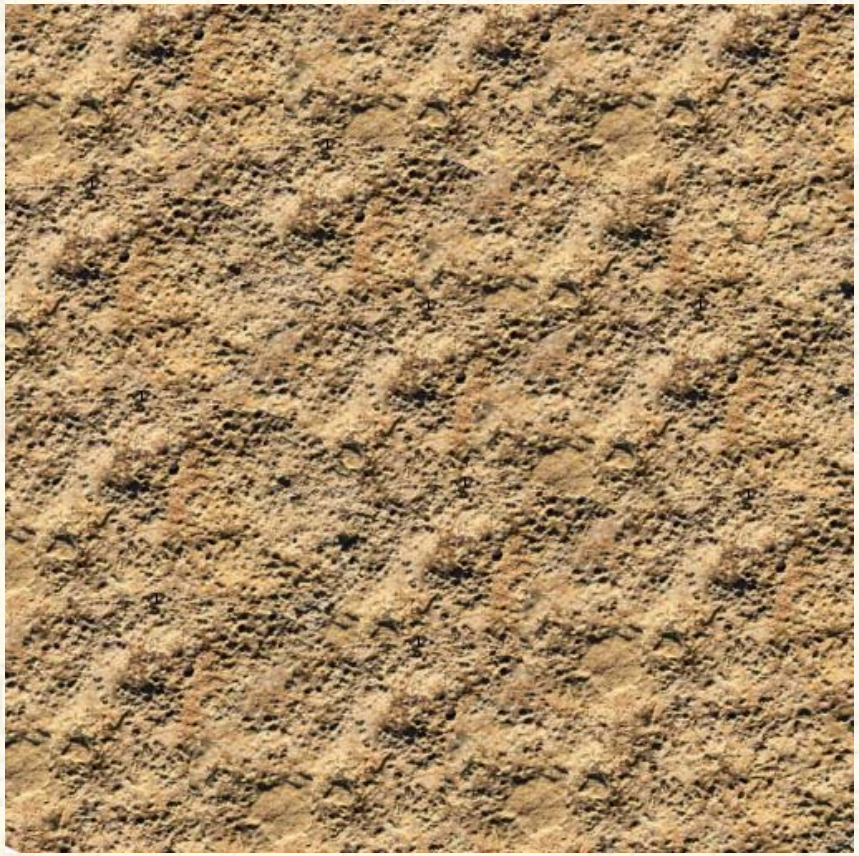
```
##define (M_LAYER_GRND_EX01) as <MATERIAL_3A>
   #tag RS       =  (                RS_ZENABLE ,              1 )
   #tag RS       =  (           RS_ZWRITEENABLE ,              1 )
   #tag RS       =  (              RS_DEPTHBIAS ,        0.00000 )
   #tag RS       =  (              RS_FILLMODE ,       RA_SOLID )
   #tag RS       =  (              RS_CULLMODE ,     RA_CULLCCW )
   #tag RS       =  (            RS_VERTEXBLEND ,     BL_DISABLE )
   #tag TS       =  (                TS_COLOROP ,     TO_DISABLE ,   0 )
   #tag TS       =  (                TS_ALPHAOP ,     TO_DISABLE ,   0 )
   #tag TS       =  (                TS_COLOROP ,     TO_DISABLE ,   1 )
   #tag TS       =  (                TS_ALPHAOP ,     TO_DISABLE ,   1 )
   #tag TS       =  (                TS_COLOROP ,     TO_DISABLE ,   2 )
   #tag TS       =  (                TS_ALPHAOP ,     TO_DISABLE ,   2 )
   #tag TS       =  (                TS_COLOROP ,     TO_DISABLE ,   3 )
   #tag TS       =  (                TS_ALPHAOP ,     TO_DISABLE ,   3 )
   #tag TS       =  (                TS_COLOROP ,     TO_DISABLE ,   4 )
   #tag TS       =  (                TS_ALPHAOP ,     TO_DISABLE ,   4 )
   #tag RS       =  (       RS_ALPHATESTENABLE ,              0 )
   #tag RS       =  (              RS_ALPHAREF ,              0 )
   #tag RS       =  (              RS_ALPHAFUNC ,     CA_GREATER )
   #tag RS       =  (       RS_ALPHABLENDENABLE ,              0 )
   #tag RS       =  (              RS_SRCBLEND ,    FA_SRCALPHA )
   #tag RS       =  (              RS_DESTBLEND , FA_INVSRCALPHA )
   #tag SA       =  (              SS_MAGFILTER ,      SA_NONE   ,   0 ) ' No sampling of base texture
   . . .
   #tag SA       =  (              SS_MIPFILTER ,      SA_NONE   ,   2 ) ' No sampling of base texture
   #tag SA       =  (              SS_MAGFILTER ,      SA_LINEAR ,   3 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (              SS_MINFILTER ,      SA_LINEAR ,   3 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (              SS_MIPFILTER ,      SA_LINEAR ,   3 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (              SS_MAGFILTER ,      SA_LINEAR ,   4 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (              SS_MINFILTER ,      SA_LINEAR ,   4 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (              SS_MIPFILTER ,      SA_LINEAR ,   4 ) ' Linear sample DCT for smooth blend
   #tag VShader  =  (       VS_LAYER_GRND_EX01 )
   #tag PShader  =  (       PS_LAYER_GRND_EX01 )
   #tag Material =  (       MSHD_STDCAMLIT , - )
   #tag TX       =  (     SG_SOIL_DRYCRACKLITE ,  0 )  ' Terrain base texture 0, soil
   #tag TX       =  (        SG_ROCK_LIMESTONE ,  1 )  ' Terrain base texture 1, rock
   #tag TX       =  (      SG_ROCK_GRANITEDARK ,  2 )  ' Terrain base texture 2, clif
   #tag TX       =  (    S_DCT_LAYER_GRND_EX01 ,  3 )  ' Terrain control texture A
   #tag TX       =  (                     NONE ,  4 )  ' Terrain base texture 4
   #tag TX       =  (                     NONE ,  5 )  ' Terrain base texture 5
   #tag TX       =  (                     NONE ,  6 )  ' Terrain base texture 6
   #tag TX       =  (                     NONE ,  7 )  ' Terrain control texture B
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 5: Layer material, substrate. Example of tile texture used in the 3-channel DCT shader.

- Texture will be repeatedly tiled using the intrinsic tile UV system.
- Tile sizing must account for texture size and viewing distance.
- Texture should be periodic, or seams would appear.
- Texture must be even, or tiling would be visible.
- Consider lighting and shader design in texture selection.



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Layer material, substrate, vertex shader. It is one of the simplest world shader designs.**

```
struct VS_INPUT
{   // Inbound vertex buffer stream
    float3 vPosition        : POSITION ;
    float3 vNormal          : NORMAL   ;
    float2 vTexCoord0       : TEXCOORD0;
    float2 vTexCoord1       : TEXCOORD1;
};


struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ;
    float4 vDiffuse         : COLOR0   ;
    float4 vSpecular        : COLOR1   ;
    float2 vTexCoord0       : TEXCOORD0;
    float2 vTexCoord1       : TEXCOORD1;
    float2 vTexCoord2       : TEXCOORD2;
    float2 vTexCoord3       : TEXCOORD3;
    float2 vTexCoord4       : TEXCOORD4;
};
```

- **The shader expects full-stream position/normal tile data (can be greatly cut in production). It has simple per-vertex D+S illumination, and routes the base+DCT texcoords to the pixel shader.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Vertex shader, substrate. Note again that there is no room for per-pixel data flow.**

```
//--------------------------------------------------------------------------//
//  Shader principal operations                                             //
//--------------------------------------------------------------------------//

VS_OUTPUT Vertex_Shader_Ex01_LG1(const VS_INPUT In)
{   // Output declaration and initialization
    VS_OUTPUT Out     = (VS_OUTPUT) 0;

    // World-View-Projection transform
    Out.vPosition     = mul(float4(In.vPosition,1.0f), mWrld);
    Out.vPosition     = mul(Out.vPosition, mViewProj);
    WrldN             = mul(In.vNormal, (float3x3) mWrld);
    WrldN             = normalize(WrldN);

    // Ambient-Diffuse illumination
    LdotN             = dot(-vLDir, WrldN);
    LdotN             = max(LdotN, 0);
    Out.vDiffuse      = vLDiff*LdotN + vLAmbi;
    Out.vDiffuse      = min(Out.vDiffuse, float4(1,1,1,1));

    /* Specular illumination */
    AuxV              = mul(float4(In.vPosition,1), (float4x3) mWrldView);
    AuxV              = normalize(AuxV);
    AuxV              = normalize(vLDir+AuxV);
    Out.vSpecular     = saturate(dot(WrldN, -AuxV));
    Out.vSpecular     = pow(Out.vSpecular, vMPwr);
    Out.vSpecular    *= vLSpec;

    // Texture coordinates output for 3-texture blend
    Out.vTexCoord0.xy = In.vTexCoord0.xy ;
    Out.vTexCoord1.xy = In.vTexCoord0.xy ;
    Out.vTexCoord2.xy = In.vTexCoord0.xy ;
    Out.vTexCoord3.xy = In.vTexCoord1.xy ;

    // Output stream
    return Out;
}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Pixel shader, substrate, for three channel, non-alpha DCT blend. This is the max for PS1.3…**

```
ps.1.4
;------------------------------------------------------------------------;
;                                                                        ;
;   Terrain tres-texture blending using distribution control texture     ;
;   with per-vertex diffuse+specular illumination from vertex shader     ;
;                                                                        ;
;   c0  - auxiliary constant (0,H,1,1)                                   ;
;   t0  - terrain texture A                                              ;
;   t1  - terrain texture B                                              ;
;   t2  - terrain texture C                                              ;
;   t3  - distribution control texture for ABC*RGB blend                 ;
;                                                                        ;
;------------------------------------------------------------------------;
def    c0,  0.0,  0.5,  1.0,  1.0      ; Local constant

;------------------------------------------------------------------------;
;   Phase2: Independent texture read                                     ;
;------------------------------------------------------------------------;
texld  r0, t0                          ; Read terrain texture A
texld  r1, t1                          ; Read terrain texture B
texld  r2, t2                          ; Read terrain texture C
texld  r3, t3                          ; Read dct texture

;------------------------------------------------------------------------;
;   Phase2: Dependent read of light functon map                          ;
;------------------------------------------------------------------------;
mul    r0.rgb ,  r0   ,   r3.r         ; Blend texture A
mad    r0.rgb ,  r1   ,   r3.g ,   r0  ; Blend texture B
mad    r0.rgb ,  r2   ,   r3.b ,   r0  ; Blend texture C
mad    r0.rgb ,  r0   ,   v0   ,   v1  ; Per-vertex lighting
mov    r0.a   ,  c0.a                  ; No transparency

;------------------------------------------------------------------------;
;   End                                                                  ;
;------------------------------------------------------------------------;
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Basic world layer single-pass multi-texturing strategies:**
  - Three base textures (T3)
  - Four base textures (T4)
  - Six base textures (T6)
  - These strategies are predicated on per-vertex lighting
- **T3 strategy**
  - Single DCT, runs on all pixel shader versions
  - Rendered pixel = DCT_R (T1) + DCT_G (T2) + DCT_B (T3)
  - Four spare 8-bit alpha channels available (1 global, 3 per tile)
- **T4 strategy**
  - Single DCT, requires PS1.4 capable hardware
  - Rendered pixel = DCT_R (T1) + DCT_G (T2) + DCT_B (T3) + DCT_A (T4)
  - Thee tile spare 8-bit alpha channels available
- **T6 strategy**
  - Coupled DCT, requires PS2.0 capable hardware
  - Rendered pixel=DCT_{R1}(T1)+DCT_{G1}(T2)+DCT_{B1}(T3)+DCT_{R2}(T4)+DCT_{G2}(T5)+DCT_{B2}(T6)
  - Eight spare 8-bit alpha channels available (2 global, 6 per tile)

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Spare channel usage for terrain renders**
  - Specular properties (tile or DCT)
  - Intensity variation (DCT)
  - Color variation (DCT)
  - Alpha variation (tile or DCT)
  - Index perturbations
- **Per-pixel lighting for terrain renders**
  - Must ensure that hardware can handle it
  - Typical PS2.0 schema is T3 color+T3 bump map @ 1 DCT
  - Typical PS1.x schema is T1 color+T1 map @ ½ DCT
  - May require multiple passes to render a layer (esp. PS1.x). This is particularly suitable when using LoD variables to control render quality.
  - May require tangent space simplifying assumptions/reductions
- **The WorldRT3A components and procedural generators are not constrained to a particular lighting & texturing approach. Rather, they are intended to empower designers to implement their own lighting without coding.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 5: Layer material, cover. Note the activation of alpha blending and the use of depth offset!

```
##define (M_LAYER_COVR_EX01) as <MATERIAL_3A>
   #tag RS       =  (              RS_ZENABLE ,              1 )
   #tag RS       =  (         RS_ZWRITEENABLE ,              0 ) ' Do not update cover
   #tag RS       =  (            RS_DEPTHBIAS ,       -0.00001 ) ' Ensure bias
   #tag RS       =  (              RS_FILLMODE ,       RA_SOLID )
   #tag RS       =  (              RS_CULLMODE ,     RA_CULLCCW )
   #tag RS       =  (           RS_VERTEXBLEND ,     BL_DISABLE )
   #tag TS       =  (               TS_COLOROP ,     TO_DISABLE ,   0 )
   . . .
   #tag TS       =  (               TS_ALPHAOP ,     TO_DISABLE ,   4 )

   #tag RS       =  (      RS_ALPHATESTENABLE ,              1 ) ' Enable alpha
   #tag RS       =  (             RS_ALPHAREF ,              1 ) ' Enable alpha
   #tag RS       =  (            RS_ALPHAFUNC ,     CA_GREATER ) ' Enable alpha
   #tag RS       =  (     RS_ALPHABLENDENABLE ,              1 ) ' Enable alpha
   #tag RS       =  (             RS_SRCBLEND ,    FA_SRCALPHA ) ' Enable alpha
   #tag RS       =  (            RS_DESTBLEND , FA_INVSRCALPHA ) ' Enable alpha

   #tag SA       =  (             SS_MAGFILTER ,      SA_NONE   ,   0 ) ' No sampling of base texture
   #tag SA       =  (             SS_MINFILTER ,      SA_NONE   ,   0 ) ' No sampling of base texture
   #tag SA       =  (             SS_MIPFILTER ,      SA_NONE   ,   0 ) ' No sampling of base texture
   . . .
   #tag SA       =  (             SS_MAGFILTER ,      SA_NONE   ,   2 ) ' No sampling of base texture
   #tag SA       =  (             SS_MINFILTER ,      SA_NONE   ,   2 ) ' No sampling of base texture
   #tag SA       =  (             SS_MIPFILTER ,      SA_NONE   ,   2 ) ' No sampling of base texture
   #tag SA       =  (             SS_MAGFILTER ,      SA_LINEAR ,   3 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (             SS_MINFILTER ,      SA_LINEAR ,   3 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (             SS_MIPFILTER ,      SA_LINEAR ,   3 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (             SS_MAGFILTER ,      SA_LINEAR ,   4 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (             SS_MINFILTER ,      SA_LINEAR ,   4 ) ' Linear sample DCT for smooth blend
   #tag SA       =  (             SS_MIPFILTER ,      SA_LINEAR ,   4 ) ' Linear sample DCT for smooth blend
   #tag VShader  =  (       VS_LAYER_COVR_EX01 )
   #tag PShader  =  (       PS_LAYER_COVR_EX01 )
   #tag Material =  (       MSHD_STDCAMLIT , - )
   #tag TX       =  (          SG_ROAD_YELBRK ,  0 )  ' Terrain base texture 0, road
   #tag TX       =  (        SG_GRAS_DEEPGREEN ,  1 )  ' Terrain base texture 1, grass
   #tag TX       =  (          SG_SNOW_PACKED ,  2 )  ' Terrain base texture 2, snow
   #tag TX       =  (    S_DCT_LAYER_COVR_EX01 ,  3 )  ' Terrain control texture A
   #tag TX       =  (                     NONE ,  4 )  ' Terrain base texture 4
   #tag TX       =  (                     NONE ,  5 )  ' Terrain base texture 5
   #tag TX       =  (                     NONE ,  6 )  ' Terrain base texture 6
   #tag TX       =  (                     NONE ,  7 )  ' Terrain control texture B
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Pixel shader, cover. The alpha channel of the DCT is now used for the pixel alpha component to transpare uncovered areas.**

```
ps.1.4
;------------------------------------------------------------------------;
;                                                                        ;
;   Terrain tres-texture blending using distribution control texture     ;
;   with per-vertex diffuse+specular illumination from vertex shader     ;
;                                                                        ;
;   c0  - auxiliary constant (0,H,1,1)                                   ;
;   t0  - terrain texture A                                              ;
;   t1  - terrain texture B                                              ;
;   t2  - terrain texture C                                              ;
;   t3  - distribution control texture for ABC*RGB blend                 ;
;                                                                        ;
;------------------------------------------------------------------------;
def    c0,  0.0,  0.5,  1.0,  1.0      ; Local constant

;------------------------------------------------------------------------;
;   Phase2: Independent texture read                                     ;
;------------------------------------------------------------------------;
texld  r0, t0                          ; Read terrain texture A
texld  r1, t1                          ; Read terrain texture B
texld  r2, t2                          ; Read terrain texture C
texld  r3, t3                          ; Read dct texture

;------------------------------------------------------------------------;
;   Phase2: Dependent read of light functon map                          ;
;------------------------------------------------------------------------;
mul    r0.rgb ,  r0   ,   r3.r         ; Blend texture A
mad    r0.rgb ,  r1   ,   r3.g ,   r0  ; Blend texture B
mad    r0.rgb ,  r2   ,   r3.b ,   r0  ; Blend texture C
mad    r0.rgb ,  r0   ,   v0   ,   v1  ; Per-vertex lighting
mov    r0.a   ,  r3.a                  ; Cover transparency from DCT

;------------------------------------------------------------------------;
;   End                                                                  ;
;------------------------------------------------------------------------;
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Layer material, water. At near LoD, the water is rendered using the simplest animated normal and specular map shaders of Chap.1.04.**

```
##define (M_LAYER_WATR_EX01) as <MATERIAL_3A>
   #tag RS        =  (            RS_ZWRITEENABLE ,             1 )
   #tag RS        =  (               RS_DEPTHBIAS ,       0.00000 )
   #tag RS        =  (          RS_SPECULARENABLE ,             0 )
   #tag RS        =  (                RS_FILLMODE ,      RA_SOLID )
   #tag RS        =  (                RS_CULLMODE ,    RA_CULLCCW )
   #tag RS        =  (        RS_ALPHATESTENABLE ,             1 )
   #tag RS        =  (       RS_ALPHABLENDENABLE ,             1 )
   #tag RS        =  (                RS_ALPHAFUNC ,    CA_GREATER )
   #tag RS        =  (                 RS_SRCBLEND ,   FA_SRCALPHA )
   #tag RS        =  (                RS_DESTBLEND ,  FA_INVSRCALPHA )
   #tag SA        =  (                SS_MAGFILTER ,      SA_LINEAR ,   0 )
   #tag SA        =  (                SS_MINFILTER ,      SA_LINEAR ,   0 )
   #tag SA        =  (                SS_MIPFILTER ,      SA_LINEAR ,   0 )
   . . .
   #tag SA        =  (                SS_MAGFILTER ,      SA_LINEAR ,   4 )
   #tag SA        =  (                SS_MINFILTER ,      SA_LINEAR ,   4 )
   #tag SA        =  (                SS_MIPFILTER ,      SA_LINEAR ,   4 )
   #tag SA        =  (                 SS_ADDRESSU ,        SA_WRAP ,   0 )
   #tag SA        =  (                 SS_ADDRESSV ,        SA_WRAP ,   0 )
   #tag SA        =  (                 SS_ADDRESSU ,        SA_WRAP ,   1 )
   #tag SA        =  (                 SS_ADDRESSV ,        SA_WRAP ,   1 )
   #tag SA        =  (                 SS_ADDRESSU ,        SA_WRAP ,   2 )
   #tag SA        =  (                 SS_ADDRESSV ,        SA_WRAP ,   2 )
   #tag SA        =  (                 SS_ADDRESSU ,       SA_CLAMP ,   3 )
   #tag SA        =  (                 SS_ADDRESSV ,       SA_CLAMP ,   3 )
   #tag SA        =  (                 SS_ADDRESSU ,       SA_CLAMP ,   4 )
   #tag SA        =  (                 SS_ADDRESSV ,       SA_CLAMP ,   4 )
   #tag TX        =  (          S_BMAPWATER_A , 0 )
   #tag TX        =  (          S_BMAPWATER_A , 1 )
   #tag TX        =  (        S_ENVMAPWATER_A , 2 )
   #tag TX        =  (       S_FRESMAPWATER_A , 3 )
   #tag TX        =  (       S_SPECMAPWATER_A , 4 )
   #tag VShader   =  (         VS_LAYER_WATR_EX01 )
   #tag PShader   =  (         PS_LAYER_WATR_EX01 )
   #tag Material  =  (         MSHD_STDCAMLIT , - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Vertex shader, water. Water transparency and wave constants are hardwired in the shader. TC channels carry local illumination data.**

```
//*********************************************************************//
//  Shader configuration constants: remember z is negative under water     //
//*********************************************************************//
// Base and depth bias RGBA water properties
static float4  ClrZbase  = float4(  0.0,  0.1,  0.1,  0.7);
static float4  ClrZbias  = float4(  0.0,  0.2, -0.3, -1.0);
// Wave speed, scale and motion coefficients
static float4  WavSScl   = float4(  0.0000025, 0.0000025, 8.0, 8.0);
static float4  WavMove   = float4( 1.1,   1.3, -1.2, -1.4);

//*********************************************************************//
//  Shader input and output structures                                 //
//*********************************************************************//
struct VS_INPUT
{   // Inbound vertex buffer stream
    float3 vPosition        : POSITION ;
    float3 vNormal          : NORMAL   ;
    float2 vTexCoord0       : TEXCOORD0;
    float2 vTexCoord1       : TEXCOORD1;
};

struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ;  // Clip space position
    float4 vDiffuse         : COLOR0   ;  // Diffuse color for water depth
    float2 vTexCoord0       : TEXCOORD0;  // Tex sampling coordinates
    float2 vTexCoord1       : TEXCOORD1;  // Tex sampling coordinates
    float3 wLite            : TEXCOORD2;  // Light vector, world space
    float3 wView            : TEXCOORD3;  // View vector, world space
    float3 wHalf            : TEXCOORD4;  // Half vector, world space
};

//*********************************************************************//
//  Shader principal operations                                        //
//*********************************************************************//
VS_OUTPUT Vertex_Shader_Ex01_LW1(const VS_INPUT In)
{   /* Output declaration and initialization */
    VS_OUTPUT Out     = (VS_OUTPUT) 0;
. . .
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Vertex shader, water. Water tiles have Z-data, but it is not used for geometry. Instead, it is used to key per-vertex depth RGBA output.**

```
/* Position output, clip space */
WrldP             = mul(float4(In.vPosition.xy , 0.0f , 1.0f), mWrld);
Out.vPosition     = mul(WrldP, mViewProj);

/* Tangent coordinate system vectors in world space */
WrldN             = float3(0.0, 0.0, 1.0);
WrldT             = float3(1.0, 0.0, 0.0);
WrldB             = float3(0.0, 1.0, 0.0);
/*  Lite, view and half vectors in world space */
xLite             = normalize(-vLDir);
xView             = vEPos - WrldP.xyz;
xView             = normalize(xView);
xHalf             = 0.5*(xLite+xView);
xHalf             = normalize(xHalf);
/* Lite, view and half vectors in tangent space */
mW2T              = float3x3(WrldT, WrldB, WrldN);
mW2T              = transpose(mW2T);
xLite             = normalize(mul(xLite, mW2T ));
xView             = normalize(mul(xView, mW2T ));
xHalf             = normalize(mul(xHalf, mW2T ));

/* View, light and half vectors output in [-1,1] range */
Out.wLite         = xLite;
Out.wView         = xView;
Out.wHalf         = xHalf;

/* Diffuse color and transparency based on water depth */
Out.vDiffuse      = ClrZbase + ClrZbias*In.vPosition.zzzz;
Out.vDiffuse.a    *= Out.vDiffuse.a;

/* Wave adjustment vector */
xWave             = WavSScl.xxyy * vTime;
xWave             = WavMove * xWave;

/* Texture sampling coordinates for the two moving wave bumpmaps */
Out.vTexCoord0.x  = xWave.x + WavSScl.z*In.vTexCoord1.x;
Out.vTexCoord0.y  = xWave.y + WavSScl.z*In.vTexCoord1.y;
Out.vTexCoord1.x  = xWave.z + WavSScl.w*In.vTexCoord1.x;
Out.vTexCoord1.y  = xWave.w + WavSScl.w*In.vTexCoord1.y;
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 5: Grid layer material. Note Z-render states.**

```
##define (M_LAYER_GRID_EX01) as <MATERIAL_3A>
   #tag RS       =  (            RS_ZWRITEENABLE ,              0 )
   #tag RS       =  (               RS_DEPTHBIAS ,        -0.00002 )
   #tag RS       =  (                RS_FILLMODE ,    RA_WIREFRAME )
   #tag RS       =  (        RS_ALPHATESTENABLE ,              0 )
   #tag RS       =  (                RS_ALPHAREF ,              0 )
   #tag RS       =  (              RS_ALPHAFUNC ,      CA_GREATER )
   #tag RS       =  (       RS_ALPHABLENDENABLE ,              0 )
   #tag RS       =  (                RS_SRCBLEND ,     FA_SRCALPHA )
   #tag RS       =  (              RS_DESTBLEND ,  FA_INVSRCALPHA )
   #tag TX       =  (                 NONE ,  0 )  ' Terrain base texture 1
   #tag TX       =  (                 NONE ,  1 )  ' Terrain base texture 2
   #tag TX       =  (                 NONE ,  2 )  ' Terrain base texture 3
   #tag TX       =  (                 NONE ,  3 )  ' Terrain control texture A
   #tag TX       =  (                 NONE ,  4 )  ' Terrain base texture 4
   #tag TX       =  (                 NONE ,  5 )  ' Terrain base texture 5
   #tag TX       =  (                 NONE ,  6 )  ' Terrain base texture 6
   #tag TX       =  (                 NONE ,  7 )  ' Terrain control texture B
   #tag VShader  =  (                      NONE )
   #tag PShader  =  (                      NONE )
```

- **Very often we will want to render another pass of the same geometry with different material. The grid layers are a perfect example of how to do it.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

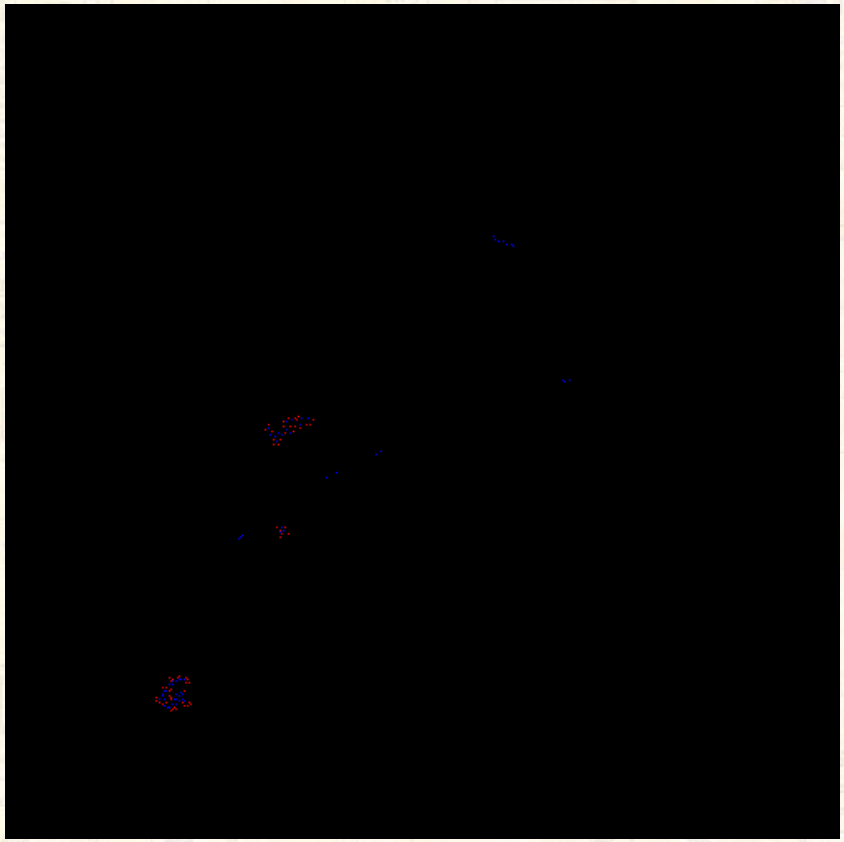
D108B:Ex01 – WorldRT3A

- **Step 5: Pixel shader, water.**
  - Refer to the caustics demo, Chapter 1.04.F for implementation details and for texture map design.
  - Some of the rendering shortcomings of the water in Ex01 can be fixed by just plugging a better shader.
- **However:**
  - True reflection effects will require multi-pass rendering to a render target used as reflection map.
  - True wave rendering with reverberations will require considerably more complex shaders with RT passes.
  - True transparency of immersed objects will require at minimum an immersed tile layer of deep water color.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

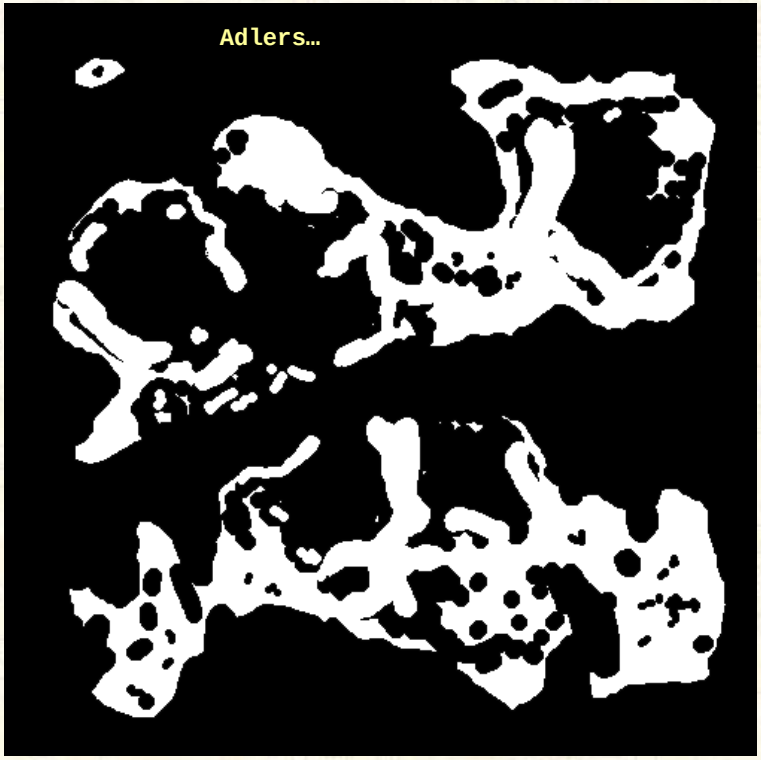
- **Step 6: Object scattering procedures. The detail of the procedures is covered in Chapter 3.07.**
  - The object variety is somewhat limited... mushrooms, rocks, and three tree types.
  - Mushrooms and rocks are placed using an exact distribution map, shown here.
  - Not all objects generated by the procedure will be placed in the world, since they are subject to code type checking.

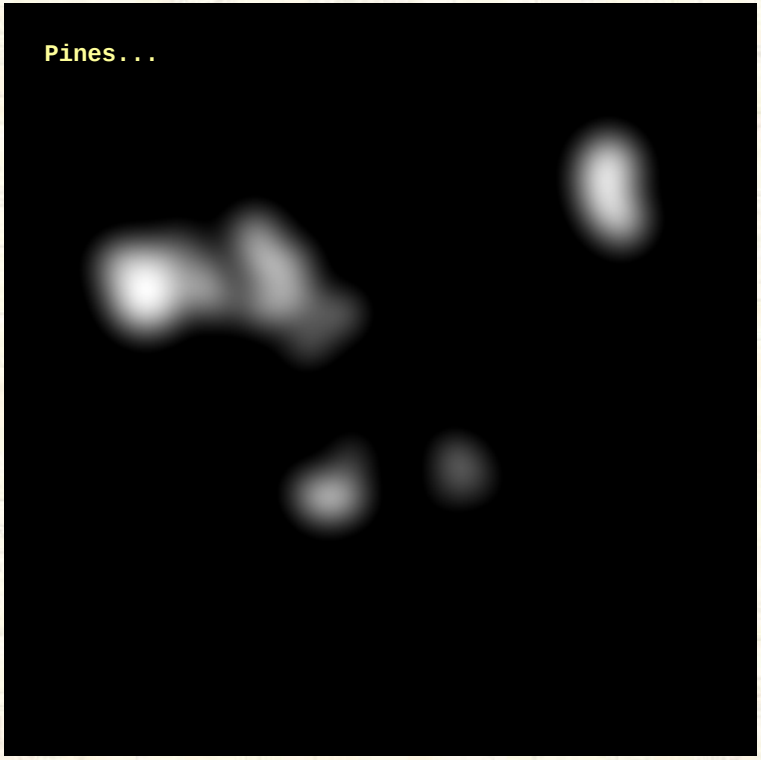


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 6: Object scattering procedures. Trees are placed using an ecosystem procedure from Chapter 3.03.F. Below are some of the suitability maps, keyed off the DCT and elevation maps.**





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 6: Object scattering procedure, mushrooms.

- The procedure uses the red channel in the map.
- Note that the ASM1RT world assembly procedure requires the referenced particle system procedures to be in physical mode.

```
##define (S_MAPLUMPS_EX01) as <SURFACE>
   #tag File      =  "Files_Media\Map_EnvSuitability_SolidLumps_Ex01.bmp"
   #tag Definer   =  (     DEF_TEX )

##define (PROCSYSGEN_SHROOMS_EX01) as <PROCOBJ_SCATTER_2FV>
   #tag MethodPos =  (    PROC_PSYS_SETMAP ) ' Deterministic placement
   #tag MethodVar =  (    PROC_PSYS_PDFMAP )
   #tag DimPos    =  (                   2 )
   #tag DimVar    =  (                   1 )
   #tag Physical  =  (                 YES )
   #tag Seed      =  (                   0 )
   #tag Range_X   =  (     -12.6 ,    12.6 ) ' Discretization related
   #tag Range_Y   =  (     -12.6 ,    12.6 ) ' Discretization related
   #tag GridSize  =  (     0.050 ,   0.050 ) ' Discretization related
   #tag GridJitX  =  "Arg(u,v); p=Set(0.5); Ret(p);"
   #tag GridJitY  =  "Arg(u,v); p=Set(0.5); Ret(p);"
   #tag Var_Scale =  "Arg(u,v); p=Rndu( 0.05,0.10);  Ret(p);"
   #tag Var_Yaw   =  "Arg(u,v); p=Rndu(-15.0,15.0); Ret(p);"
   #tag Var_Pitch =  "Arg(u,v); p=Rndu(-15.0,15.0); Ret(p);"
   #tag Var_Roll  =  "Arg(u,v); p=Rndu(-90.0,90.0); Ret(p);"

   #tag ObjSource =  (      O_OBJ_SHROOM_A ,     - )
   #tag SetMap    =  (     S_MAPLUMPS_EX01 )
   #tag NotMap    =  (                NONE )
   #tag SetCode   =  (          0x00FF0000 )
   #tag NotCode   =  (          0x00000000 )
   #tag Cluster_N =  (         1 ,       1 ) ' Per cluster range count
   #tag Cluster_R =  (       0.0 ,     0.0 ) ' Scaled units
   #tag Cluster_F =  (                NONE ) ' Scaling function of units above
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 6: Object scattering procedure, low-z trees, ecosystem functions. Generated all together.**

```
##define (S_LIVEADLER_MAP) as <SURFACE>
   #tag File      =  "Files_Media\Map_EnvSuitability_TreeAdler_Ex01.bmp"
   #tag Definer   =  (       DEF_S2D )
   #tag LoadNow   =  (             1 )

##define (F_LIVEADLER_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); r=LMap2(x,y); Ret(r); "
   #tag ZMap      =  (   S_LIVEADLER_MAP )
   #tag ZMin      =  (         0.000 )
   #tag ZMax      =  (         1.000 )
##define (F_SSRVADLER_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y,q,d); m=lerp(d); m=mul(m,q); m=clamp(m,0.0,1.0); Ret(m); "
   #tag Data      =  (         0.000 ,      0.000 )
   #tag Data      =  (         0.050 ,      0.000 )
   #tag Data      =  (         0.080 ,      1.000 )
   #tag Data      =  (         0.150 ,      0.000 )
   #tag Data      =  (         1.000 ,      0.000 )

##define (F_MORTADLER_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y,q,a); m=Set(0.010); m=clamp(m,0.0,1.0); Ret(m); "


##define (S_LIVEPALM_MAP) as <SURFACE>
   #tag File      =  "Files_Media\Map_EnvSuitability_TreePalm_Ex01.bmp"
   #tag Definer   =  (       DEF_S2D )
   #tag LoadNow   =  (             1 )

##define (F_LIVEPALM_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); r=LMap2(x,y); Ret(r); "
   #tag ZMap      =  (   S_LIVEPALM_MAP )
   #tag ZMin      =  (         0.000 )
   #tag ZMax      =  (         1.000 )
##define (F_SSRVPALM_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y,q,d); m=lerp(d); m=mul(m,q); m=clamp(m,0.0,1.0); Ret(m); "
   #tag Data      =  (         0.000 ,      0.000 )
   #tag Data      =  (         0.050 ,      0.000 )
   #tag Data      =  (         0.080 ,      1.000 )
   #tag Data      =  (         0.150 ,      0.000 )
   #tag Data      =  (         1.000 ,      0.000 )

##define (F_MORTPALM_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y,q,a); m=Set(0.010); m=clamp(m,0.0,1.0); Ret(m); "
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 6: Object scattering procedure, low-z trees. Trees are rendered as billboards.**

```
##define (PROCSYSGEN_TREESLOW_EX01) as <PROCDAT_SCATTER_2FV>
   #tag MethodPos =  (  PROC_PSYS_ECOPLANT )
   #tag MethodVar =  (  PROC_PSYS_ECOPLANT )
   #tag DimPos    =  (                   2 )
   #tag DimVar    =  (                   1 )
   #tag Physical  =  (                 YES )
   #tag Seed      =  (                 777 )
   #tag Range_X   =  (     -12.6 ,    12.6 ) ' Discretization related
   #tag Range_Y   =  (     -12.6 ,    12.6 ) ' Discretization related
   #tag GridSize  =  (     0.050 ,   0.050 ) ' Discretization related
   #tag GridJitX  =  "Arg(u,v); p=RndU(0.25,0.75); Ret(p);"
   #tag GridJitY  =  "Arg(u,v); p=RndU(0.25,0.75); Ret(p);"
   #tag MaxOrgs   =  (                3000 )
   #tag NbSeasons =  (                 512 )

   @inf Species   =  (       id ,  suitability_map ,    mortality_fcn , #init , init_fac , #matur , #seed , spred ,    seed_surviv_fcn )
   #tag Species   =  ( TREE_ADL , F_LIVEADLER_EX01 , F_MORTADLER_EX01 ,    50 ,      1.0 ,     10 ,     4 ,  0.08 ,  F_SSRVADLER_EX01 )
   #tag Species   =  ( TREE_PLM ,  F_LIVEPALM_EX01 ,  F_MORTPALM_EX01 ,    30 ,      1.0 ,     10 ,     5 ,  0.08 ,   F_SSRVPALM_EX01 )

   #tag SpObjMap  =  (       O_OBJ_TREEADLER_N ,     TREE_ADL ,     0 ,      15 ,  0.50 ,  1.00 )
   #tag SpObjMap  =  (       O_OBJ_TREEADLER_R ,     TREE_ADL ,    15 ,      30 ,  0.50 ,  1.00 )
   #tag SpObjMap  =  (       O_OBJ_TREEADLER_N ,     TREE_ADL ,    30 ,      60 ,  0.66 ,  1.00 )
   #tag SpObjMap  =  (       O_OBJ_TREEADLER_R ,     TREE_ADL ,    60 ,      90 ,  1.00 ,  1.00 )
   #tag SpObjMap  =  (       O_OBJ_TREEADLER_N ,     TREE_ADL ,    90 ,     120 ,  1.00 ,  1.00 )
   #tag SpObjMap  =  (       O_OBJ_TREEADLER_R ,     TREE_ADL ,   120 , 9999999 ,  1.00 ,  1.00 )
   #tag SpObjMap  =  (        O_OBJ_TREEPALM_N ,     TREE_PLM ,     0 ,      15 ,  0.50 ,  1.00 )
   #tag SpObjMap  =  (        O_OBJ_TREEPALM_R ,     TREE_PLM ,    15 ,      30 ,  0.50 ,  1.00 )
   #tag SpObjMap  =  (        O_OBJ_TREEPALM_N ,     TREE_PLM ,    30 ,      60 ,  0.66 ,  1.00 )
   #tag SpObjMap  =  (        O_OBJ_TREEPALM_R ,     TREE_PLM ,    60 ,      90 ,  1.00 ,  1.00 )
   #tag SpObjMap  =  (        O_OBJ_TREEPALM_N ,     TREE_PLM ,    90 ,     120 ,  1.00 ,  1.00 )
   #tag SpObjMap  =  (        O_OBJ_TREEPALM_R ,     TREE_PLM ,   120 , 9999999 ,  1.00 ,  1.00 )
```

- **One can use the same material at different TC orientations to create an illusion of diversity.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 7: Generation procedure, tile definers. These derive from world layer and shader designs.**

```
##define (VTD_LAYER_GRND_EX01) as <DC_GVTXDEF3>
   #tag IsFVF     =  (            NO )
   #tag IsPixl    =  (            NO )
   #tag IsNorm    =  (           YES )
   #tag IsDiff    =  (            NO )
   #tag IsSpec    =  (            NO )
   #tag IsTexN    =  (             2 )
   #tag IsSkin    =  (             0 )
   #tag IsBndx    =  (            NO )
   #tag IsSprt    =  (            NO )
   #tag Memory    =  (   GMEM_DEVICE )
   #tag IndexFmt  =  (       INDEX16 )
   #tag IsDynamic =  (             - )
   #tag IsWrtonly =  (             - )

##define (VTD_LAYER_WATR_EX01) as <DC_GVTXDEF3>
   #tag IsFVF     =  (            NO )
   #tag IsPixl    =  (            NO )
   #tag IsNorm    =  (            NO )
   #tag IsDiff    =  (           YES )
   #tag IsSpec    =  (            NO )
   #tag IsTexN    =  (             2 )
   #tag IsSkin    =  (             0 )
   #tag IsBndx    =  (            NO )
   #tag IsSprt    =  (            NO )
   #tag Memory    =  (   GMEM_DEVICE )
   #tag IndexFmt  =  (       INDEX16 )
   #tag IsDynamic =  (             - )
   #tag IsWrtonly =  (             - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 7: Generation procedure, near LoD tile defs. This is where the sausage is really made ...

```
##define (F_GRIDHEIGHT_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); h=Fcn1(x,y,0.0,0.0); r=Gt(h,0.0,h,0.0);    Ret(r); "
   #tag Fcn1      =  (    F_ELEV_EX01 )

##define (F_LAYER_WATR_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); h=Fcn1(x,y,0.0,0.0); r=Gt(h,0.0,0.0,h);    Ret(r); "
   #tag Fcn1      =  (    F_ELEV_EX01 )

##define (F_OCLUD_WATR_EX01) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); h=Fcn1(x,y,0.0,0.0); r=Gt(h,0.0,-1.0,1.0); Ret(r); "
   #tag Fcn1      =  (    F_ELEV_EX01 )


##define (PROC_WGEN_EX01) as <PROCWRL_ASM_RT3A>

   #tag Tile_Layers       =  (                        3 )
   #tag Tile_Numbers      =  (  0 ,       72 ,       72 )
   #tag Tile_SideSize     =  (  0 ,     0.35 ,     0.35 )
   #tag Tile_Vertices     =  (  0 ,        8 ,        8 )
   #tag Tile_Definer      =  (  0 , VTD_LAYER_GRND_EX01 )
   #tag Tile_ElevationF   =  (  0 ,         F_ELEV_EX01 )
   #tag Tile_CarvingF     =  (  0 ,                NONE )
   #tag Tile_UV0_Method   =  (  0 ,     PROC_UVSYS_TILE )
   #tag Tile_UV1_Method   =  (  0 ,    PROC_UVSYS_WORLD )
   #tag Tile_Kinex        =  (  0 ,     NONE ,        - )
   #tag Tile_BoxOverride  =  (  0 ,     0.00 ,     0.00 )

   #tag Tile_Numbers      =  (  1 ,      72 ,        72 )
   #tag Tile_SideSize     =  (  1 ,    0.35 ,      0.35 )
   #tag Tile_Vertices     =  (  1 ,       6 ,         6 )
   #tag Tile_Definer      =  (  1 , VTD_LAYER_WATR_EX01 )
   #tag Tile_ElevationF   =  (  1 ,   F_LAYER_WATR_EX01 )
   #tag Tile_CarvingF     =  (  1 ,   F_OCLUD_WATR_EX01 )
   #tag Tile_UV0_Method   =  (  1 ,     PROC_UVSYS_TILE )
   #tag Tile_UV1_Method   =  (  1 ,    PROC_UVSYS_WORLD )
   #tag Tile_Kinex        =  (  1 ,     NONE ,        - )
   #tag Tile_BoxOverride  =  (  1 ,    -0.10 ,     0.10 )

. . . continued
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 7: Generation, near LoD tile notes:

- Most worldgen procedures use the concept of tile layers. These are layers of pure geometry, confined to the procedure only. Tile layers are mapped to world layers.
- Is is important to have a consistent match between tile discretization, DCT sizes, object procedures, and partitions. Note that water tiles do not need the resolution of land tiles.
- The individual tile kinexes are deactivated, because each tile carries exact XYZ stream data, and the shader constants are preset at the head of the tile layers by a dummy object.
- Water tiles should have Z=0, but carry z data for shader FX nevertheless. Their fZ elevation is truncated for negative values only to ensure proper shore match. Furthermore, an artificial box elevation is specified for them.
- An carving mask is used to truncate vertices from water tiles that will not render anyway.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

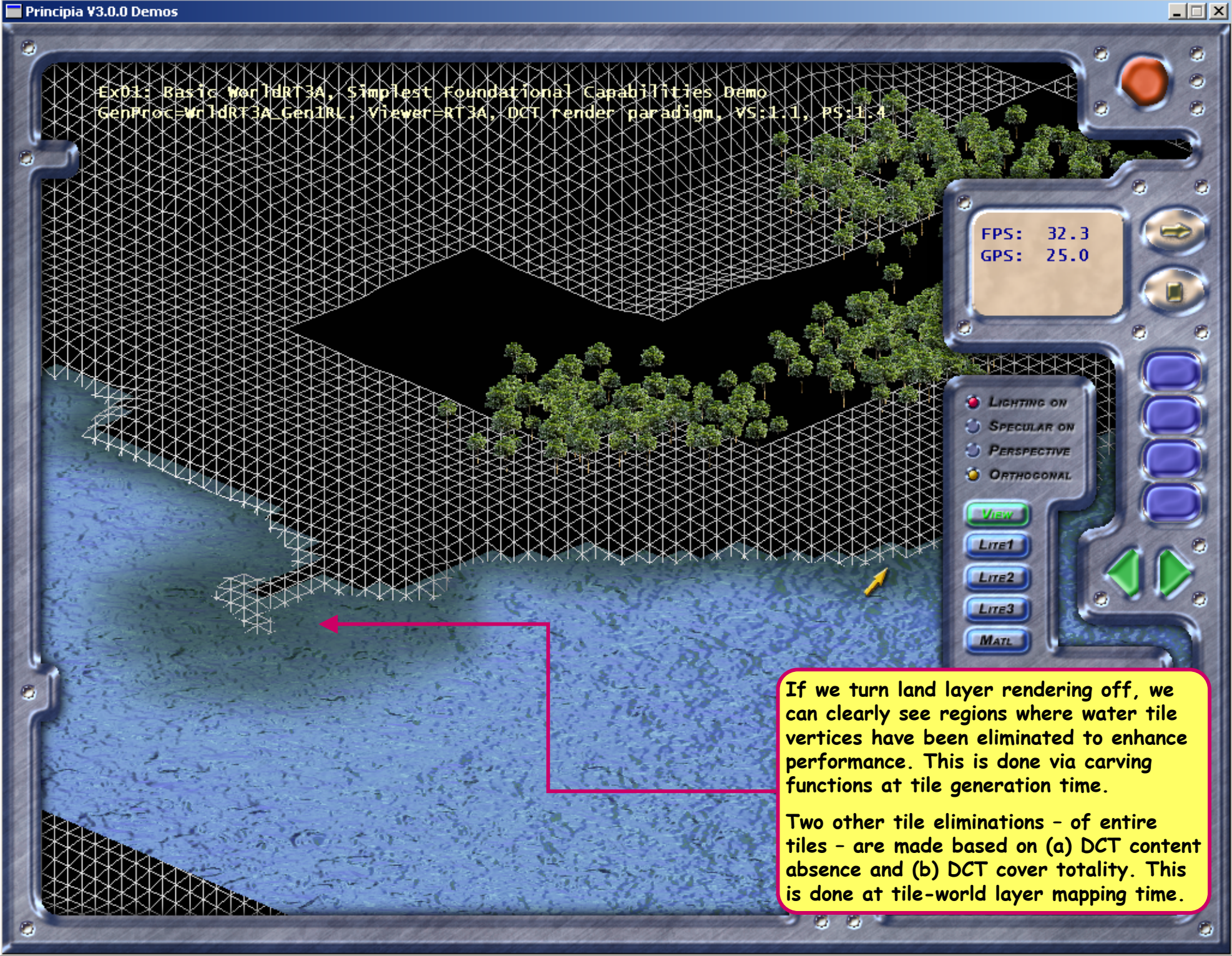
D108B:Ex01 – WorldRT3A

Step 7: Generation procedure, far LoD tiles.

```
#tag Tile_Numbers      =  (  2 ,       12 ,       12 )
#tag Tile_SideSize     =  (  2 ,     2.10 ,     2.10 )
#tag Tile_Vertices     =  (  2 ,       16 ,       16 )
#tag Tile_Definer      =  (  2 , VTD_LAYER_GRND_EX01 )
#tag Tile_ElevationF   =  (  2 ,         F_ELEV_EX01 )
#tag Tile_CarvingF     =  (  2 ,                NONE )
#tag Tile_UV0_Method   =  (  2 ,     PROC_UVSYS_TILE )
#tag Tile_UV1_Method   =  (  2 ,    PROC_UVSYS_WORLD )
#tag Tile_Kinex        =  (  2 ,      NONE ,       - )
#tag Tile_BoxOverride  =  (  2 ,     0.00 ,     0.00 )
```

- Only a single set of tiles can be used at far LoD, for both water and land.
- Far LoD tiles do not need vertex culling, hence no need for carving function.
- Far LoD tiles should still fit within the world partitions at integer counts (for RECT2D only).

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Simplest Foundational Capabilities Demo
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 32.3
GPS: 25.0
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
If we turn land layer rendering off, we can clearly see regions where water tile vertices have been eliminated to enhance performance. This is done via carving functions at tile generation time.
Two other tile eliminations – of entire tiles – are made based on (a) DCT content absence and (b) DCT cover totality. This is done at tile-world layer mapping time.

D108B:Ex01 – WorldRT3A

Step 7: Generation procedure, definition of the world Z (elevation) and T (local type) grids.

```
#tag Grid_TileSource   =  (                      0 )
#tag Grid_ElevationF   =  (        F_GRIDHEIGHT_EX01 )
#tag Grid_TypeExport   =  (             S_WTYPE_EX01 )
#tag Grid_TypeImport   =  (                    NONE )
#tag Grid_TypeAsmMap   =  (    S_DCT_LAYER_GRND_EX01 ,    TC_SOIL ,   TC_ROCK ,  TC_CLIFF ,          - )
#tag Grid_TypeAsmMap   =  (    S_DCT_LAYER_COVR_EX01 ,    TC_ROAD ,  TC_GRASS ,   TC_SNOW ,          - )
#tag Grid_TypeAsmMap   =  (    S_DCT_LAYER_WATR_EX01 ,   TC_WATER ,         - ,         - ,   TC_MOUNT )
```

- The Z grid is taken from the substrate tile layer.
- The DWORD type code grid can be exported to a data texture for subsequent review and editing.
- An initial population of the type grid based on DCT channel content is also effected at this time.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 7: Generation procedure, mapping of procedural tile sets to world layers.

```
@@@@ Wrld_Layer_Map    =  (  TileIndex+++WrldLayerID ,    VisiMask+TileViewableEvaluationFlag+EvalCode ,      CoverMask+CoverViewableEvaluationCodeForTile )

#tag Wrld_Layer_Map    =  (  0 ,        L_TILES_G1N ,  S_DCT_LAYER_GRND_EX01 ,       NO , 0xFFFFFFFF , S_DCT_LAYER_COVR_EX01 , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Map    =  (  0 ,        L_TILES_G2N ,  S_DCT_LAYER_COVR_EX01 ,      YES , 0x00FFFFFF ,                   NONE , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Map    =  (  1 ,        L_TILES_W1N ,  S_DCT_LAYER_WATR_EX01 ,       NO , 0xFFFFFFFF ,                   NONE , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Map    =  (  2 ,        L_TILES_G1F ,  S_DCT_LAYER_GRND_EX01 ,       NO , 0xFFFFFFFF ,                   NONE , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Map    =  (  2 ,        L_TILES_G2F ,  S_DCT_LAYER_COVR_EX01 ,      YES , 0x00FFFFFF ,                   NONE , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Map    =  (  2 ,        L_TILES_W1F ,  S_DCT_LAYER_WATR_EX01 ,      YES , 0x00FF0000 ,                   NONE , NONE , NONE ,  0x00FFFFFF )
```

This is the crux of the generation process:

- Reference linkages are made from tile to world layers
- DCT masks for elimination of “no content” tiles can be provided, together with channel pattern match code.
- Up to three cover DCT masks for elimination of “fully covered” tiles can be provided, with a match code.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 7: Generation procedure, placement of objects from particle procedure outputs.**

```
#tag Obj_ProcGen  = (  L_OBJ_SOLIDS ,   PROCSYSGEN_SHROOMS_EX01 , TC_LUMPS ,  -1.0, 1.0 , -1.0, 1.0 , GRID_BOXRECT , GRID_NODEREF ,   NONE )
#tag Obj_ProcNot  = (      TC_CLIFF ,   PROCSYSGEN_SHROOMS_EX01 ,             -2.0, 2.0 , -2.0, 2.0 , GRID_METRECT , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (       TC_ROAD ,   PROCSYSGEN_SHROOMS_EX01 ,              3.0, 0.0 ,  3.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_WATER ,   PROCSYSGEN_SHROOMS_EX01 ,              2.0, 0.0 ,  2.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_LUMPS ,   PROCSYSGEN_SHROOMS_EX01 ,              1.0, 0.0 ,  1.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )

#tag Obj_ProcGen  = (  L_OBJ_SOLIDS ,  PROCSYSGEN_BOULDERS_EX01 , TC_LUMPS ,  -0.0, 0.0 , -0.0, 0.0 , GRID_BOXRECT , GRID_NODEREF ,   NONE )
#tag Obj_ProcNot  = (       TC_ROAD ,  PROCSYSGEN_BOULDERS_EX01 ,              3.0, 0.0 ,  3.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_LUMPS ,  PROCSYSGEN_BOULDERS_EX01 ,              1.0, 0.0 ,  1.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )

#tag Obj_ProcGen  = (  L_OBJ_TREES1 ,  PROCSYSGEN_TREESLOW_EX01 ,  TC_TREE ,  -1.0, 1.0 , -1.0, 1.0 , GRID_METRECT , GRID_NODEREF ,   NONE )
#tag Obj_ProcNot  = (      TC_CLIFF ,  PROCSYSGEN_TREESLOW_EX01 ,             -0.0, 0.0 , -0.0, 0.0 , GRID_METRECT , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (       TC_ROAD ,  PROCSYSGEN_TREESLOW_EX01 ,              3.0, 0.0 ,  3.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_WATER ,  PROCSYSGEN_TREESLOW_EX01 ,              2.0, 0.0 ,  2.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_LUMPS ,  PROCSYSGEN_TREESLOW_EX01 ,              2.0, 0.0 ,  2.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )

#tag Obj_ProcGen  = (  L_OBJ_TREES2 , PROCSYSGEN_TREESHIGH_EX01 ,  TC_TREE ,  -1.0, 1.0 , -1.0, 1.0 , GRID_METRECT , GRID_NODEREF ,   NONE )
#tag Obj_ProcNot  = (       TC_TREE , PROCSYSGEN_TREESHIGH_EX01 ,             -0.0, 0.0 , -0.0, 0.0 , GRID_METRECT , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (       TC_SOIL , PROCSYSGEN_TREESHIGH_EX01 ,             -0.0, 0.0 , -0.0, 0.0 , GRID_METRECT , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (       TC_ROAD , PROCSYSGEN_TREESHIGH_EX01 ,              3.0, 0.0 ,  3.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_WATER , PROCSYSGEN_TREESHIGH_EX01 ,              1.0, 0.0 ,  1.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
#tag Obj_ProcNot  = (      TC_LUMPS , PROCSYSGEN_TREESHIGH_EX01 ,              2.0, 0.0 ,  2.0, 0.0 , GRID_METCIRC , GRID_NODEREF ,   0.01 )
```

- **Note the conditionals between object placements and terrain type grid. Principia features grid testing methods available to the user application needing to maintain such interaction at runtime.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Footprint establishment codes:

- Footprint and cell scope marking codes (part of the powerful Principia topology operations library if you are interested to delve deeper)
- Determine where to place procedural objects or not, based on marking over the <GridT> field, and to mark the field upon placing.

Footprint establishment modality codes:

- BoxRECT: based on object actual box rectangular footprint. The extents are +/-XY bands in addition to the box size.
- BoxCIRC: based on object actual box elliptical footprint. The extents are +Rx/0/+Ry/0 ellipse radii additions.
- MetRECT: uses rectangular index footprint. The extents are the ranges of +/-IJ node reach from the object IJ center.
- MetCIRC: uses elliptical integer index footprint. The extents are +I/0/+J/0 ellipse radii around object IJ center.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Cell scope modality:

- NODEREF: only the (I,J) node is marked or tested
- CELLREF: same as NODEREF.
- QUADREF: marks/tests (I,J) (I+1,J) (I,J+1) and (I+1,J+1) cell corners (four taps). The modality thus defines a “shaped footprint”
- CROSREF: accesses a five-tap cross-pattern fixed on the (I,J) node.
- EXEXREF: accesses a five-tap X-pattern centered on the (I,J) node.
- CROSREF: accesses a nine-tap square box around the (I,J) node.
- SIDEREF: accesses a four-tap cross-pattern around, but excluding the (I,J) node.
- DIAGREF: accesses a four-tap X-pattern around, but excluding the (I,J) node.
- CIRCREF: accesses an eight-tap circular pattern around, but excluding the (I,J) node.

Cell scope usage:

- Cell scope modality determines how the contents of each (I,J) node is addressed by state setting, testing and deletion procedures.
- These codes are used in a variety of other operations to define a pattern of data manipulation in a grid – e.g. for 2D filtering or feature extraction.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Footprint coverage threshold:

- Testing operations that take place over a footprint return as a result the fraction of nodes that meets the test (result=0.0 means no nodes pass, result=1.0 means all nodes pass).
- All footprint testing allows the user to specify the fraction threshold to use to determine how many nodes must test true to make the footprint test to pass. This enables to define very specific tests.
- Using a small threshold (e.g. ~0.001) means the test will pass if a single node passes. A large threshold (e.g. ~0.999) means the test will pass if all nodes in the footprint pass. ***Do not use thresholds of 0.0 or 1.0 because of rounding errors.***
- The count of nodes in the footprint is determined by the extents and not by the cell scope mode. Thus the coverage fraction is evaluated independent of the cell scope modality. This enables one to construct some pretty complex logical tests, e.g. find how many X-shapes in a given footprint meet a given condition.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

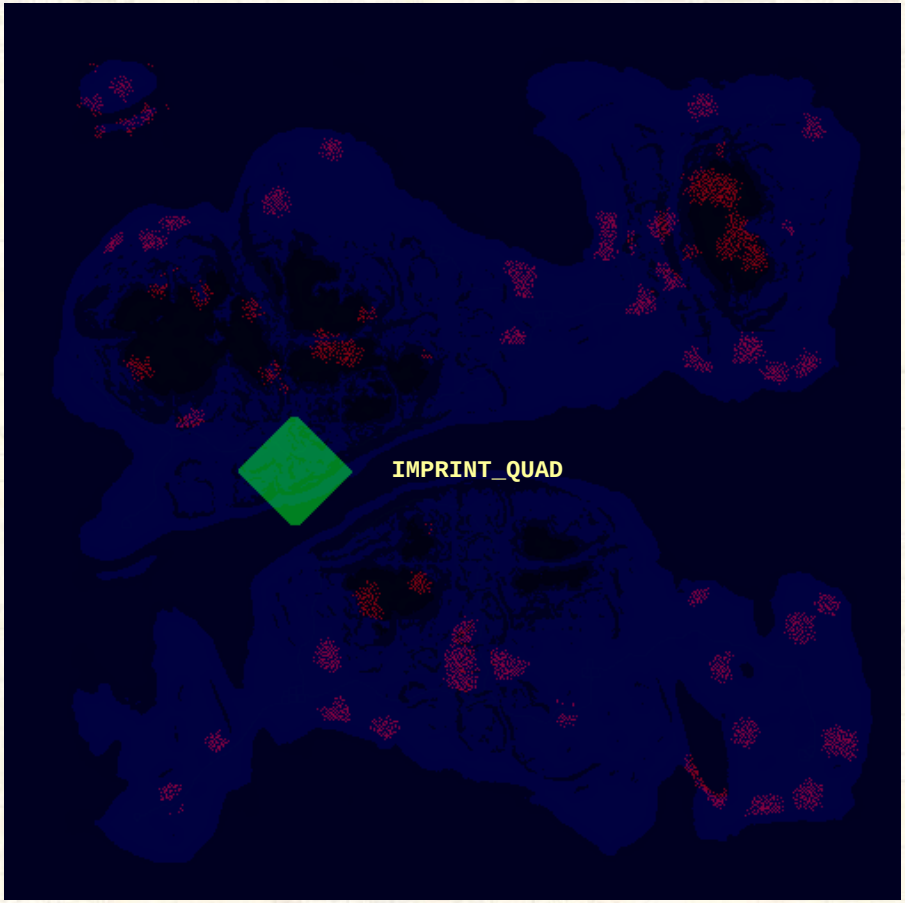
D108B:Ex01 – WorldRT3A

- **Principia uses the foregoing control parameters consistently in all situations requiring footprint testing and marking (e.g. object placers):**
  - Footprint extents (4 floats)
  - Footprint modality code
  - Cell scope modality code
  - Footprint coverage threshold
- **MetRECT footprints are very common, and it is important to understand the use of extents:**
  - Extents define the rectangular space address to be tested around the IJ node with **exclusive** upper bounds
  - A single node footprint is (0,0,0,0) OR (0,1,0,1).
  - A 4-node footprint for the entire cell is (0,2,0,2)
  - A 9-node square footprint fixed on the IJ node is (-1,2,-1,2)

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Imprint marking is used for 3D objects with variable scale. It helps prevent object overlaps.**



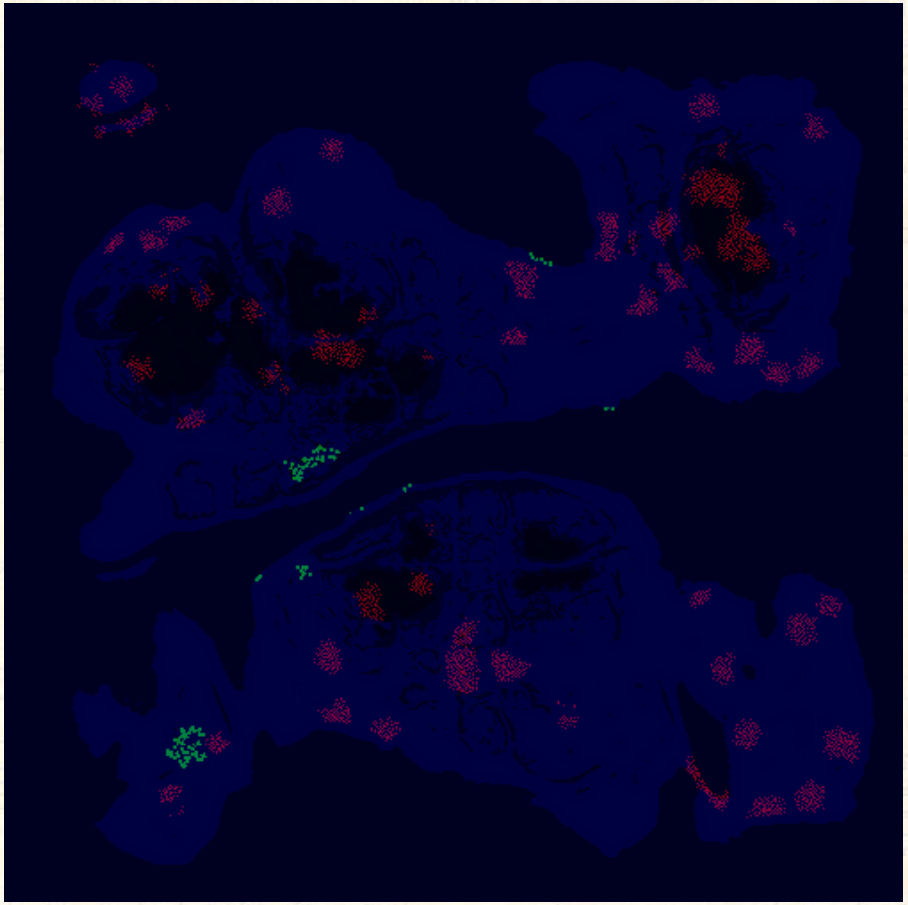
Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 7: Generation procedure. This finally generates the world and saves a GridT image.**

```
##run (PROC_WGEN_EX01) with target (WRLD_EX01)

##save (S_WTYPE_EX01) in
      ("Files_Media/Map_TerrainType_C.bmp") alpha
      in (Files_Media/Map_TerrainType_X.bmp")
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 8: World viewer, cursor subcomponent. The cursor is a basic cell cursor from the RT3A family. It over-renders the cell with the given material.**

```
##define (S_CRS_EX01) as <SURFACE>
   #tag File      =  "Files_Media/Tex_WordlCursor_Shadetex_C.bmp"
   #tag Alpha     =  "Files_Media/Tex_WordlCursor_Shadetex_X.bmp"
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )

##define (M_CRS_EX01) as <MATERIAL_3A>
   #tag RS      =  (               RS_LIGHTING ,             1 )
   #tag RS      =  (               RS_FILLMODE ,        RA_SOLID )
   #tag RS      =  (               RS_CULLMODE ,     RA_CULLNONE )
   #tag RS      =  (                RS_ZENABLE ,             1 )
   #tag RS      =  (           RS_ZWRITEENABLE ,             0 )
   #tag RS      =  (              RS_DEPTHBIAS ,        -0.00004 )
   #tag TS      =  (                TS_COLOROP ,   TO_SELECTARG1 ,   0 )
   #tag TS      =  (              TS_COLORARG1 ,      TA_TEXTURE ,   0 )
   #tag TS      =  (              TS_COLORARG2 ,      TA_DIFFUSE ,   0 )
   #tag TS      =  (                TS_ALPHAOP ,   TO_SELECTARG1 ,   0 )
   #tag TS      =  (              TS_ALPHAARG1 ,      TA_TEXTURE ,   0 )
   #tag TS      =  (              TS_ALPHAARG2 ,      TA_DIFFUSE ,   0 )
   #tag RS      =  (               RS_ALPHAREF ,             1 )
   #tag RS      =  (              RS_ALPHAFUNC ,      CA_GREATER )
   #tag RS      =  (       RS_ALPHABLENDENABLE ,             1 )
   #tag RS      =  (        RS_ALPHATESTENABLE ,             1 )
   #tag RS      =  (               RS_SRCBLEND ,     FA_SRCALPHA )
   #tag RS      =  (              RS_DESTBLEND ,  FA_INVSRCALPHA )
   #tag TX      =  (           S_CRS_EX01 ,  0 )
   #tag TX      =  (                NONE ,  1 )
   #tag VShader =  (                     NONE )
   #tag PShader =  (                     NONE )

##define (WCRS_EX01) as <WORLDCURSOR_3R>
   #tag Grid_Exclude  =  (      TC_WATER ,   0.00 , 0.00 , 0.00 , 0.00 , GRID_METRECT , GRID_NODEREF ,  0.001 )
   #tag Grid_Exclude  =  (      TC_MOUNT ,   0.00 , 0.00 , 0.00 , 0.00 , GRID_METRECT , GRID_NODEREF ,  0.001 )
   #tag Cell_Build    =  (  VTD_FVF_FILE ,    NONE ,  L_WCURS )
   #tag Cell_Display  =  (             0 ,   M_CRS ,      NONE)
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 8: World viewer and camera kbd-controller.

- Because there are two hidden grid layers and we want to toggle both with one key, the KJAY controller generates the X_GRID signal twice.
- The <ObjectTrack> tag tells the viewer to compile object interaction information for the named layers.

```
##define (KJAY_EX01) as <KJAY_2A>
   #method        =  (    ON_KEYPRS  ,    CH_WRL ,     X_GRID  ,       G ) ' Once for the near layer
   #method        =  (    ON_KEYPRS  ,    CH_WRL ,     X_GRID  ,       G ) ' Twice for the far layer
   #method        =  (    ON_KEYPRS  ,    CH_WRL ,     X_DIAG  ,       A ) ' Toggle diagnostics
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_DISTN  ,     PLS ) ' Distance zoom near
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_DISTF  ,     MNS ) ' Distance zoom far
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_ZOOMN  ,     COM ) ' Lens angle zoom near
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_ZOOMF  ,     DOT ) ' Lens angle zoom far
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_ELEVU  ,       U ) ' Elevation up
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_ELEVD  ,       D ) ' Elevation down
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_AZIML  ,       L ) ' Distance zoom near
   #method        =  (    ON_KEYPRS  ,    CH_NAV ,    X_AZIMR  ,       R ) ' Distance zoom far

##define (VIEW_EX01) as <WORLDVIEWER_RT3A>
   #tag World         =  (     WRLD_EX01 )
   #tag Camera        =  (   CAMERA_ST3D )
   #tag Cursor        =  (     WCRS_EX01 )
   #tag PlateBtm      =  (     KJAY_EX01 ,          - ,         - )
   #tag PlateTop      =  (          NONE ,          - ,         - )
   #tag ObjectTrack   =  (               L_OBJ_SOLIDS ,         - ,         - )
   #tag ObjectTrack   =  (               L_OBJ_TREES1 ,         - ,         - )
   #tag ObjectTrack   =  (               L_OBJ_TREES2 ,         - ,         - )
   #tag LodVarExt     =  (      V_LODCTL ,   LOD_NEAR ,       0.0 ,      12.0  )
   #tag LodVarExt     =  (      V_LODCTL ,    LOD_FAR ,      12.0 ,   99999.9  )
   #method            =  ( EXE_TOGGLTYPE ,     CH_WRL ,    X_GRID ,    L_GRIDN ,    LWT_HIDE )
   #method            =  ( EXE_TOGGLTYPE ,     CH_WRL ,    X_GRID ,    L_GRIDF ,    LWT_HIDE )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 8: Camera methods added in NAV3D module to enable keyboard-driven world navigation.**

```
#tag UserCtl_PanW      =  (                200 )
#tag UserCtl_PanH      =  (                150 )
#tag UserCtl_DistN     =  "Arg(d); d=Dif(d,0.5); Ret(d);"
#tag UserCtl_DistF     =  "Arg(d); d=Add(d,0.5); Ret(d);"
#tag UserCtl_DistExt   =  (      2.0 ,    150.0 )
#tag UserCtl_ZoomN     =  "Arg(a); a=Dif(a,2.0); Ret(a);"
#tag UserCtl_ZoomF     =  "Arg(a); a=Add(a,2.0); Ret(a);"
#tag UserCtl_ZoomExt   =  (      2.0 ,    150.0 )
#tag UserCtl_ElevU     =  "Arg(a); a=Add(a,2.0); Ret(a);"
#tag UserCtl_ElevD     =  "Arg(a); a=Dif(a,2.0); Ret(a);"
#tag UserCtl_ElevExt   =  (    -88.0 ,     88.0 )
#tag UserCtl_AzimL     =  "Arg(a); a=Add(a,5.0); Ret(a);"
#tag UserCtl_AzimR     =  "Arg(a); a=Dif(a,5.0); Ret(a);"
#tag UserCtl_AzimExt   =  ( -99999.0 ,  99999.0 )
#method                =  (    EXE_PANU ,    CH_NAV ,     X_PANU )
#method                =  (    EXE_PAND ,    CH_NAV ,     X_PAND )
#method                =  (    EXE_PANL ,    CH_NAV ,     X_PANL )
#method                =  (    EXE_PANR ,    CH_NAV ,     X_PANR )
#method                =  (   EXE_DISTN ,    CH_NAV ,    X_DISTN )
#method                =  (   EXE_DISTF ,    CH_NAV ,    X_DISTF )
#method                =  (   EXE_ZOOMN ,    CH_NAV ,    X_ZOOMN )
#method                =  (   EXE_ZOOMF ,    CH_NAV ,    X_ZOOMF )
#method                =  (   EXE_ELEVU ,    CH_NAV ,    X_ELEVU )
#method                =  (   EXE_ELEVD ,    CH_NAV ,    X_ELEVD )
#method                =  (   EXE_AZIML ,    CH_NAV ,    X_AZIML )
#method                =  (   EXE_AZIMR ,    CH_NAV ,    X_AZIMR )
#method                =  (     ON_PANS ,    CH_NAV ,     X_ULKX )
#method                =  (     ON_PANS ,    CH_NAV ,     X_ULKY )
#method                =  (     ON_PANS ,    CH_NAV ,     X_ULKZ )
#method                =  (     ON_DIST ,    CH_NAV ,     X_IVM1 )
#method                =  (     ON_ZOOM ,    CH_NAV ,     X_IVM4 )
#method                =  (     ON_ELEV ,    CH_NAV ,     X_IVM2 )
#method                =  (     ON_AZIM ,    CH_NAV ,     X_IVM3 )
```

- **The ON methods signal the control panel boxes to update their contents upon keyboard navigation.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Step 9: User application code for visualizing performance and world layer render statistics.**
  - Main purpose is to showcase how to access viewer and world internal layer and partition data structures.
  - Show number of visible partitions
  - Show total number of visibility tests (CPU load)
  - Show pointer cell UV index, elevation and type
  - Show pointed object UV index and parent layer
- **Such basic capabilities are usually needed in the design of application logic. Note that the detailed layer statistics are internally generated by Principia in development mode only.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 9: User application code for visualizing performance and world layer render statistics.

```
   /* Auxiliary local variables */
   DECLARE_LOCAL  static   WX_WorldRT3A*        WRL     = NULL;
   DECLARE_LOCAL  static   CX_WViewRT3A*        WVU     = NULL;
   DECLARE_LOCAL  static   CX_Textout2A*        TXA     = NULL;
   DECLARE_LOCAL  static   CX_Textout2A*        TXV     = NULL;
   DECLARE_LOCAL  static   PX_Signal*           DIA     = NULL;
   DECLARE_LOCAL  static   SX_String*           TsA     = NULL;
   DECLARE_LOCAL  static   SX_String*           TsV     = NULL;
   DECLARE_LOCAL  static   SX_String*           WrkStr  = NULL;
   DECLARE_LOCAL  static   SX_Variable*         VLD     = NULL;
   DECLARE_LOCAL  static   bool                 Diags   = true;
   DECLARE_LOCAL  static   int                  GridI   = 0x00;
   DECLARE_LOCAL  static   int                  GridJ   = 0x00;
   DECLARE_LOCAL  static   flt                  GridX   = ZERO;
   DECLARE_LOCAL  static   flt                  GridY   = ZERO;
   DECLARE_LOCAL  static   flt                  GridZ   = ZERO;
   DECLARE_LOCAL  static   int                  i,j,k,N = 0x00;
   DECLARE_LOCAL  static   DWORD                TC      = 0x00;
   DECLARE_LOCAL  static   AX_Object**          ObjSet  = NULL;
   DECLARE_LOCAL  static   AX_Object*           ObjWrk  = NULL;

   /* Obtain reference to application components */
   STANDARD_PRINCIPIA_REFERENCE(WRL, "WRLD_EX01"   , WX_WorldRT3A*);
   STANDARD_PRINCIPIA_REFERENCE(WVU, "VIEW_EX01"   , CX_WViewRT3A*);
   STANDARD_PRINCIPIA_REFERENCE(TXA, "TX_INFO_GEN" , CX_Textout2A*);
   STANDARD_PRINCIPIA_REFERENCE(TXV, "TX_INFO_LAY" , CX_Textout2A*);
   STANDARD_PRINCIPIA_REFERENCE(VLD, "V_LODCTL"    , SX_Variable* );

   /* Initialize page function contents */
   if (TsA==NULL) TsA = new SX_String(0x4FF);
   if (TsV==NULL) TsV = new SX_String(0x4FF);
   if (DIA==NULL) DIA = new PX_Signal("CH_WRL", "X_DIAG");

   /* Test whether diagnostics are to be shown */
   if (DIA->TestClear()) Diags = !Diags;
   if (!Diags) then{
      TXA->ExternStr = NULL;
      TXV->ExternStr = NULL;
      return;
   endif}

. . .
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 9: User application code for visualizing performance and world layer render statistics.

```
. . .

   /* Generate textout contents: render load */
   TsA->Assign("Proc_Load: ");
   i = 0; for (k=0; k<WRL->NSect; k++) if (WRL->SectorCode[k]) i++;
   TsA->Append("[%d] " , i);                                          // visible sectors
   TsA->Append("[%d] " , (int) Api_PerfMonitor->Post_BoxVisTestPF);  // box screen tests
   TsA->Append("[%d\\" , (int) Api_PerfMonitor->Post_NbObjVisPF  );  // visible objects
   TsA->Append("%d] "  , (int) Api_PerfMonitor->Post_NbObjTotPF  );  // total objects
   TsA->Append("[%d]"  , 0x00                                     );  // collision test

   /* Generate textout contents: pointed tiles */
   TsA->Append("\\n");
   TsA->Append("Over_Tile: ");
   if (WVU->PtrOver_Tile) then{
   for (k=0; k<WVU->PtrOver_Tile->N; k++) begin{
       ENDECODE_DWORD_TOTWOINT(WVU->PtrOver_Tile->ePtr[k]->Com32A,GridI,GridJ);
       TsA->Append("[%d,", GridI);
       TsA->Append("%d] ", GridJ);
   endfor}
   endif}

   /* Generate textout contents: world cursor coordinates */
   TsA->Append("\\n");
   TsA->Append("Over_Cell: ");
   if (WVU->PtrOver_Cell) then{
   for (k=0; k<WVU->PtrOver_Cell->N; k++) begin{
       GridI = WVU->PtrOver_Cell->ePtr[k];
       GridJ = WVU->PtrOver_Cell->aPtr[k];
       GridX = WRL->Xmin + float(GridI) * WRL->GridDX;  // also WRL->Cursor_GridX
       GridY = WRL->Ymin + float(GridJ) * WRL->GridDY;  // also WRL->Cursor_GridY
       GridZ = WRL->Elevation(GridX, GridY);            // also WRL->Cursor_GridZ
       TsA->Append("[%d,"    , GridI);
       TsA->Append("%d]="    , GridJ);
       TsA->Append("[%.3f," , GridX);
       TsA->Append("%.3f,"  , GridY);
       TsA->Append("%.3f]"  , GridZ);
   endfor}
   endif}

. . .
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Step 9: User application code for visualizing performance and world layer render statistics.

```
   /* Generate textout contents: world pointed objects */
   TsA->Append("\\n");
   TsA->Append("Cell_Code: ");
   if (WVU->Cursor_GridI!=INVALID) then{
   if (WVU->Cursor_GridJ!=INVALID) then{
      i  = WVU->Cursor_GridI;
      j  = WVU->Cursor_GridJ;
      k  = WRL->GridT->Get(i,j);
      TsA->Append("[%d"   , i);
      TsA->Append(",%d,"  , j);
      TsA->Append("%#x]"  , k);
   endfor}
   endfor}

   /* Generate textout contents: world pointed objects */
   TsA->Append("\\n");
   TsA->Append("Over_Obj : ");
   for (k=0; k<WRL->NbLayers; k++) begin{
       N = WRL->LayerGetStack(k, ObjSet);
       if (!(WRL->LayerTypeTest(k, LWT_TILE))) then{
       for (j=0; j<N; j++) begin{
           ObjWrk = ObjSet[j];
           if (ObjWrk->InteractionFlag & INTER_OVER) then{
              ENDECODE_DWORD_TOTWOINT(ObjWrk->Com32A,GridI,GridJ);
              TsA->Append("[%d," , GridI);
              TsA->Append("%d @" , GridJ);
              TsA->Append("%d]"  ,     k);
           endif}
       endfor}
       endif}
   endfor}

   /* Generate textout contents: viewer state data */
   TsA->Append("\\n");
   TsA->Append("View_Lod : ");
   WrkStr = Api_NameTable->FindKey(VLD->ValueI);
   if (WrkStr) TsA->Append(WrkStr->Str);

   /* Generate textout contents for world layer statistics and route display */
   WRL->DiagLayers(TsV);
   TXA->ExternStr = TsA->Str;
   TXV->ExternStr = TsV->Str;

END_ACTOR_CODE}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **The world in Ex01 has several shortcomings that prevent its use in a production application.**
- **Physical realism general shortcomings (A):**
  - Overall, the terrain texture appears flat and simply painted on, without proper balance between solidity and scatter, and without the natural mix of small and large scale detail.
  - Terrain elevation is not natural. There are no gullies, plateaus, taluses, and the water shore features an uniform depth plunge w/o rocks or beaches.
  - Base terrain layer substrate distribution, especially in areas near water, is too uniform and not realistic.
  - Cover layer distribution, especially grass, is too concentrated in uniform solid patches.
  - Roads do not look realistic at all. They are too blocky, XY aligned instead of weaving, look painted on the terrain, and are stretched based on slope.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

Render realism shortcomings (B):

- No object shadows
- No elevation self-shadows or sky illumination
- No realistic land feature reflections in the water
- All texture blends are smooth transitions
- Texture alignment on slope is XY, not natural wrap.
- Texture on slopes are stretched due to alignment.
- No bump mapping for small-scale textural detail
- Water blend with bottom texture creates color cast
- At large scale, textures are excessively pixelated
- At small scale, textures are blurred
- Grass textures lack 3D fuzziness and tone variation
- No atmospherics
- Some sharp lines at tile-texture boundaries.
- Waves do not sweep in elevation or hug the shoreline
- Grass and other in-texture features do not poke out sharply.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- **Object shortcomings (C):**
  - Landscape is mostly empty!
  - Same-species distributions are excessively clumpy
  - Not enough variation within same species
  - Different natural objects do not intermix
  - No natural groupings within object distributions
  - No variation of occurrence within distributions
  - No scale harmony within object distributions
  - No terrain variation around objects
  - Mountain slopes too smooth
  - At large scales, trees grainy and blend with landscape
  - Trees simply stuck in ground as toothpicks
  - Some trees stick through nearby high hills
  - Objects are static (trees do not move, no NPCs)
- **Performance shortcomings: low FPS on low-end hardware (even with partitioning and 2xLoD).**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

- These shortcomings will be addressed in the next demos using refinements of generation and render processes.
- They should not detract from what was accomplished in this first example:
  - We rendered a fairly complex, fully-featured, production-size world …
  - With reasonable quality and performance …
  - Without a single line of coding! We instead painted simple distribution maps to control the results.
- The only coding needed to create a simple game from this point on is your object and GUI logic!

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 – WorldRT3A

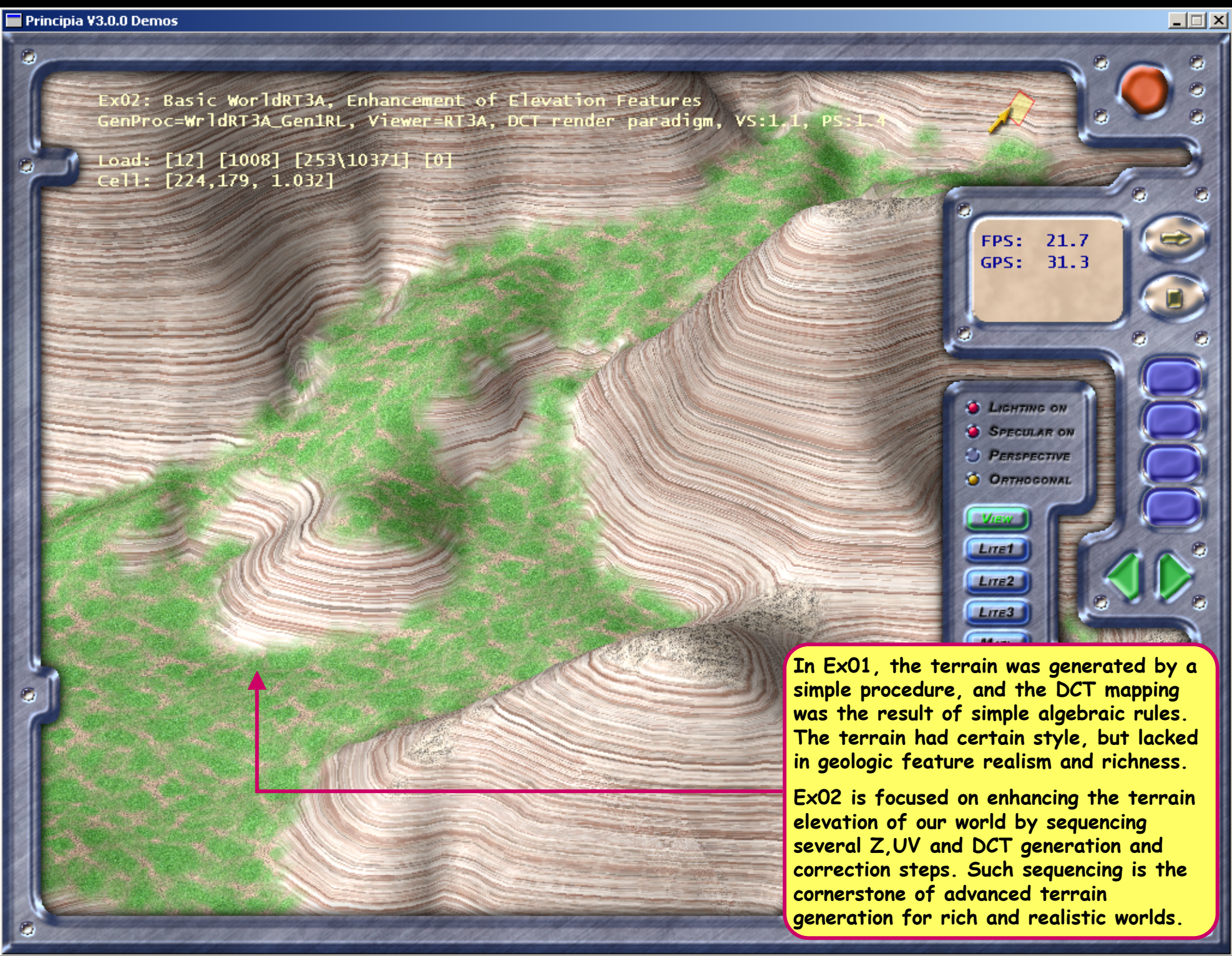
- **Most enhancements shall be based on refining components external to the world object:**
  - Geometry generation procedures
  - DC texture generation procedures
  - Artwork and manual DC textures
  - Particle scattering procedures
  - Shaders
- **All enhancement use the same WorldRT3A object standard generic features, e.g.**
  - Rectangular elevation and type meshes
  - Partitioning, cursor and object tracking ...etc.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex01 - Concepts

- Major choices for world visual design
- World layers, types and operations
- Partitioning and render sequencing
  - By material (minimize state changes)
  - By depth (minimize pixel flow)
  - By FX precedence (reflections, transparency)
  - Insertion of dummy objects and header materials
- Distribution control textures (DCT)
- Procedural contents generation
- Non-visible tile content elimination

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

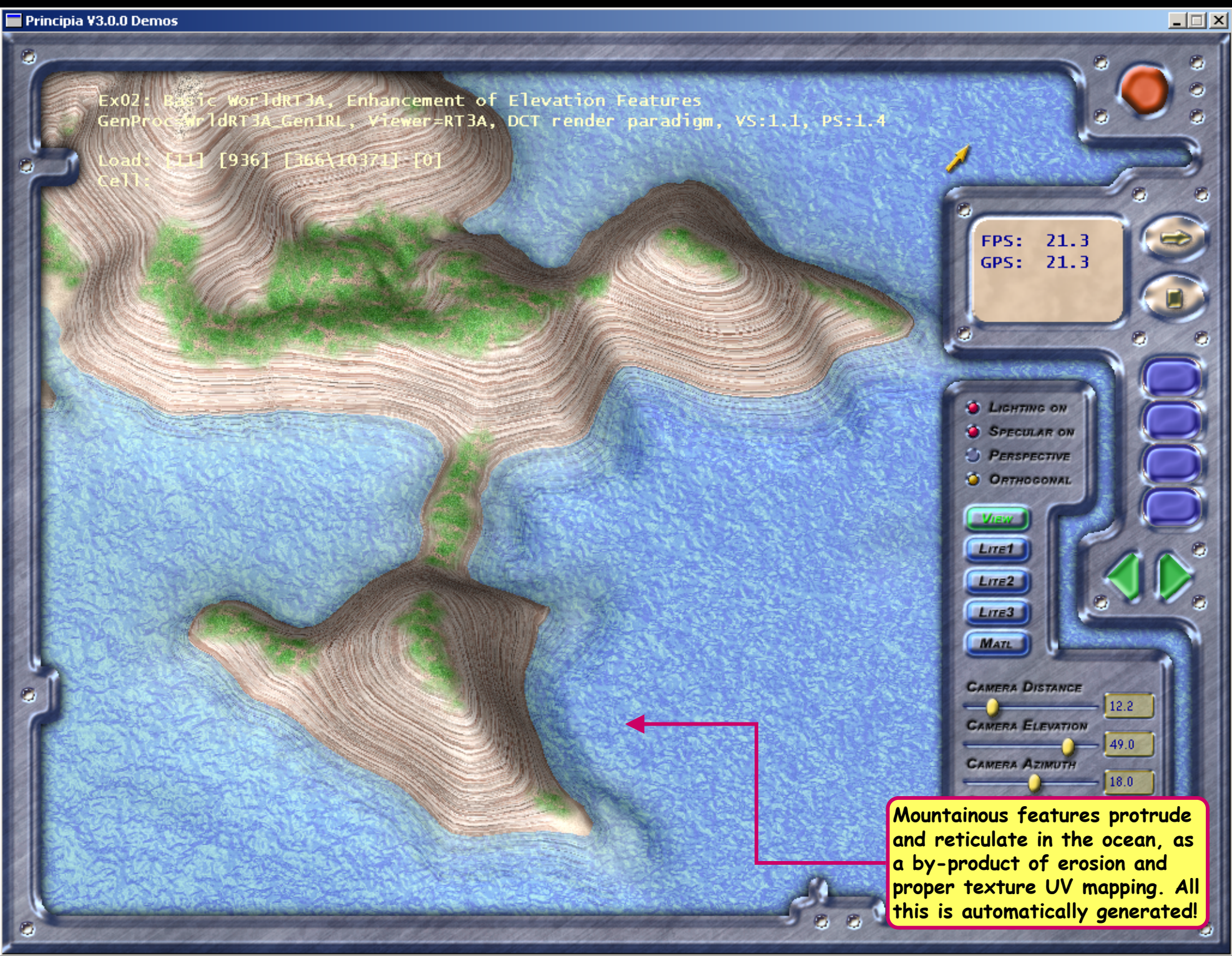

Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [12] [1008] [253\10371] [0]
Cell: [224,179, 1.032]
FPS: 21.7
GPS: 31.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
In Ex01, the terrain was generated by a simple procedure, and the DCT mapping was the result of simple algebraic rules. The terrain had certain style, but lacked in geologic feature realism and richness.
Ex02 is focused on enhancing the terrain elevation of our world by sequencing several Z,UV and DCT generation and correction steps. Such sequencing is the cornerstone of advanced terrain generation for rich and realistic worlds.


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1 1 PS:
Load: [15] [1224] [344\10371] [0]
Cell:
Our Ex02 world has taluses and carved canyons with smooth natural-looking rock layering, cascading alluvial plains, and an overall plausible distribution of water, land, cliffs and roughness.
FPS: 21.7
GPS: 21.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
7.7
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
5.4
4.1
0.0


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [8] [720] [310\10371] [0]
Cell: [249,443, 0.484]
FPS: 21.3
GPS: 21.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
7.7
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
1.1
9.6
0.0
Unlike Ex01, there are two types of seashore – smooth low beach, and dropping cliffs. There is also a mix of low and high plains, with terraces and slides.


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.
Load: [14] [1152] [285\10371] [0]
Cell: [189,316, 0.273]
Rivers (water rendering is covered later) meander naturally in their canyons, and open smoothly to sea. The sea bottom is contoured to give proper transparency transition to depth.
FPS: 21.3
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
7.7
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
-2.0
3.5
0.0


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [11] [936] [353\10371] [0]
Cell: [26,368, 1,198]
FPS: 21.3
GPS: 21.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
The use of an erosion procedure ensures that mountainous features have the grooved, folding topology of real mountains. There is a mix of plains at different elevations, cut by meandering rivers. There are isolated tali protruding from the plain without peaking. Note the good UV mapping of the river trace.


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProcWrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [11] [936] [366\10371] [0]
Cell:
FPS: 21.3
GPS: 21.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
12.2
Camera Elevation
49.0
Camera Azimuth
18.0
Mountainous features protrude and reticulate in the ocean, as a by-product of erosion and proper texture UV mapping. All this is automatically generated!


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:
Load: [10] [864] [354\10371] [0]
Cell: [45 20, 1.558]
Once again, note the properly carved-reticulated aspect of the mountains, and their joining with river beds. Unlike Ex01, mountains and plains have differing elevation and are automatically terraced based on the erosion procedure.
FPS: 21.3
GPS: 21.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
12.2
Camera Elevation
49.0
Camera Azimuth
18.0
Camera FoV
20.0
Look At Point
-10.4
-4.7
0.0

D108B:Ex02 – WorldRT3A

- **Objectives**
  - Improve the terrain elevation and texturing of Ex01 using a sequence of terrain enhancement procedures.
- **Requirements and Implementation**
  - More realistic elevation, presenting a plausible mix of erosionary and alluvial terrain features such as carved riverbeds, eroded talus complexes ...etc.
  - Natural-fitting, richer, less repetitive texturing
- **Ex02 just scratches the surface of what Principia offers in the area of terrain generation. Its goal is to introduce basic process design concepts.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

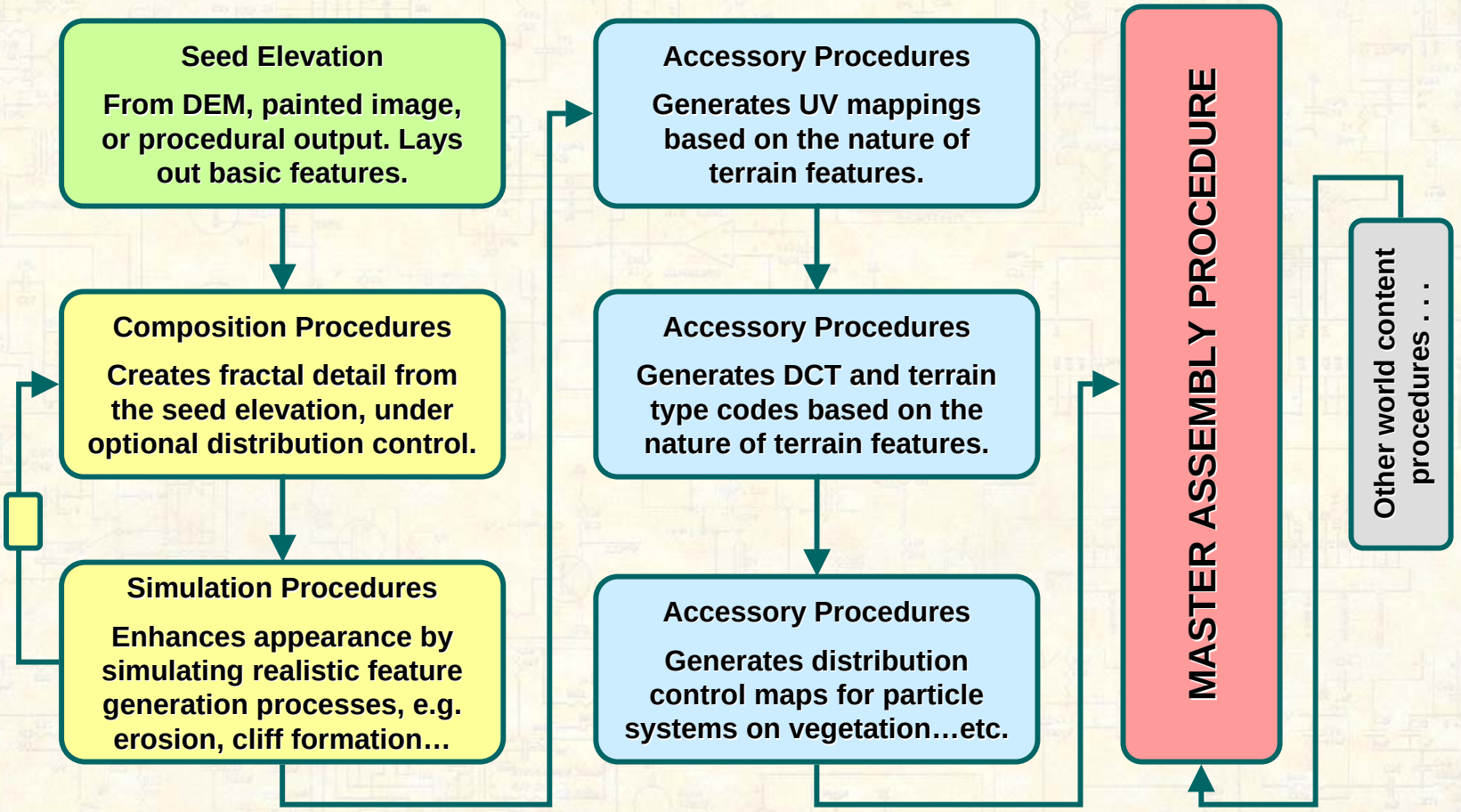
D108B:Ex02 – WorldRT3A

- **Terrain elevation generation methods**
  - Hand drawn maps
  - Procedural generation
  - Physical simulation
- **Practical terrain generation involves the sequenced combination of such methods**
  - Principia separates elevation generation from the world master assembly procedures.
  - Principia provides many families of procedural components for designing generation processes yielding visually rich and appropriate terrains.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

Practical Terrain Generation Process Design



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Principia terrain procedural components. Covered in Part III of the User Guide & Demos.**
  - Basic heightfields in the **FX_Hfield*K** family: Generate seed elevations for use by later procedural compositing.
  - Basic heightfields in the **FX_HField**** faimily: Fractal convolution and other detail enhancement compositing.
  - Texgen procedures in the **FX_TexGen_*DCT*** family: Particle system property and Render distribution control texture (DCT) generation and terrain type (GridT) mapping.
  - Specialized procedures in the **FX_TerGen_****** family: Advanced height field processing for a variety of appearance enhancement and physical feature simulation tasks.
  - Various procedures in the **FX_TexMod****** family: General purpose texture processing, such as filtering or composition.
- **Components exchange data encoded in texture form. This enables intermediate editing.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

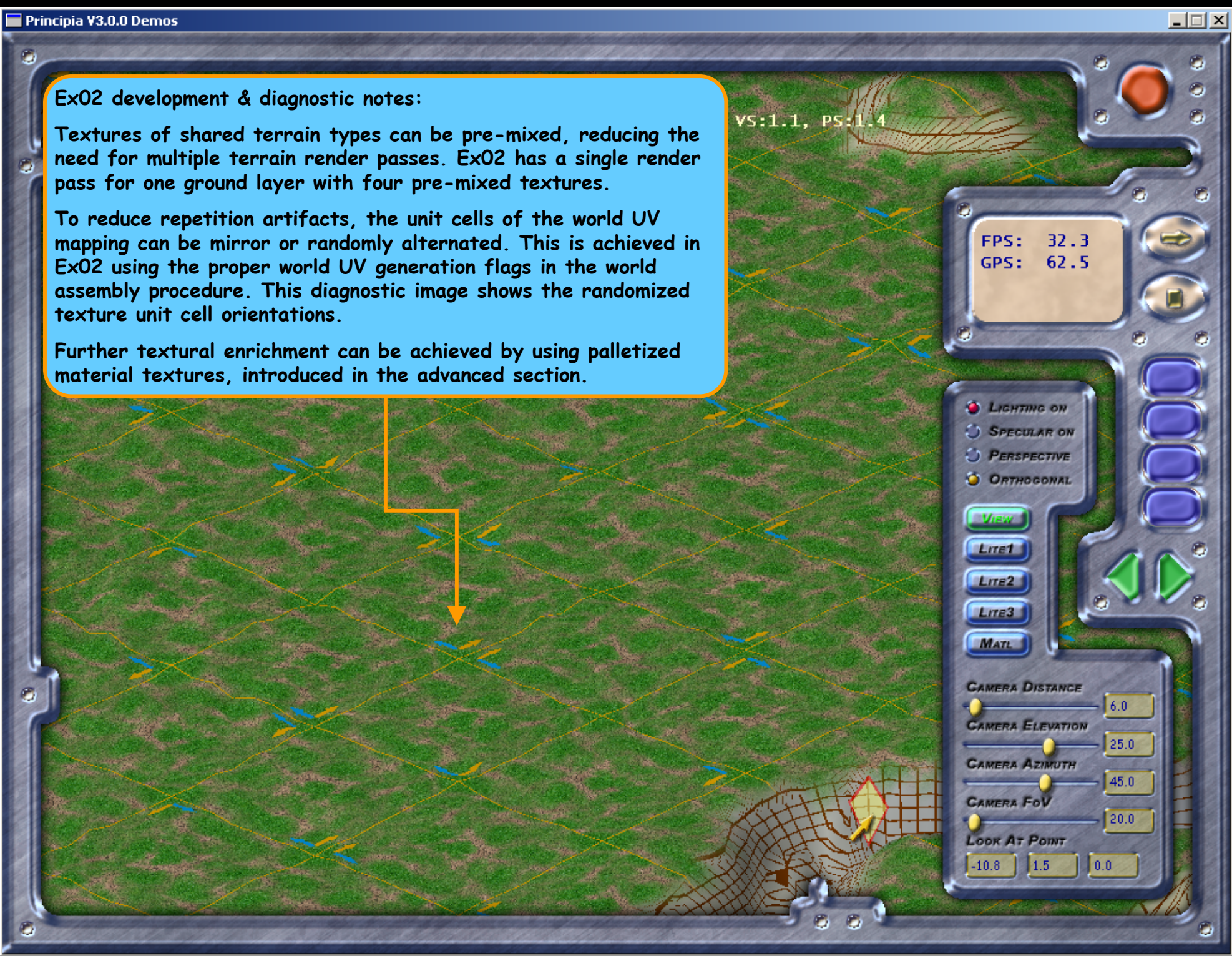
- **Basic heightfield primal generation options (Ch. 3.03):**
  - Elevation image map.
  - Deterministic space or frequency domain functions.
  - Stochastic space or frequency domain functions.
  - Subdivision processes.
  - Faulting processes.
  - Particle deposition processes.
  - Voronoi mapping and other topological processes.
- **Basic heightfield compositing options (Ch. 3.03):**
  - Filtering.
  - Generalized stochastic fractal convolution (with many specific instances in different procedures).
  - Mixing. This is especially useful for adding runtime detail to pre-generated terrains, or enhancing the latter.

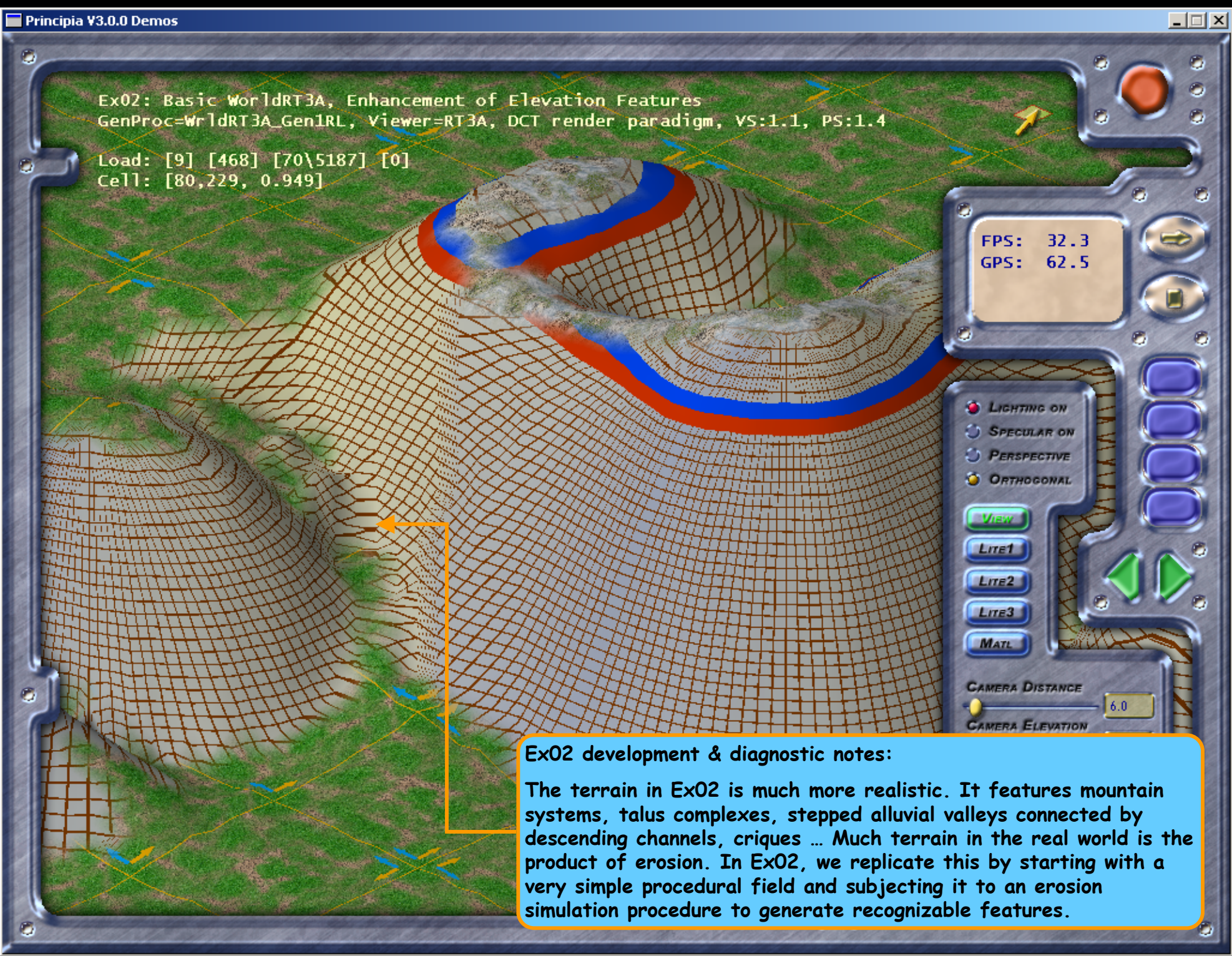
Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

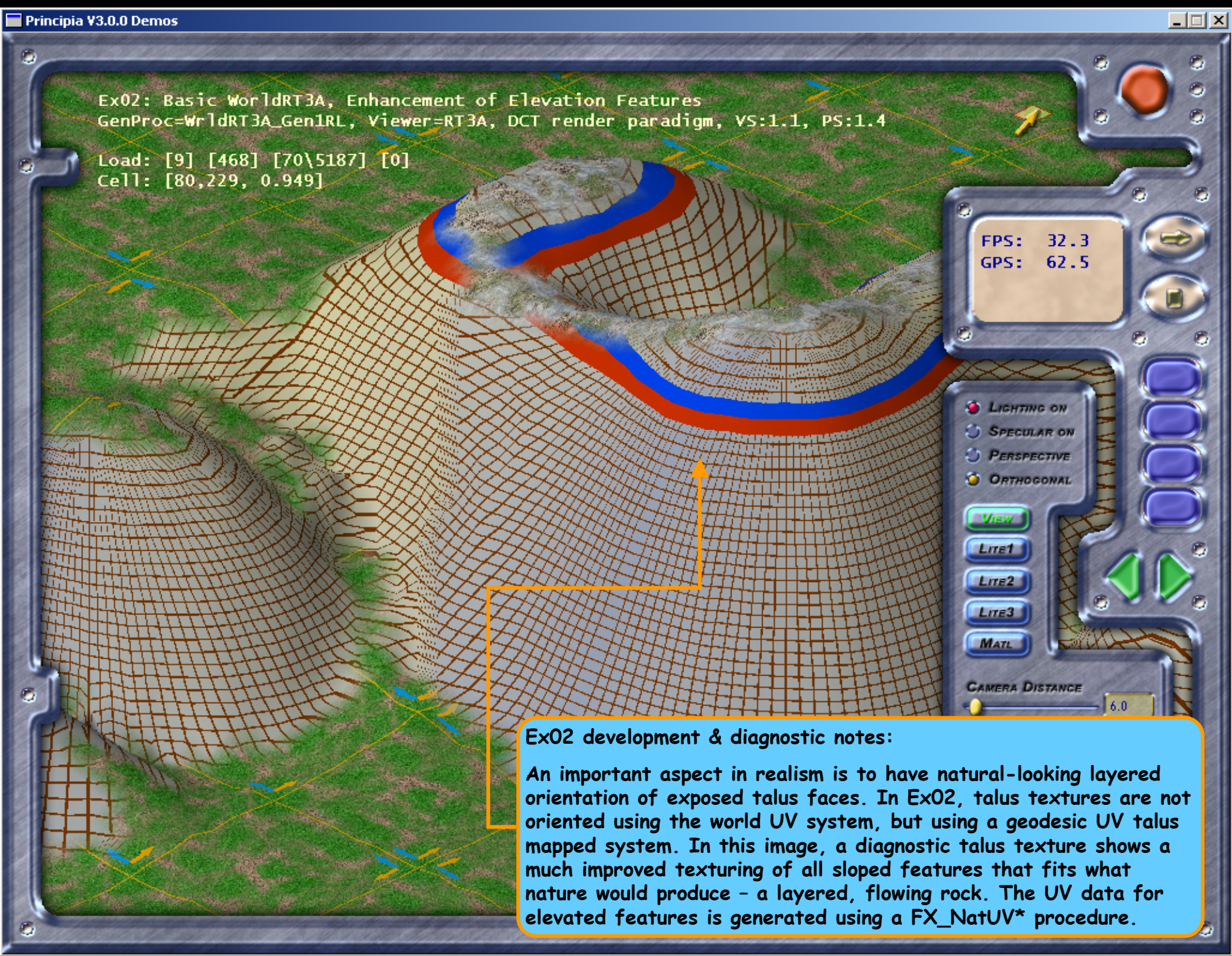
D108B:Ex02 – WorldRT3A

- **Process design is driven by desired appearance, in turn governed by many factors such as:**
  - Visual style
  - Viewing scale
  - Viewing flexibility
  - Layer rendering
- **Visual style:**
  - In games, a particular “surreal” look may be better than realism. Procedural design is ~ style!
- **Viewing scale:**
  - Far overhead views may require emphasizing features resulting from orogenic processes, while closer views may emphasize weathering and breakage detail.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex02 development & diagnostic notes:
Textures of shared terrain types can be pre-mixed, reducing the need for multiple terrain render passes. Ex02 has a single render pass for one ground layer with four pre-mixed textures.
To reduce repetition artifacts, the unit cells of the world UV mapping can be mirror or randomly alternated. This is achieved in Ex02 using the proper world UV generation flags in the world assembly procedure. This diagnostic image shows the randomized texture unit cell orientations.
Further textural enrichment can be achieved by using palletized material textures, introduced in the advanced section.
VS:1.1, PS:1.4
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
-10.8
1.5
0.0


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [9] [468] [70\5187] [0]
Cell: [80,229, 0.949]
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
Ex02 development & diagnostic notes:
The terrain in Ex02 is much more realistic. It features mountain systems, talus complexes, stepped alluvial valleys connected by descending channels, criques … Much terrain in the real world is the product of erosion. In Ex02, we replicate this by starting with a very simple procedural field and subjecting it to an erosion simulation procedure to generate recognizable features.


Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [9] [468] [70\5187] [0]
Cell: [80,229, 0.949]
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Ex02 development & diagnostic notes:
An important aspect in realism is to have natural-looking layered orientation of exposed talus faces. In Ex02, talus textures are not oriented using the world UV system, but using a geodesic UV talus mapped system. In this image, a diagnostic talus texture shows a much improved texturing of all sloped features that fits what nature would produce – a layered, flowing rock. The UV data for elevated features is generated using a FX_NatUV* procedure.


Principia V3.0.0 Demos
Ex02 development & diagnostic notes:
Thanks to the erosion procedures, our terrain has twisting river canyons that have a realistic, talus-fitting UV mapping, and spill into water basins in a natural manner. Virtually all aspects of the erosion processes and their visual outcome can be fine-tuned.
VS:1.1, PS:1.4
FPS: 32.3
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
-2.3
2.3
0.0

Principia ¥3.0.0 Demos

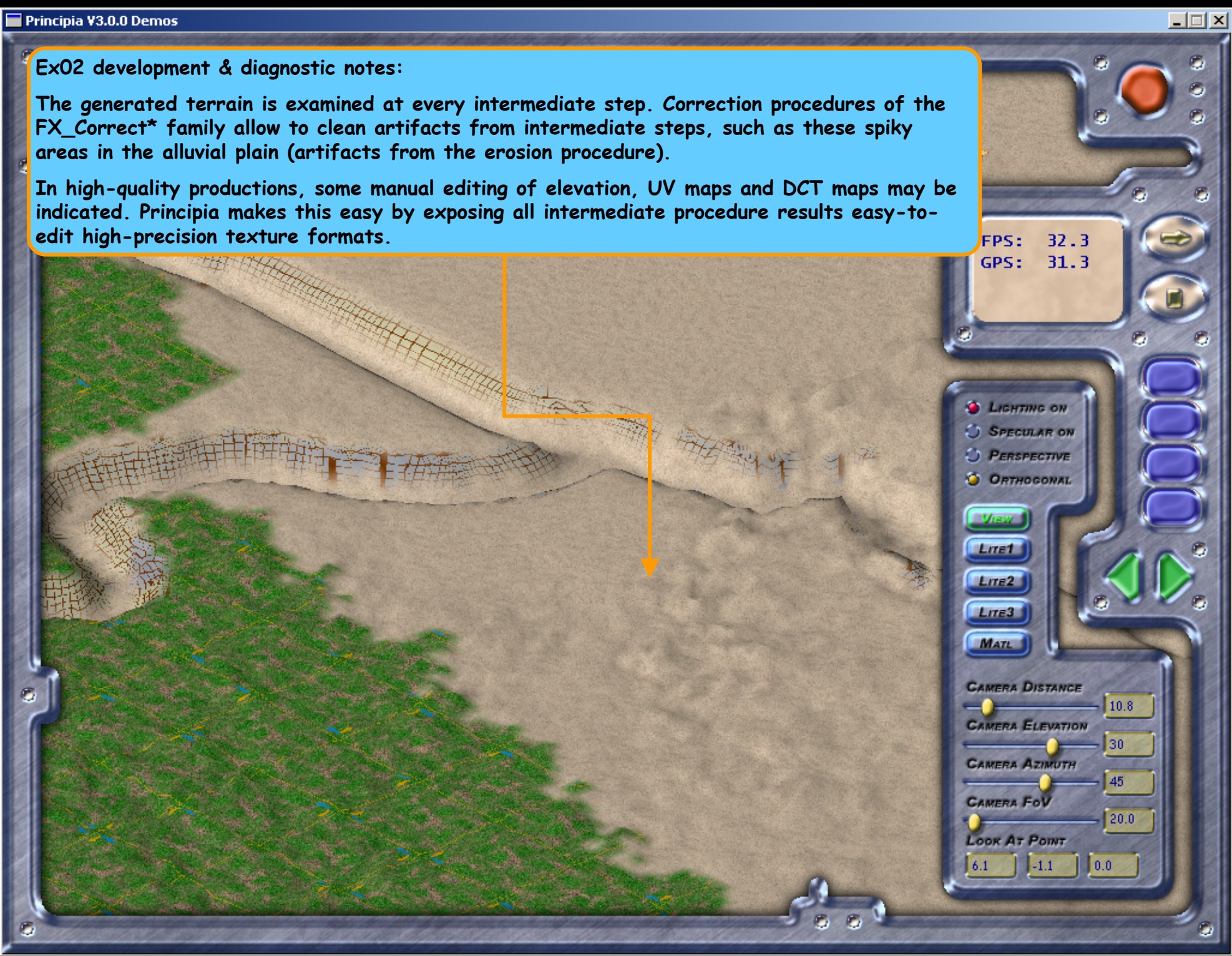
Ex02 development & diagnostic notes:

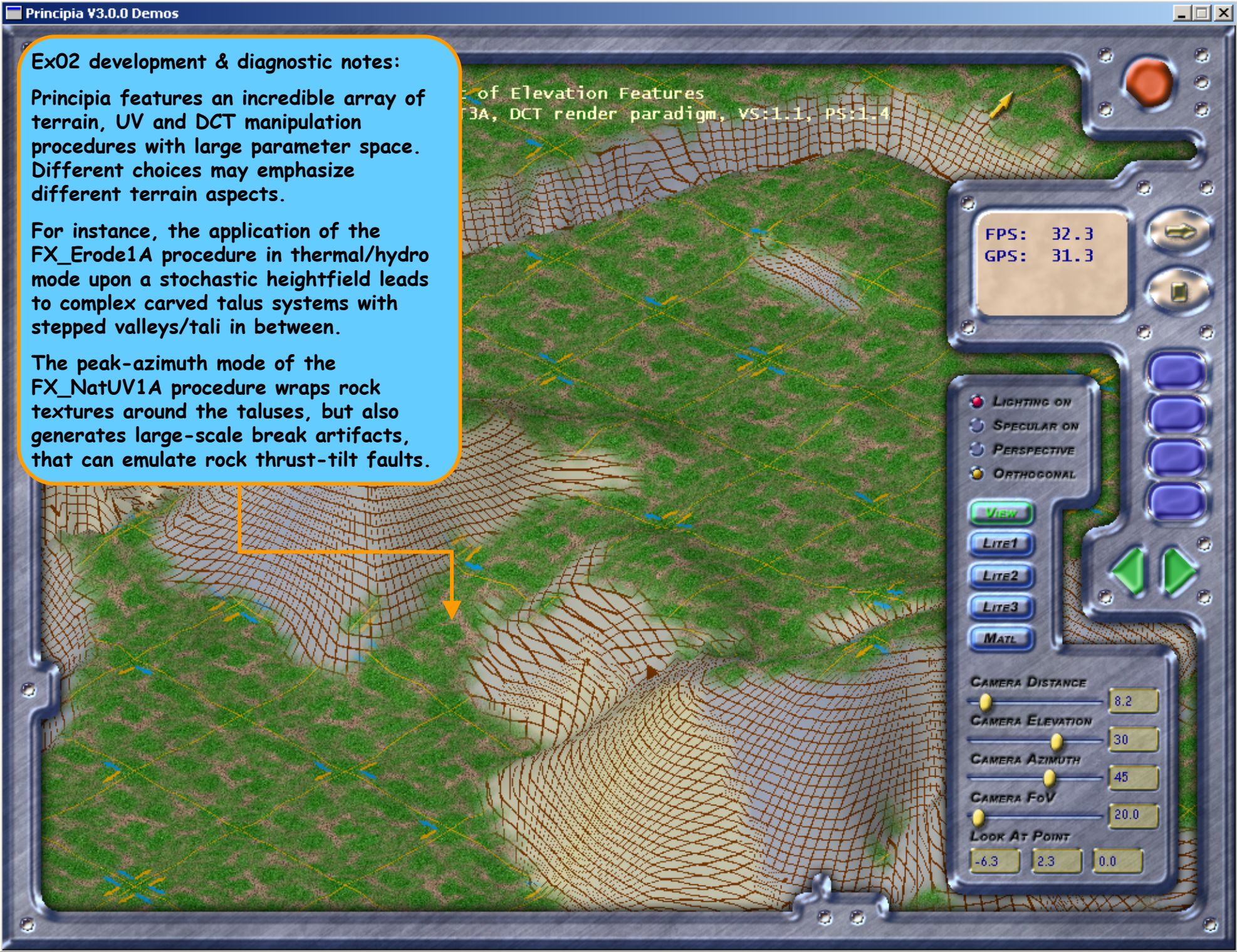
Riverbeds twist and branch naturally in terms of elevation and UV texturing. This is automatic thanks to the FX_Erode* and FX_NatUV* procedures used.

Note also that the proper texture types are allocated to plains, mountains, river beds and basin beds. This is also automatically accomplished by a feature-recognition DCT generation procedure that analyzes the terrain elevation and properly codes the terrain. The user can assign different nature materials to taluses, river beds …etc, easily and realistically.



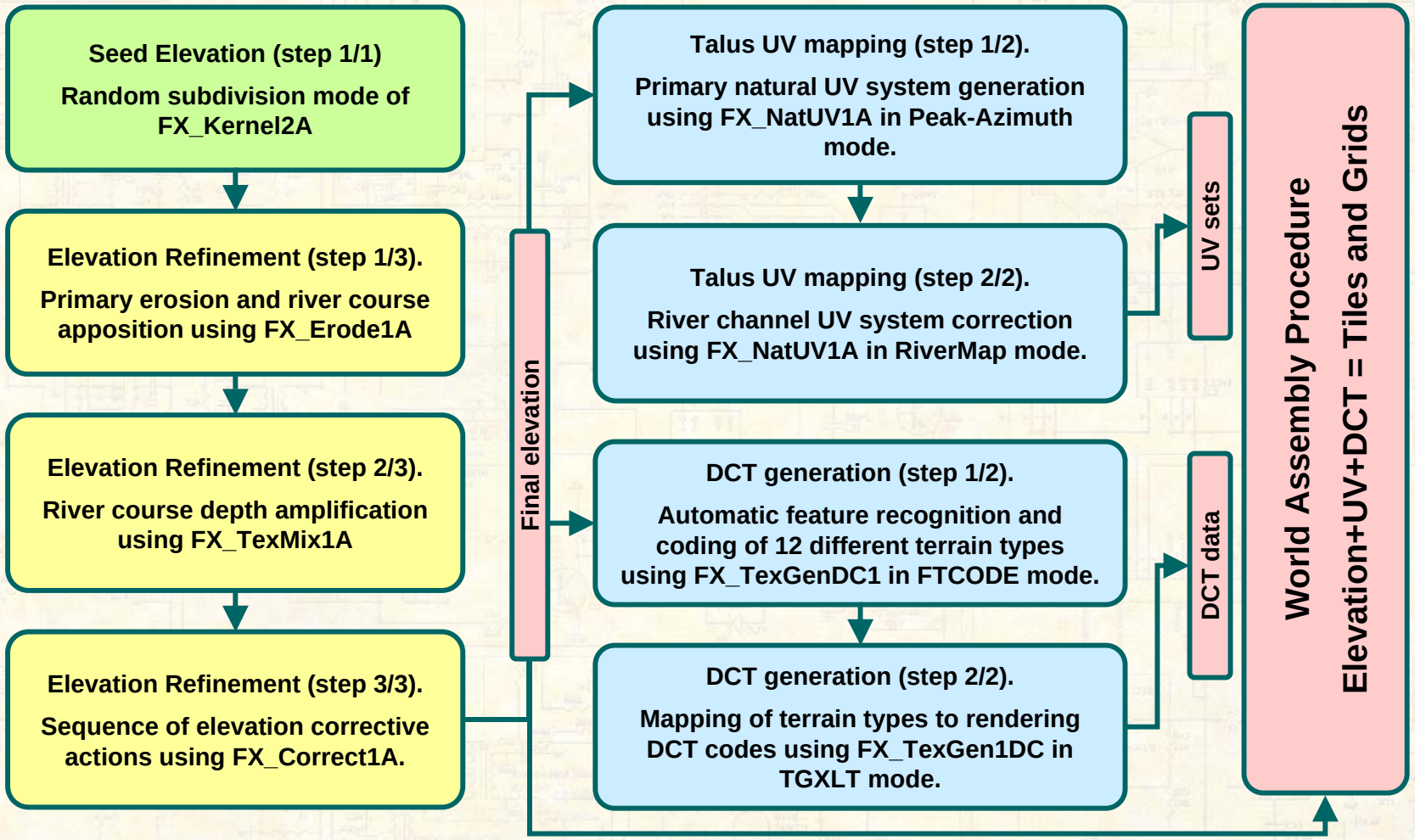

Principia V3.0.0 Demos
Ex02: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Gen1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [28] [1152] [596\5187] [0]
Cell: [254,184, 0.703]
Ex02 development & diagnostic notes:
Riverbeds carve canyons though mountainous areas naturally. Note the overall good UV mapping of taluses at junctions and other complex areas, as shown by the diagnostic textures.
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
23.3
Camera Elevation
57.0
Camera Azimuth
-117.0
Camera FoV
20.0
Look At Point
-3.4
-4.2
0.0


Principia V3.0.0 Demos
Ex02 development & diagnostic notes:
The generated terrain is examined at every intermediate step. Correction procedures of the FX_Correct* family allow to clean artifacts from intermediate steps, such as these spiky areas in the alluvial plain (artifacts from the erosion procedure).
In high-quality productions, some manual editing of elevation, UV maps and DCT maps may be indicated. Principia makes this easy by exposing all intermediate procedure results easy-to-edit high-precision texture formats.
FPS: 32.3
GPS: 31.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
10.8
Camera Elevation
30
Camera Azimuth
45
Camera FoV
20.0
Look At Point
6.1
-1.1
0.0


Principia V3.0.0 Demos
Ex02 development & diagnostic notes:
Principia features an incredible array of terrain, UV and DCT manipulation procedures with large parameter space. Different choices may emphasize different terrain aspects.
For instance, the application of the FX_Erode1A procedure in thermal/hydro mode upon a stochastic heightfield leads to complex carved talus systems with stepped valleys/tali in between.
The peak-azimuth mode of the FX_NatUV1A procedure wraps rock textures around the taluses, but also generates large-scale break artifacts, that can emulate rock thrust-tilt faults.
of Elevation Features
T3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 32.3
GPS: 31.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
8.2
Camera Elevation
30
Camera Azimuth
45
Camera FoV
20.0
Look At Point
-6.3
2.3
0.0

D108B:Ex02 – WorldRT3A

Ex02 terrain generation procedure steps



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

Seed elevation, step 1/1:

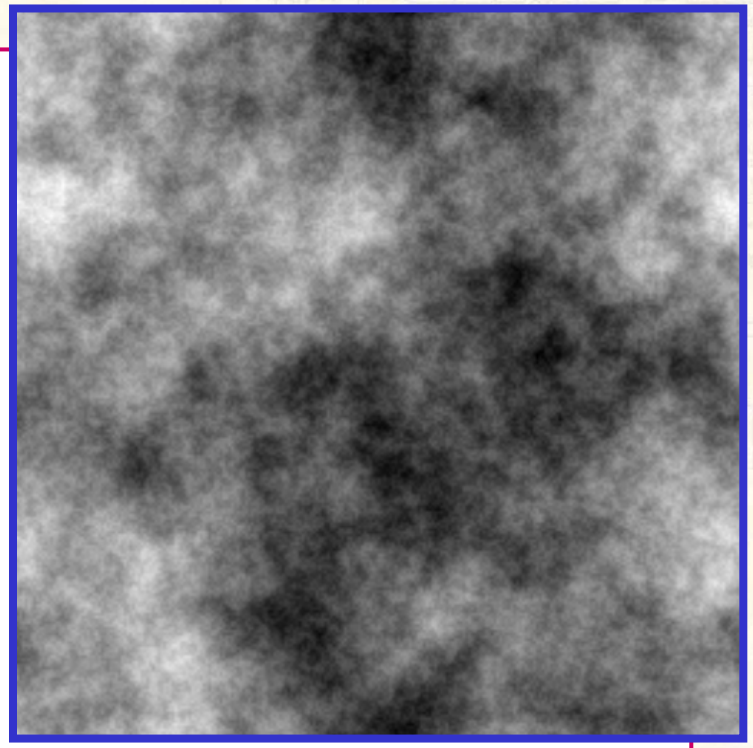
```
##define (K_ELEV_STEP1) as <PROCDAT_KERNEL_2A>
   #tag Kernel_Type            =  (  KRN_RDIVIDE )
   #tag Adjust_Function        =  (         NULL )
   #tag DistortXF              =  (         NULL )
   #tag DistortYF              =  (         NULL )
   #tag Kernel_Function        =  "Arg(u,v); n=RndU(-1.0,1.0); Ret(n);"
   #tag Lattice_Seed           =  (            1 )
   #tag Lattice_Order          =  (          257 )
   #tag Lattice_Inter          =  (   INTER_LERP )
   #tag Lattice_Smooth         =  (          0.5 )

##define (F_ELEV_STEP1) as <FUNCTION_S>
   #tag Operation =  "Arg(u,v); v=Proc(u,v); Ret(v);"
   #tag Proc      =  (  K_ELEV_STEP1 )

##define (PROC_HFGEN_EX02) as <PROCTEX_GENCLR_1A>
   #tag Method    =  (  PROC_TXGEN_RGBA )
   #tag FcnRedHue =  (     F_ELEV_STEP1 )
   #tag RedAll    =  (              YES )

##define (S_ELEV_STEP1) as <SURFACE>
   #tag Width     =  (              505 )
   #tag Heigt     =  (              505 )
   #tag Procedure =  (  PROC_HFGEN_EX02 )
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##save (S_ELEV_STEP1) in ("Files_Media/Ex02_Map_Elevation_Step1.png")
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

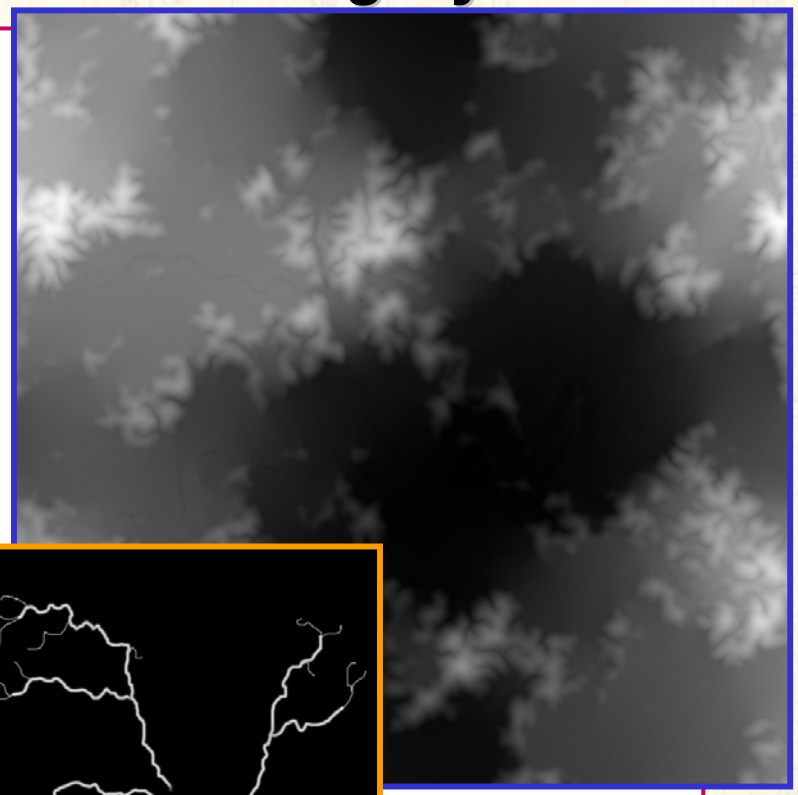
D108B:Ex02 – WorldRT3A

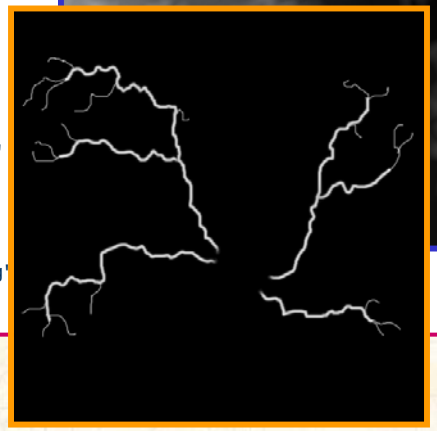
- **Erosionary refinement, step 1/3. Note the input river course control map encoded in grayscale.**

```
##define (S_RIVR_STEP2) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_RiverEroders.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (PROC_ERODE_STEP2) as <PROCWRL_ERODE_1A>
   #tag ex.ElevMap       =  (        S_ELEV_STEP2 )
   #tag in.ElevMap       =  (        S_ELEV_STEP1 )
   #tag in.RainMap       =  (        S_ELEV_STEP1 )
   #tag in.TempMap       =  (                   - )
   #tag in.SoftMap       =  (                   - )
   #tag in.RivrMap       =  (        S_RIVR_STEP2 )
   #tag in.ZNorm         =  (                  NO )
   #tag ex.ZNorm         =  (                 YES )
   #tag RandomSeed       =  (             4325456 )
   #tag Iter.Total       =  (                2000 )
   #tag Freq.Therm       =  (                  80 )
   #tag Freq.Hydro       =  (                   1 )
   #tag Therm.MinSlope   =  (                30.0 )
   #tag Therm.TimeRate   =  "Arg(t); r=Gt(t,0.90,0.10,0.22); Ret(r);"
   #tag Hydro.RainTime   =  "Arg(t); r=Gt(t,0.90,0.00,1.00); Ret(r);"
   #tag Hydro.RainProb   =  (                  10 )
   #tag Hydro.RainFreq   =  (                   5 )
   #tag Hydro.RainScale  =  (              0.0010 )
   #tag Hydro.RainSlope  =  (              0.0000 )
   #tag Hydro.Deposit    =  (                0.03 )
   #tag Hydro.Washout    =  (                0.30 )
   #tag Hydro.Sedicap    =  (                5.00 )
   #tag Hydro.EvapTime   =  "Arg(t); r=Set(1.00); Ret(r);"
   #tag Hydro.EvapRate   =  (               0.005 )
   #tag River.FallTime   =  "Arg(t); r=Gt(t,0.90,0.0,1.0); Ret(r);"
   #tag River.FallProb   =  (                 100 )
   #tag River.FallFreq   =  (                  10 )
   #tag River.FallScale  =  (              0.0005 )

##run (PROC_ERODE_STEP2)
##save (S_ELEV_STEP2) in ("Files_Media/Ex02_Map_Elevation_Step2.png"
```





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Erosionary refinement, step 2/3. That just deepens the river channels a bit by using the river course control map from step 1/3.**

```
##define (PROC_TEXGEN_STEP3) as <PROCTEX_OPSMIX_1A>
   #tag Image1    =  (     S_ELEV_STEP2 )
   #tag Image2    =  (     S_RIVR_STEP2 )
   #tag OpR       =  "arg(a,b,c,d,x,y,z,u); p=Mul(x,-0.06); q=add(a,p); q=clamp(q,0.0,1.0); ret(q);"
   #tag OpG       =  "arg(a,b,c,d,x,y,z,u); p=Mul(y,-0.06); q=add(b,p); q=clamp(q,0.0,1.0); ret(q);"
   #tag OpB       =  "arg(a,b,c,d,x,y,z,u); p=Mul(z,-0.06); q=add(c,p); q=clamp(q,0.0,1.0); ret(q);"
   #tag OpA       =  (                - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Terrain corrective sequence (Step3)**
  - Step3A: Artifact peak fix using 3x3 topological filter (FX_Correct1A mode: Peak3x3)
  - Step3B: Shoreline gap closure using topological connectivity filter (FX_Correct1A mode: ShoreNoise)
  - Step3C: Ocean-to-Land elevation shifts and local averaging (FX_Correct1A mode: OceanLevel)
  - Step3D: Topologic averaging of large plains to fix high frequency artifacts from erosion, and topologic averaging of shoreline to fix blocky beach contour (FX_Correct1A modes: PlainFilter/ShoreFilter).
- **Principia offers a rich set of topologic operations, performing field algebra with consideration for the underlying operant field topology.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

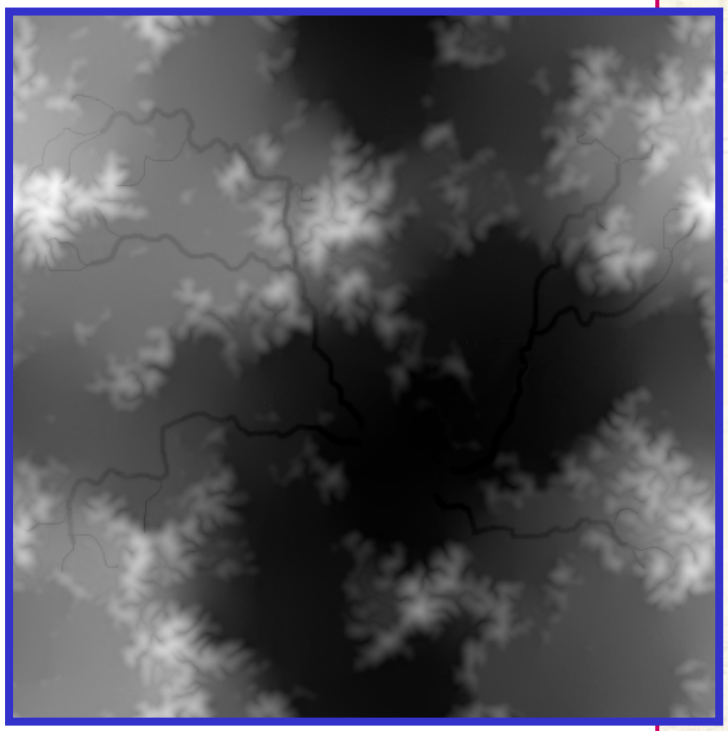
- **Artifact correction, step 3A. This eliminates topological noise, especially from water beds. The final elevation is encapsulated in a function.**

```
##define (S_RIVR_STEP4) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_RiverEroders.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (S_ELEV_STEP4A) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_Elevation_Step3.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (PROC_CORRECT_STEP4A) as <PROCWRL_CORRECT_1A>
   #tag ex.ElevMap         =  (       S_ELEV_STEP4A )
   #tag ex.ElevFcn         =  (               NONE )
   #tag ex.CodeMap         =  (               NONE )
   #tag in.ElevMap         =  (       S_ELEV_STEP4A )
   #tag in.ElevFcn         =  (               NONE )
   #tag in.GradFcn         =  (               NONE )
   #tag in.AuthMap         =  (               NONE )
   #tag in.AuthLevel       =  (               ZERO )
   #tag ex.CNorm           =  (                YES )
   #tag ex.ZNorm           =  (                 NO )
   #tag in.ZNorm           =  (                 NO )
   #tag Fix.Peak3x3Size    =  (               0.50 )
   #tag Fix.Peak3x3Bias    =  (               0.00 )

##run (PROC_CORRECT_STEP4A)
##save (S_ELEV_STEP4A) in ("Files_Media/Ex02_Map_Elevation_Step4A.png")
##delete (S_ELEV_STEP4A)
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Artifact correction, step 3B, to eliminate common hole artifacts from stochastic procedurals.**

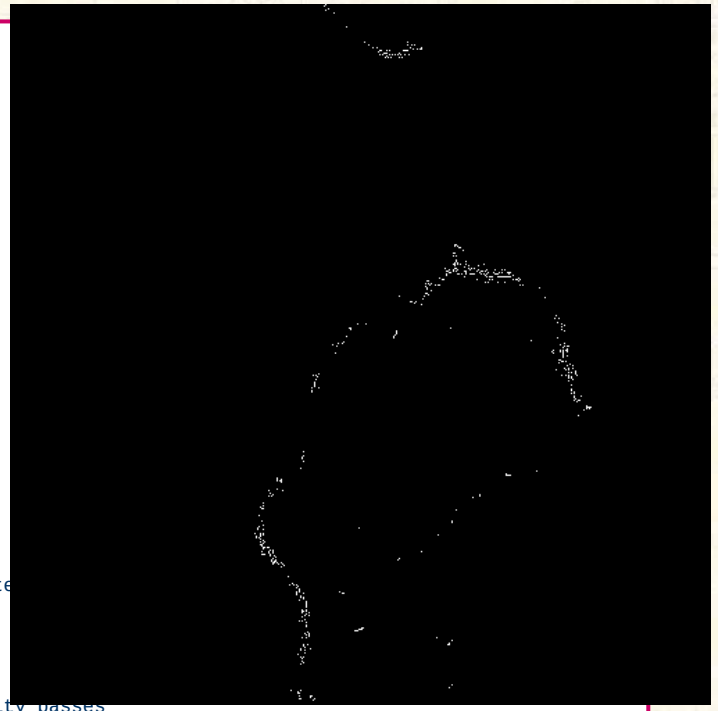
```
##define (S_ELEV_STEP4B) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_Elevation_Step4A.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (F_ELEV_STEP4B_IN) as <FUNCTION_S>
   #tag Operation =  "Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag ZMap      =  ( S_ELEV_STEP4B )
   #tag ZMin      =  (          0.000 )
   #tag ZMax      =  (          2.200 )

##define (F_ELEV_STEP4B_EX) as <FUNCTION_S>
   #tag Operation =  "Arg(x,y,z); z=Div(z,2.200); Ret(z);"

##define (PROC_CORRECT_STEP4B) as <PROCWRL_CORRECT_1A>
   #tag ex.ElevMap         =  (       S_ELEV_STEP4B )
   #tag ex.ElevFcn         =  (    F_ELEV_STEP4B_EX )
   #tag ex.CodeMap         =  (       S_DIAG_STEP4B )
   #tag in.ElevMap         =  (       S_ELEV_STEP4B )
   #tag in.ElevFcn         =  (    F_ELEV_STEP4B_IN )
   #tag in.GradFcn         =  (    F_ELEV_STEP4B_IN )
   #tag in.AuthMap         =  (                NONE )
   #tag in.AuthLevel       =  (                ZERO )
   #tag ex.CNorm           =  (                 YES )
   #tag ex.ZNorm           =  (                  NO )
   #tag in.ZNorm           =  (                  NO )
   #tag Com.Scale          =  (     0.05 ,     0.05 ) ' Discretization relate
   #tag Com.FtrRiverMap    =  (        S_RIVR_STEP4 )
   #tag Com.FtrOceanLevel  =  (               0.150 )
   #tag Com.FtrRiverLevel  =  (               0.001 )
   #tag Com.FtrSlopeMinG   =  (                25.0 )
   #tag Com.FtrSlopeSize   =  (                   8 )
   #tag Fix_ShorePasses    =  (                   6 ) 'Shore fix, connectivity passes
   #tag Fix_ShoreLconn     =  (                   5 ) 'Shore fix, land point max isolatedness
   #tag Fix_ShoreLiter     =  (                   8 ) 'Shore fix, land iterator
   #tag Fix_ShoreOconn     =  (                   5 ) 'Shore fix, water point max isolatedness
   #tag Fix_ShoreOiter     =  (                   8 ) 'Shore fix, water iterator

##run (PROC_CORRECT_STEP4B)
##save (S_ELEV_STEP4B) in ("Files_Media/Ex02_Map_Elevation_Step4B.png")
##save (S_DIAG_STEP4B) in ("Files_Media/Ex02_Map_Elevation_Correct4B.png")
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Artifact correction, step 3B, before and after.**





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Ocean-to-land shift and averaging, step 3C. Note that scalar procedures such as here require to convert the [01] elevation map to full-range fcn:**

```
##define (S_ELEV_STEP4C) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_Elevation_Step4B.png"
   #tag Definer   =  (        DEF_16BIT )

##define (F_ELEV_STEP4C_IN) as <FUNCTION_S>
   #tag Operation =  "Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag ZMap      =  ( S_ELEV_STEP4C )
   #tag ZMin      =  (         0.000 )
   #tag ZMax      =  (         2.200 )

##define (F_ELEV_STEP4C_EX) as <FUNCTION_S>
   #tag Operation =  "Arg(x,y,z); z=Div(z,2.200); Ret(z);"
```

- **Step3C (ocean-to-land shift) purpose**
  - **Deepen ocean in controlled manner based on distance to shore to achieve a better transparency distribution.**
  - **Clean-up underwater talus artifacts from rivers, but leave descending land taluses un-modified.**
  - **Elevate all land area to achieve better beach definition for later wave simulation in the shader.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

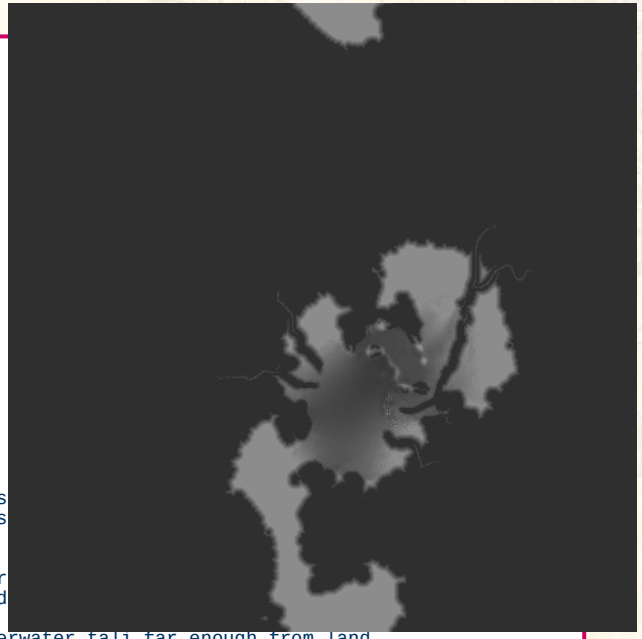
Ocean-to-land shift and averaging, step 3C.

```
##define (PROC_CORRECT_STEP4C) as <PROCWRL_CORRECT_1A>

   #tag ex.ElevMap         =  (       S_ELEV_STEP4C )
   #tag ex.ElevFcn         =  (    F_ELEV_STEP4C_EX )
   #tag ex.CodeMap         =  (      S_DIAG_STEP4C )
   #tag in.ElevMap         =  (       S_ELEV_STEP4C )
   #tag in.ElevFcn         =  (    F_ELEV_STEP4C_IN )
   #tag in.GradFcn         =  (    F_ELEV_STEP4C_IN )
   #tag in.AuthMap         =  (                NONE )
   #tag in.AuthLevel       =  (                ZERO )
   #tag ex.CNorm           =  (                 YES )
   #tag ex.ZNorm           =  (                  NO )
   #tag in.ZNorm           =  (                  NO )
   #tag Com.Scale          =  (     0.05 ,     0.05 ) ' Discretization related
   #tag Com.FtrRiverMap    =  (        S_RIVR_STEP4 )
   #tag Com.FtrOceanLevel  =  (               0.150 )
   #tag Com.FtrRiverLevel  =  (               0.001 )
   #tag Com.FtrSlopeMinG   =  (                25.0 )
   #tag Com.FtrSlopeSize   =  (                   8 )

   #tag Fix_OceanDZdmax    =  (                  10 ) 'Maximum process unit cell s
   #tag Fix_OceanDZdist    =  (                   5 ) 'Normalizing distance from s
   #tag Fix_OceanDZdfcn    =  "Arg(x,y,d); d=Min(d,1.0); z=Mul(d,-0.08); Ret(z);"
   #tag Fix_OceanDZaext    =  (                   2 ) 'Averaging unit cell span
   #tag Fix_OceanDZaref    =  (                   5 ) 'Blending distance from shor
   #tag Fix_OceanDZafcn    =  "Arg(x,y,d); d=Min(d,1.0); f=Set(d); Ret(f);" 'Blend
   #tag Fix_OceanDZapas    =  (                   2 ) 'Topologic averaging passes
   #tag Fix_OceanDZtmod    =  (                   4 ) 'Enable modification of underwater tali far enough from land
   #tag Fix_LandsDZdmax    =  (                  15 ) 'Maximum process unit cell span for land elevation fixes
   #tag Fix_LandsDZdist    =  (                  12 ) 'Normalizing distance from shore for adder landfix function
   #tag Fix_LandsDZdfcn    =  "Arg(x,y,d); d=Min(d,1.0); z=Mul(d,0.05); Ret(z);" 'Adder landfix function

##run (PROC_CORRECT_STEP4C)
##save (S_ELEV_STEP4C) in ("Files_Media/Ex02_Map_Elevation_Step4C.png")
##save (S_DIAG_STEP4C) in ("Files_Media/Ex02_Map_Elevation_Correct4C.png")
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Ocean-to-land shift and averaging, step 3C. Note progressive smoothing of underwater canyons.**



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Noise filtering, step 3D. Removes noise from plains and corrects shore beach line blockiness.**

```
##define (PROC_CORRECT_STEP4D) as <PROCWRL_CORRECT_1A>

   #tag ex.ElevMap         =  (       S_ELEV_STEP4D )
   #tag ex.ElevFcn         =  (    F_ELEV_STEP4D_EX )
   #tag ex.CodeMap         =  (       S_DIAG_STEP4D )
   #tag in.ElevMap         =  (       S_ELEV_STEP4D )
   #tag in.ElevFcn         =  (    F_ELEV_STEP4D_IN )
   #tag in.GradFcn         =  (    F_ELEV_STEP4D_IN )
   #tag in.AuthMap         =  (                NONE )
   #tag in.AuthLevel       =  (                ZERO )
   #tag ex.ZNorm           =  (                  NO )
   #tag ex.CNorm           =  (                 YES )
   #tag in.ZNorm           =  (                  NO )

   #tag Com.Scale          =  (      0.05 ,    0.05 ) ' Discretization related
   #tag Com.FtrRiverMap    =  (        S_RIVR_STEP4 )
   #tag Com.FtrOceanLevel  =  (               0.150 )
   #tag Com.FtrRiverLevel  =  (               0.001 )
   #tag Com.FtrSlopeMinG   =  (                25.0 )
   #tag Com.FtrSlopeSize   =  (                   8 )
   #tag Exe.DevMod         =  (                  NO )

   #tag Fix_PfNbPass       =  (                   1 )
   #tag Fix_PfMinSize      =  (                 100 )
   #tag Fix_PfAperture     =  (                   2 )
   #tag Fix_PfEdgedist     =  (                   5 )
   #tag Fix_PfStrength     =  (                0.50 )

   #tag Fix_SfShoreExt     =  (                   5 )
   #tag Fix_SfNbPass       =  (                   4 )
   #tag Fix_SfMinSize      =  (                  32 )
   #tag Fix_SfAperture     =  (                   2 )
   #tag Fix_SfEdgedist     =  (                   3 )
   #tag Fix_SfStrength     =  (                0.50 )

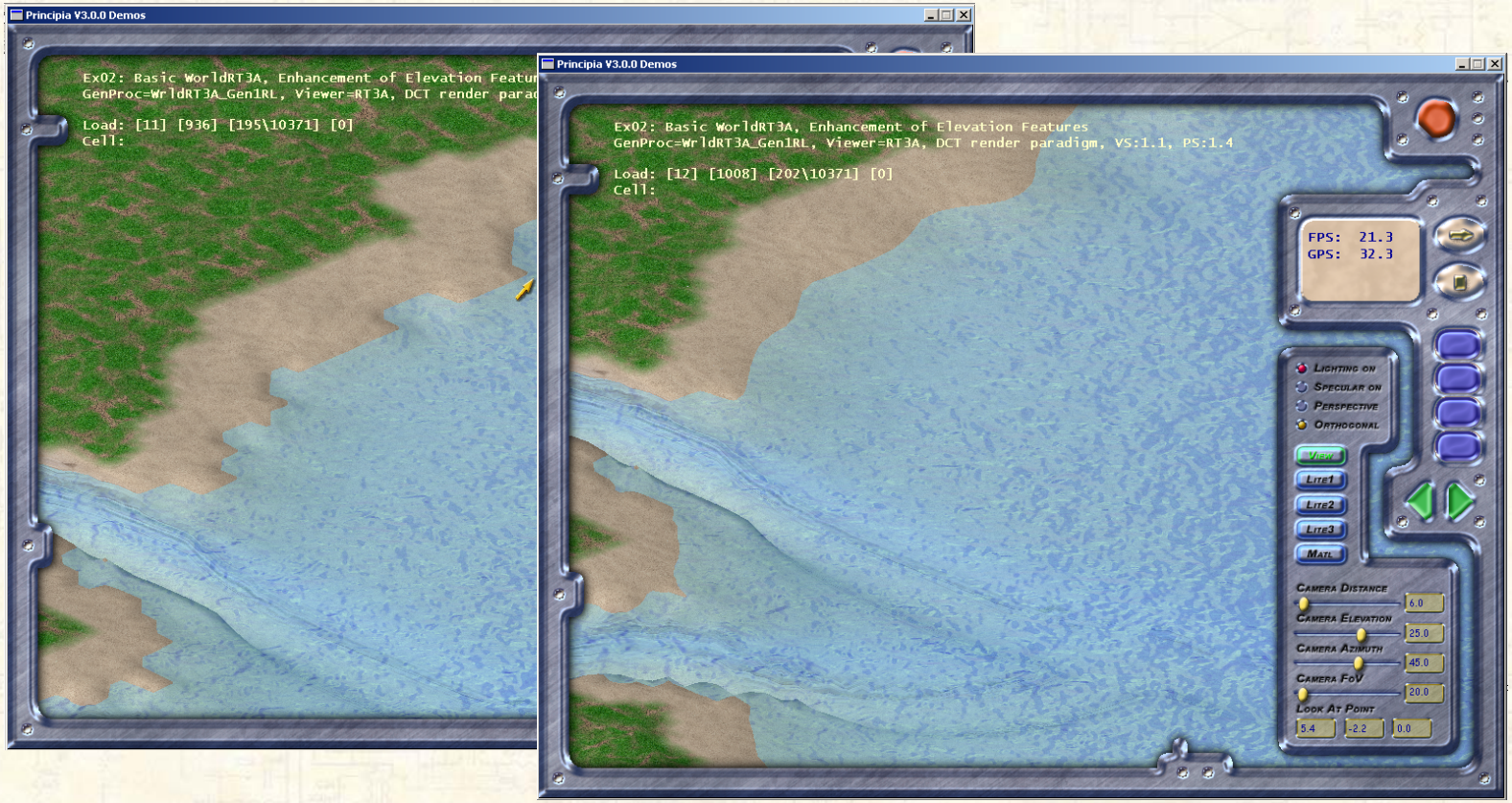

##run (PROC_CORRECT_STEP4D)

##save (S_ELEV_STEP4D) in ("Files_Media/Ex02_Map_Elevation_Final.png")
##save (S_DIAG_STEP4D) in ("Files_Media/Ex02_Map_Elevation_Correct4D.png")
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Noise filtering, step 3D. Natural beaches have smooth contours. This is achieved by a topologic average that leaves taluses and rivers alone.**



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Noise filtering, step 3D, plain smoothing. Topologic averaging should follow all other steps.**



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

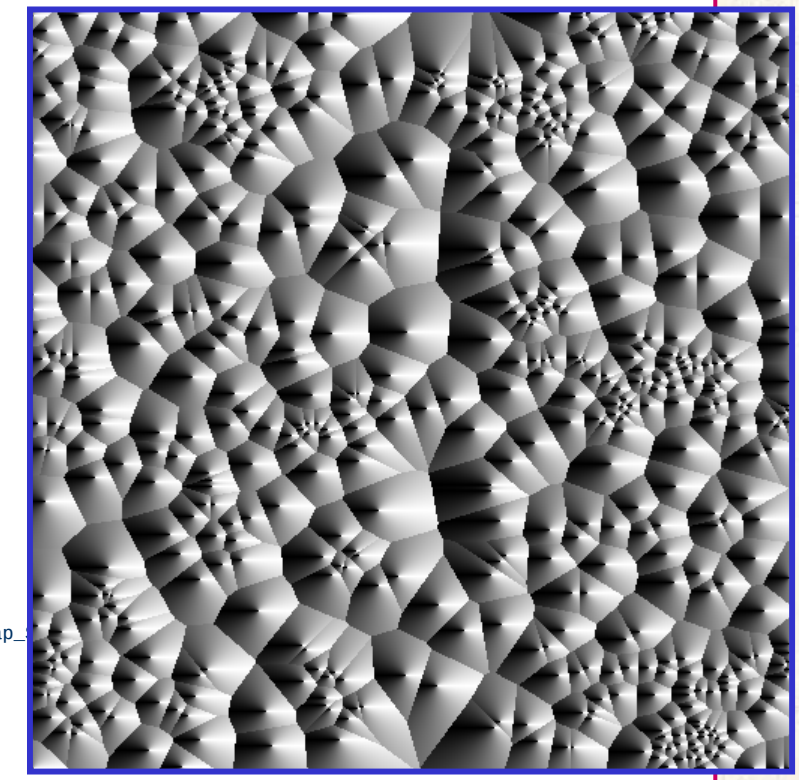
- **Talus natural UV mapping, step 1/2. The final elevation is used as input.**

```
##define (S_WRAPU) as <SURFACE>
   #tag Definer   =  (         DEF_16BIT )
   #tag Width     =  (               505 ) ' Discretization related
   #tag Heigt     =  (               505 ) ' Discretization related
   #tag LoadNow   =  (                 1 )

##define (S_WRAPV) as <SURFACE>
   #tag Definer   =  (         DEF_16BIT )
   #tag Width     =  (               505 ) ' Discretization related
   #tag Heigt     =  (               505 ) ' Discretization related
   #tag LoadNow   =  (                 1 )

##define (PROC_UVMAP_STEP1) as <PROCWRL_NATUV_1A>
   #tag Method    =  (  PROC_UVGEN_AZIPEAK )
   #tag inZFcn    =  (               F_ELEV )
   #tag inZMap    =  (               S_ELEV )
   #tag inAMap    =  (                 NONE )
   #tag exUMap    =  (              S_WRAPU )
   #tag exVMap    =  (              S_WRAPV )
   #tag Scale     =  (     0.05 ,      0.05 ) ' Discretization related
   #tag inZNorm   =  (                   NO )
   #tag Zmin      =  (                 0.00 ) ' Discretization related
   #tag Zmax      =  (                 2.20 ) ' Discretization related
   #tag UXform    =  "Arg(z); u=Clamp(z,0.0,1.0); Ret(u);"
   #tag VSpan     =  (                180.0 )
   #tag PeakDelta =  (              0.00300 )
   #tag PeakSpanI =  (                    2 )
   #tag PeakSpanJ =  (                    2 )
   #tag PeakSpanC =  (                    1 )

##run (PROC_UVMAP_STEP1) with target ("Files_Media/Ex02_Map_NaturalWrap_
##save (S_WRAPU) in ("Files_Media/Ex02_Map_NaturalWrap_Step1_U.png")
##save (S_WRAPU) in ("Files_Media/Ex02_Map_NaturalWrap_Step2_U.png")
##save (S_WRAPV) in ("Files_Media/Ex02_Map_NaturalWrap_Step1_V.png")
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

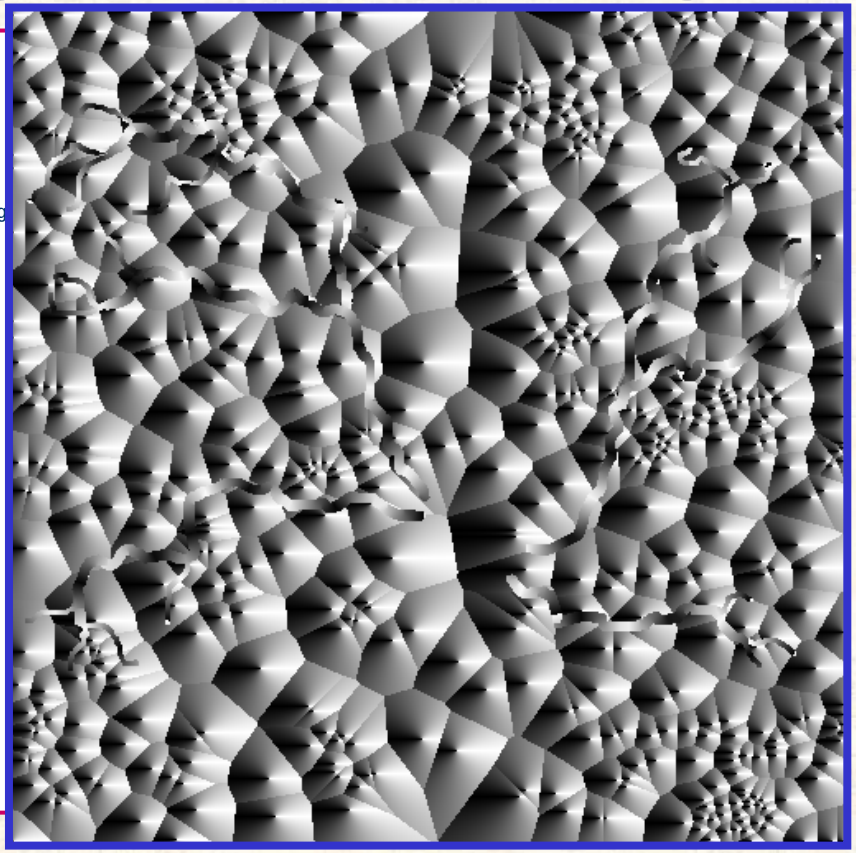
D108B:Ex02 – WorldRT3A

- **Talus natural UV mapping, step 2/2. This updates only the V-distribution from 1/2 so as to correctly map rock folding along the carved river canyons.**

```
##define (S_RIVRUVMAP_LINE) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_RiverTracer.png"
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )

##define (S_RIVRUVMAP_AUTH) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_RiverAuthorizer.png
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )

##define (PROC_UVMAP_STEP2) as <PROCWRL_NATUV_1A>
   #tag Method    =  (   PROC_UVGEN_RIVERMAP )
   #tag inZFcn    =  (                 NONE )
   #tag inZMap    =  (      S_RIVRUVMAP_LINE )
   #tag inAMap    =  (      S_RIVRUVMAP_AUTH )
   #tag exUMap    =  (               S_WRAPU )
   #tag exVMap    =  (               S_WRAPV )
   #tag inZNorm   =  (                  NO )
   #tag Zmin      =  (                0.00 )
   #tag Zmax      =  (                1.00 )
   #tag UXform    =  "Arg(z); u=Clamp(z,0.0,1.0); Ret(u);"
   #tag VSpan     =  (                30.0 )
   #tag VBand     =  (                10.0 )
   #tag SplineMap =  (                 YES )
   #tag SplineDiv =  (                  10 )
   #tag SplineItr =  (                   3 )
   #tag MinSize   =  (                  10 )
   #tag MinAper   =  (                   5 )
   #tag MaxAper   =  (                   5 )
   #tag MaxJoin   =  (                   7 )
   #tag RiverEdge =  (                0.03 )
   #tag RiverHead =  (          0xFFFF0000 )
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

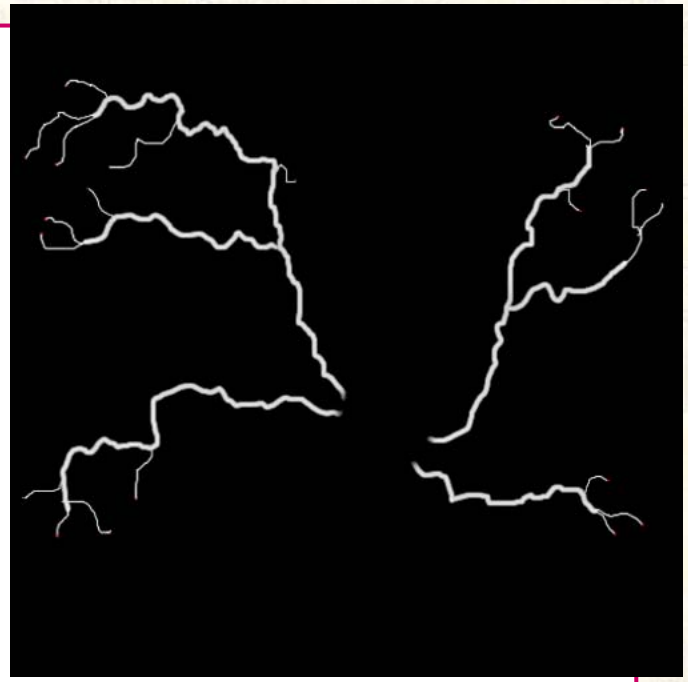
D108B:Ex02 – WorldRT3A

- **Talus natural UV mapping, step 2/2. Note the need for carefully managed river tracing and use of authorization control textures.**

```
##define (S_RIVRUVMAP_LINE) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_RiverTracer.png"
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )

##define (S_RIVRUVMAP_AUTH) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_RiverAuthorizer.png"
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )

##define (PROC_UVMAP_STEP2) as <PROCWRL_NATUV_1A>
   #tag Method    =  (   PROC_UVGEN_RIVERMAP )
   #tag inZFcn    =  (                 NONE )
   #tag inZMap    =  (      S_RIVRUVMAP_LINE )
   #tag inAMap    =  (      S_RIVRUVMAP_AUTH )
   #tag exUMap    =  (               S_WRAPU )
   #tag exVMap    =  (               S_WRAPV )
   #tag inZNorm   =  (                  NO )
   #tag Zmin      =  (                0.00 )
   #tag Zmax      =  (                1.00 )
   #tag UXform    =  "Arg(z); u=Clamp(z,0.0,1.0); Ret(u);"
   #tag VSpan     =  (                30.0 )
   #tag VBand     =  (                10.0 )
   #tag SplineMap =  (                 YES )
   #tag SplineDiv =  (                  10 )
   #tag SplineItr =  (                   3 )
   #tag MinSize   =  (                  10 )
   #tag MinAper   =  (                   5 )
   #tag MaxAper   =  (                   5 )
   #tag MaxJoin   =  (                  11 )
   #tag RiverEdge =  (                0.03 )
   #tag RiverTerm =  (           0xFF0000FF )
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

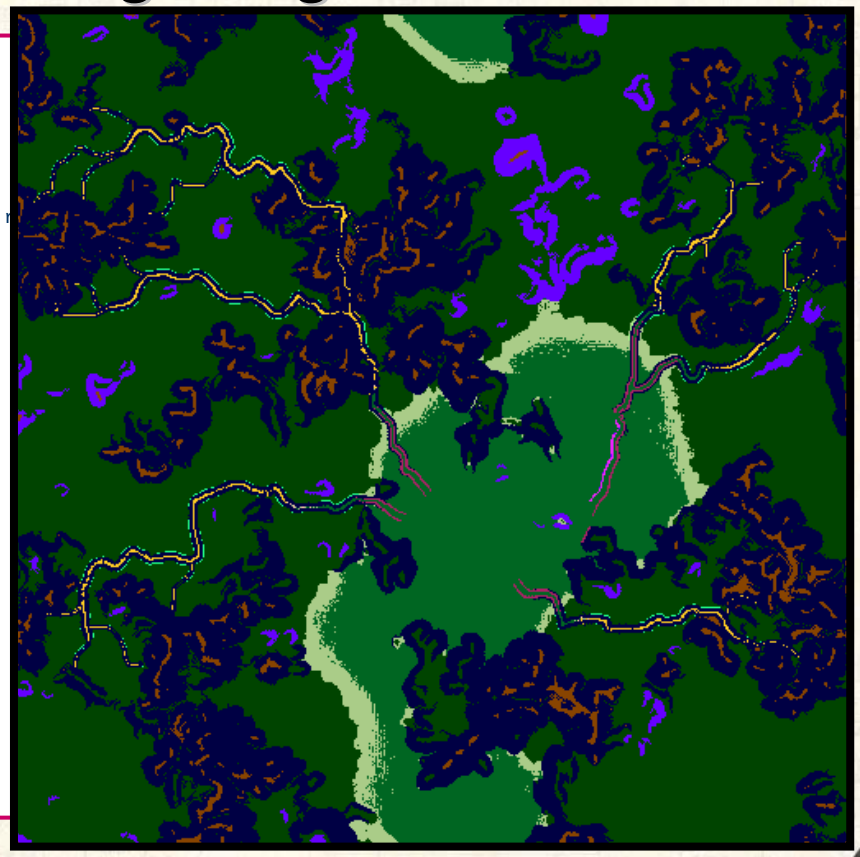
- **DCT generation, step 1/2. A statistical topology analysis is performed on the elevation to isolate and characterize different geological features.**

```
##define (PROC_TEXGEN_EX02) as <PROCTEX_GENDCT_1A>

   #tag Method       =  ( PROC_DCGEN_FTCODE )
   #tag TexA         =  (                 - )
   #tag CodA         =  (         0x00000000 )
   #tag FieldZ       =  (             F_ELEV )
   #tag FieldS       =  (             F_ELEV )
   #tag Scale        =  (    0.05 ,     0.05 ) ' Discretization r

   #tag RiverMap     =  (             S_RIVR )
   #tag OceanLevel   =  (              0.150 )
   #tag RiverLevel   =  (              0.001 )
   #tag SlopeMinG    =  (               25.0 )
   #tag SlopeMinW    =  (               25.0 )
   #tag SlopeSize    =  (                  8 )
   #tag AdjRadius    =  (                  2 )
   #tag FtFileName   =  "Files_Runtime/Ex02_Features.dat"
   #tag FtFileLoad   =  (                YES )

   #tag cdPlain1     =  (         0xFF004400 ) 'diag dct coding
   #tag cdPlain2     =  (               0x00 ) 'diag dct coding
   #tag cdPlain3     =  (               0x00 ) 'diag dct coding
   #tag cdPlain4     =  (               0x00 ) 'diag dct coding
   #tag cdPlainT     =  (         0xFF880000 ) 'diag dct coding
   #tag cdPlainH     =  (               0x00 ) 'diag dct coding
   #tag cdTalus1     =  (         0xFF000044 ) 'diag dct coding
   #tag cdTalusRdg   =  (               0x00 ) 'diag dct coding
   #tag cdTalusMnt   =  (         0xFF6600FF ) 'diag dct coding
   #tag cdRiverBtm   =  (         0xFFFF8822 ) 'diag dct coding
   #tag cdRiverShr   =  (         0xFF22FF88 ) 'diag dct coding
   #tag cdOceanBtm   =  (         0xFF002222 ) 'diag dct coding
   #tag cdOceanShr   =  (         0xFFAA8888 ) 'diag dct coding
   #tag cdROTCnv     =  (         0xFFAA2222 ) 'diag dct coding
   #tag cdOceanTal   =  (               0x00 ) 'diag dct coding
   #tag cdRiverTal   =  (               0x00 ) 'diag dct coding
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

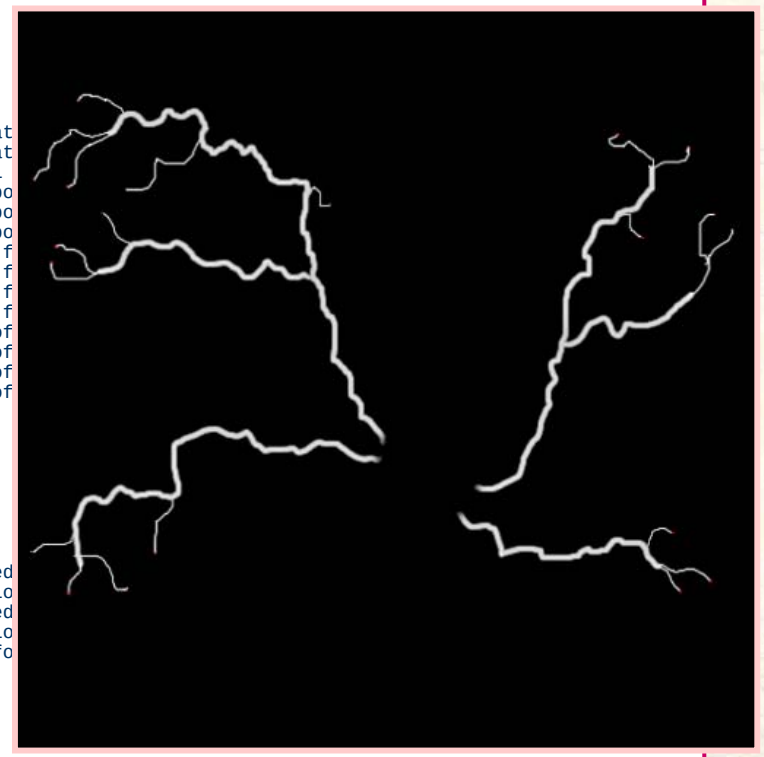
D108B:Ex02 – WorldRT3A

- **DCT generation, step 1/2. Note how we reuse our river tracing control map to manage the coding of terrain types near rivers.**

```
  #tag PT_FtZ       =  (   0.5  ,     20.0 ) 'Pass range, elevation
  #tag PT_FtSize    =  (   0.0  ,    600.0 ) 'Pass range, area
  #tag PT_FtComp    =  (   0.0  , 999999.0 ) 'Pass range, compactness
  #tag PT_FtDiff    =  (   0.0  , 999999.0 ) 'Pass range, diffheit
  #tag PT_AdjAratT  =  (   0.0  , 999999.0 ) 'Pass range, ratio of feat
  #tag PT_AdjAratP  =  (   0.0  , 999999.0 ) 'Pass range, ratio of feat
  #tag PT_AdjCountT =  (     1  ,        1 ) 'Fail if any other than 1
  #tag PT_AdjCountP =  (     0  ,        0 ) 'Fail if any plain neighbo
  #tag PT_AdjCountR =  (     0  ,        0 ) 'Fail if any ocean neighbo
  #tag PT_AdjCountO =  (     0  ,        0 ) 'Fail if any river neighbo
  #tag PT_AdjFracT  =  ( 0.000  ,    1.000 ) 'Fail if common boundary f
  #tag PT_AdjFracP  =  ( 0.000  ,    0.000 ) 'Fail if common boundary f
  #tag PT_AdjFracR  =  ( 0.000  ,    0.000 ) 'Fail if common boundary f
  #tag PT_AdjFracO  =  ( 0.000  ,    0.000 ) 'Fail if common boundary f
  #tag PT_JctRelMin =  (    NO  ,       NO ) 'Fail if (common metric of
  #tag PT_JctRelMax =  (    NO  ,       NO ) 'Fail if (common metric of
  #tag PT_JctRelAvg =  (     1  ,       NO ) 'Fail if (common metric of
  #tag PT_JctRelStd =  (    NO  ,       NO ) 'Fail if (common metric of


. . . Many coding control parameters skipped . . .

  #tag OC_ShTopDZ   =  (            0.020 )
  #tag OC_ShBtmDZ   =  (           -0.010 )
  #tag OC_ShMaxDst  =  (              4.0 )
  #tag OC_ShDstMod  =  (             NONE )
  #tag RB_AdjFracT  =  ( 0.800  ,    1.000 ) 'Pass: Riverplain bordered
  #tag RB_AdjFracP  =  ( 0.000  ,    1.000 ) 'Pass: Optional restrictio
  #tag RS_AdjFracT  =  ( 0.000  ,    0.500 ) 'Pass: Riverplain bordered
  #tag RS_AdjFracP  =  ( 0.000  ,    1.000 ) 'Pass: Optional restrictio
  #tag ROT_ZAdj     =  (            0.080 ) 'Readjustment elevation fo
```



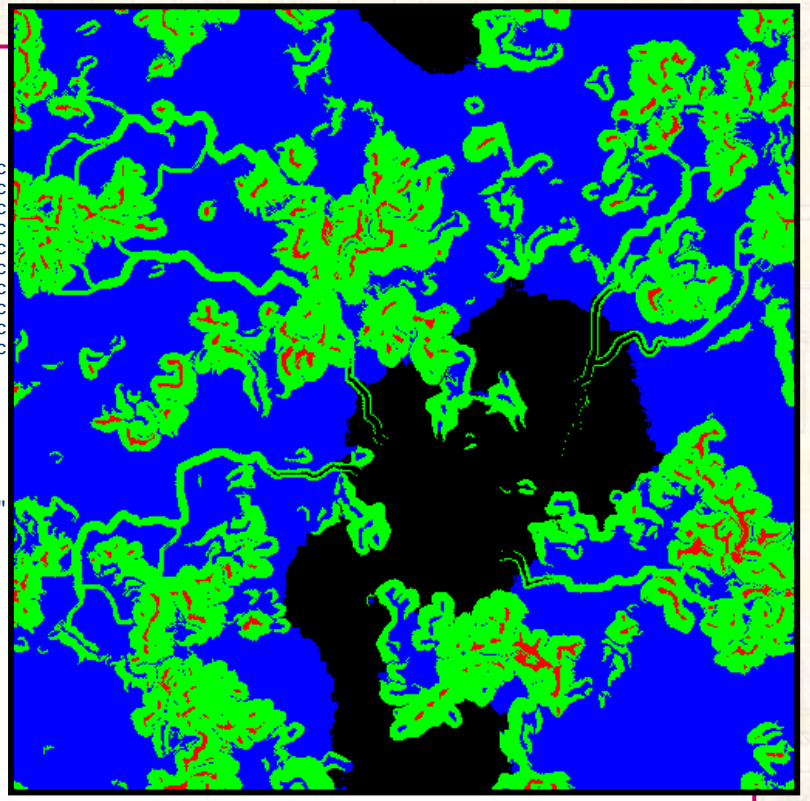
Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **DCT generation, step 2/2. We translate the type grid from step 1/2 into renderable DCT data used for the ground layer.**

```
##define (PROC_DCT_STEP2) as <PROCTEX_GENDCT_1A>
   #tag Method    =  (  PROC_DCGEN_TGXLT )
   #tag TexA      =  (               NONE )
   #tag OpMap     =  (  0xFF004400 , TC_PLAIN ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFF22FF88 , TC_PLAIN ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFF884400 , TC_HITOP ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFF000044 , TC_TALUS ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFF6600FF , TC_TALUS ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFFFFCC22 , TC_TALUS ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFF006622 , TC_LOBTM ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFFAACC88 , TC_LOBTM ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFFEE22FF , TC_LOBTM ,  YES , YES ) 'Stric
   #tag OpMap     =  (  0xFFAA2266 , TC_LOBTM ,  YES , YES ) 'Stric
   #tag Norm3     =  (                NO )
   #tag Norm4     =  (                NO )
   #tag Domine3   =  (                NO )
   #tag Domine4   =  (                NO )
   #tag DomineL   =  (               YES )

##define (S_DCT_GRNDBASE) as <SURFACE>
   #tag File      =  "Files_Media/Ex02_Map_DCT_Step1_CodedType.bmp"
   #tag Procedure =  (   PROC_DCT_STEP2 )
   #tag Definer   =  (          DEF_TEX )
   #tag LoadNow   =  (              1 )
```



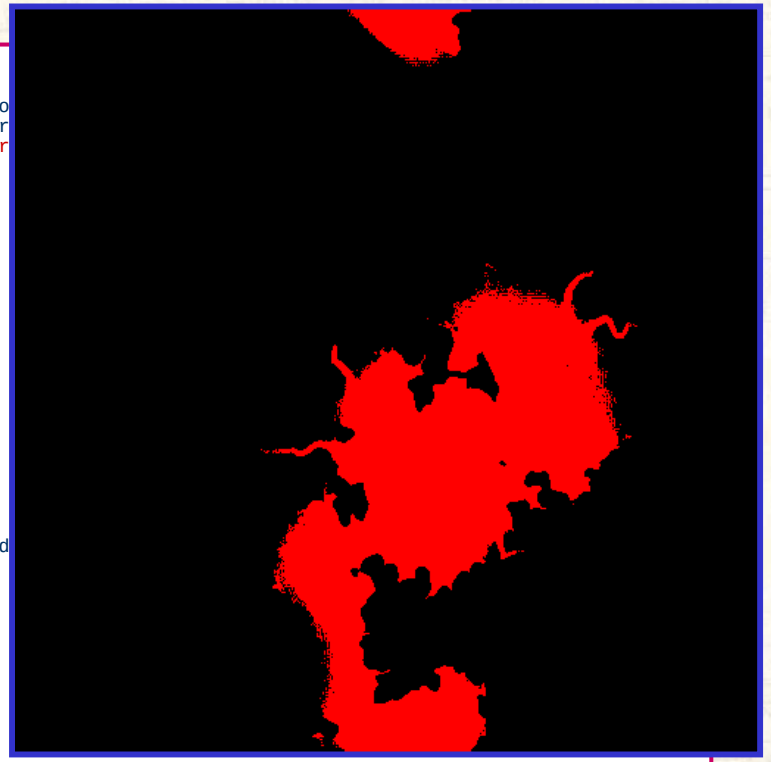
Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **DCT generation, extra. A DCT outline of the water surface is used by the assembler. Note that water level is defined by the global Z=0.15 setting.**

```
##define (F_DCT_LAYER_WATRRED_EX02) as <FUNCTION_S> ' Encodes water. No
   #tag Operation =  " Arg(u,v,z,s,l,c,n,r,g);"     ' Elevation field r
   #tag Operation =  " o=Gt(z,0.15,0.0,1.0); "      ' Elevation field r
   #tag Operation =  " Ret(o); "

##define (PROC_DCT_LAYER_WATR_EX02) as <PROCTEX_GENDCT_1A>
   #tag Method    =  ( PROC_DCGEN_HFRGBA )
   #tag Norm3     =  (                NO )
   #tag Norm4     =  (                NO )
   #tag Domine3   =  (                NO )
   #tag Domine4   =  (               YES )
   #tag DomineL   =  (                NO )
   #tag TexA      =  (                 - )
   #tag CodA      =  (        0x00000000 )
   #tag FieldZ    =  (            F_ELEV )
   #tag FieldS    =  (                 - )
   #tag FieldL    =  (                 - )
   #tag FieldC    =  (                 - )
   #tag FieldN    =  (                 - )
   #tag Scale     =  (    0.05 ,    0.05 ) ' Discretization related
   #tag FL_Level  =  (                0.0 )
   #tag FC_Level  =  (                0.0 )
   #tag Channel1R =  ( F_DCT_LAYER_WATRRED_EX02 ) ' Used for water Grid
   #tag Channel1G =  (                     NONE )
   #tag Channel1B =  (                     NONE )
   #tag Channel1A =  (                     NONE )
```



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Putting it all together: the generation script. Note the use of source references for tile UV data.**

```
##define (PROC_WGEN_EX02) as <PROCWRL_ASM_RT3A>

   #tag Tile_Layers       =  (                        2 )
   #tag Tile_Numbers      =  (  0 ,       72 ,       72 )
   #tag Tile_SideSize     =  (  0 ,     0.35 ,     0.35 )
   #tag Tile_Vertices     =  (  0 ,        8 ,        8 )
   #tag Tile_Definer      =  (  0 ,      VTD_LAYER_GRND )
   #tag Tile_Kinex        =  (  0 ,      NONE ,        - )
   #tag Tile_ElevationF   =  (  0 ,              F_ELEV )
   #tag Tile_CarvingF     =  (  0 ,                NONE )
   #tag Tile_UV0_Method   =  (  0 ,  PROC_UVSYS_RNDTILE ) 'Tile coordinate: intrinsic
   #tag Tile_UV1_Method   =  (  0 ,   PROC_UVSYS_IMPORT ) 'Tile coordinate: natural terrain import
   #tag Tile_UV2_Method   =  (  0 ,    PROC_UVSYS_WORLD ) 'Tile coordinate: world DCT
   #tag Tile_UV0_Source   =  (  0 ,        - ,        - ) 'Tile coordinate data source
   #tag Tile_UV1_Source   =  (  0 ,  S_WRAPU ,  S_WRAPV ) 'Tile coordinate data source
   #tag Tile_UV2_Source   =  (  0 ,        - ,        - ) 'Tile coordinate data source
   #tag Tile_BoxOverride  =  (  0 ,     0.00 ,     0.00 )

   #tag Tile_Numbers      =  (  1 ,       72 ,       72 )
   #tag Tile_SideSize     =  (  1 ,     0.35 ,     0.35 )
   #tag Tile_Vertices     =  (  1 ,        6 ,        6 )
   #tag Tile_Definer      =  (  1 ,      VTD_LAYER_WATR )
   #tag Tile_Kinex        =  (  1 ,      NONE ,        - )
   #tag Tile_ElevationF   =  (  1 ,   F_LAYER_WATR_EX02 )
   #tag Tile_CarvingF     =  (  1 ,                NONE )
   #tag Tile_UV0_Method   =  (  1 ,     PROC_UVSYS_TILE ) 'Tile coordinate: intrinsic
   #tag Tile_UV1_Method   =  (  1 ,    PROC_UVSYS_WORLD ) 'Tile coordinate: world DCT
   #tag Tile_UV0_Source   =  (  1 ,        - ,        - ) 'Tile coordinate data source
   #tag Tile_UV1_Source   =  (  1 ,        - ,        - ) 'Tile coordinate data source
   #tag Tile_BoxOverride  =  (  1 ,    -0.10 ,     0.10 )

   #tag Grid_TileSource   =  (                        0 )
   #tag Grid_ElevationF   =  (        F_GRIDHEIGHT_EX02 )
   #tag Grid_TypeExport   =  (            S_WTYPE_EX02 )
   #tag Grid_TypeImport   =  (                    NONE )
   #tag Grid_TypeAsmMap   =  (           S_DCT_GRNDBASE ,   TC_CLIFF ,  TC_CLIFF ,  TC_FLATS ,          - )
   #tag Grid_TypeAsmMap   =  (    S_DCT_LAYER_WATR_EX02 ,   TC_WATER ,         - ,         - ,          - )

   #tag Wrld_Layer_Map    =  (  0 ,  L_TILES_G1N ,         S_DCT_GRNDBASE ,  NO , 0xFFFFFFFF ,   NONE , NONE , NONE ,  0x00FFFFFF )
   #tag Wrld_Layer_Map    =  (  1 ,  L_TILES_W1N ,  S_DCT_LAYER_WATR_EX02 ,  NO , 0xFFFFFFFF ,   NONE , NONE , NONE ,  0x00FFFFFF )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Closing design details**
  - Cliff-side texture
  - Water surface vertex shader modifications
  - Cliff texturing vertex shader modifications
  - Improvement analysis for Ex02
- **Script design notes**
  - Because terrain refinement procedures run time can be considerable, the generation script is divided into independent blocks.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Cliff texture design:**
  - Procedurally generated (filters)
  - Minimal V-variation to downplay UV map artifacts. Required V-periodicity.
  - Match scene tones, subdue saturation.
  - Can be up-sampled and UV modified in the shader.



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Water surface vertex shader modifications**

```
//*************************************************************************//
//  Shader configuration constants: remember z is negative under water     //
//*************************************************************************//
// Base and depth bias RGBA water properties
static float4  WaterLvl  = float4(  .15,  .15,  .15,  .15);
static float4  ClrZsrfc  = float4(  0.0,  0.3,  0.7,  0.6);
static float4  ClrZdeep  = float4(  0.0,  0.2,  0.8,  0.8);
// Wave speed, scale and motion coefficients
static float4  WavSScl   = float4(  0.0000025, 0.0000025, 8.0, 8.0);
static float4  WavMove   = float4( 1.1,   1.3, -1.2, -1.4);


 . . .

/* Diffuse color and transparency based on water depth */

Out.vDiffuse      = (WaterLvl - In.vPosition.zzzz)/WaterLvl;
Out.vDiffuse      = ClrZsrfc + (ClrZdeep-ClrZsrfc)*Out.vDiffuse;
Out.vDiffuse.a   *= Out.vDiffuse.a;
```

- **These changes accommodate the fact that the water level is set at a variable design value Z=C.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Land surface vertex shader modifications scale the cliff texture to match flat land, and perturb the rock layers by tilting them procedurally:**

```
/* Texture coordinates output for 4-texture blend */
Out.vTexCoord0.xy = In.vTexCoord0.xy     ;// tile-based system, flatland
Out.vTexCoord1.x  = 3.0*In.vTexCoord1.x+0.08*In.vPosition.x  ;// natural UV system, wrapped cliff features
Out.vTexCoord1.y  = 1.0*In.vTexCoord1.y  ;// natural UV system, wrapped cliff features
Out.vTexCoord2.xy = In.vTexCoord0.xy     ;// tile-based system, flatland
Out.vTexCoord3.xy = In.vTexCoord0.xy     ;// tile-based system, flatland
Out.vTexCoord4.xy = In.vTexCoord2.xy     ;// world UV system, DCT
```

- **Cliff texturing improvements**
  - **Use a U-perturbation texture for sampling cliff texels, to create layering variegation and improve realism.**
  - **Use a relaxation procedure on the V-coordinate to allow the use of a texture with more V-detail.**
  - **Synthesize the cliff texture in the pixel shader.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

Pixel shader, alternative HLSL formulation.

```
sampler TS0 = sampler_state { //Terrain texture (R)
  texture   = Tex0;
  mipfilter = POINT;
  . . . };

. . .

sampler TSD = sampler_state { //Terrain DCT
  texture   = Tex4;
  mipfilter = LINEAR;
  . . . ;};

struct VS_OUTPUT  { // Extant processed vertex stream
    float4 vPosition        : POSITION ;
    float4 vDiffuse         : COLOR0   ;
    float4 vSpecular        : COLOR1   ;
    float2 vTexCoord0       : TEXCOORD0;
    float2 vTexCoord1       : TEXCOORD1;
    float2 vTexCoord2       : TEXCOORD2;
    float2 vTexCoord3       : TEXCOORD3;
    float2 vTexCoord4       : TEXCOORD4;};

struct PS_OUTPUT  { // Extant pixel shader stream
    float4 Color            : COLOR0   ; };

//###########################################################################//
//#   Shader principal operations                                           #//
//###########################################################################//

PS_OUTPUT PS_Ex03_LG1(VS_OUTPUT In){  PS_OUTPUT Out = (PS_OUTPUT)0;

    /* Sample texture and DCT data */
    cT1 = tex2D(TS0, In.vTexCoord0.xy);
    cT2 = tex2D(TS1, In.vTexCoord1.xy);
    cT3 = tex2D(TS2, In.vTexCoord2.xy);
    cT4 = tex2D(TS3, In.vTexCoord3.xy);
    cDC = tex2D(TSD, In.vTexCoord4.xy);

    /* Pixel color generation. Note shadow reduction in specular areas */
    Out.Color      = cT1*cDC.rrrr + cT2*cDC.gggg + cT3*cDC.bbbb + cT4*cDC.aaaa;
    Out.Color      = In.vDiffuse * Out.Color + In.vSpecular;

    /* Shader output */
    return Out;};
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

Ex02 Terrain Improvement Analysis

- While Ex02 is a significant step forward (terrain is almost production grade), much can be improved:

Cliff terrain

- Real cliffs are not as smoothly sloped and contoured. They have a variety of protrusions, vertical drops, pockets of sediment collection and small-scale textured detail.
- This will be addressed in Parts II and IV thru alternative world geometry structures, and more advanced terrain/UV/DCT generation procedures.

Textural appearance:

- Textures between flatlands and cliffs appear somewhat out of scale. This can be improved by proper sizing of tiles, but is ultimately the result of the intrinsic grid stretching.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

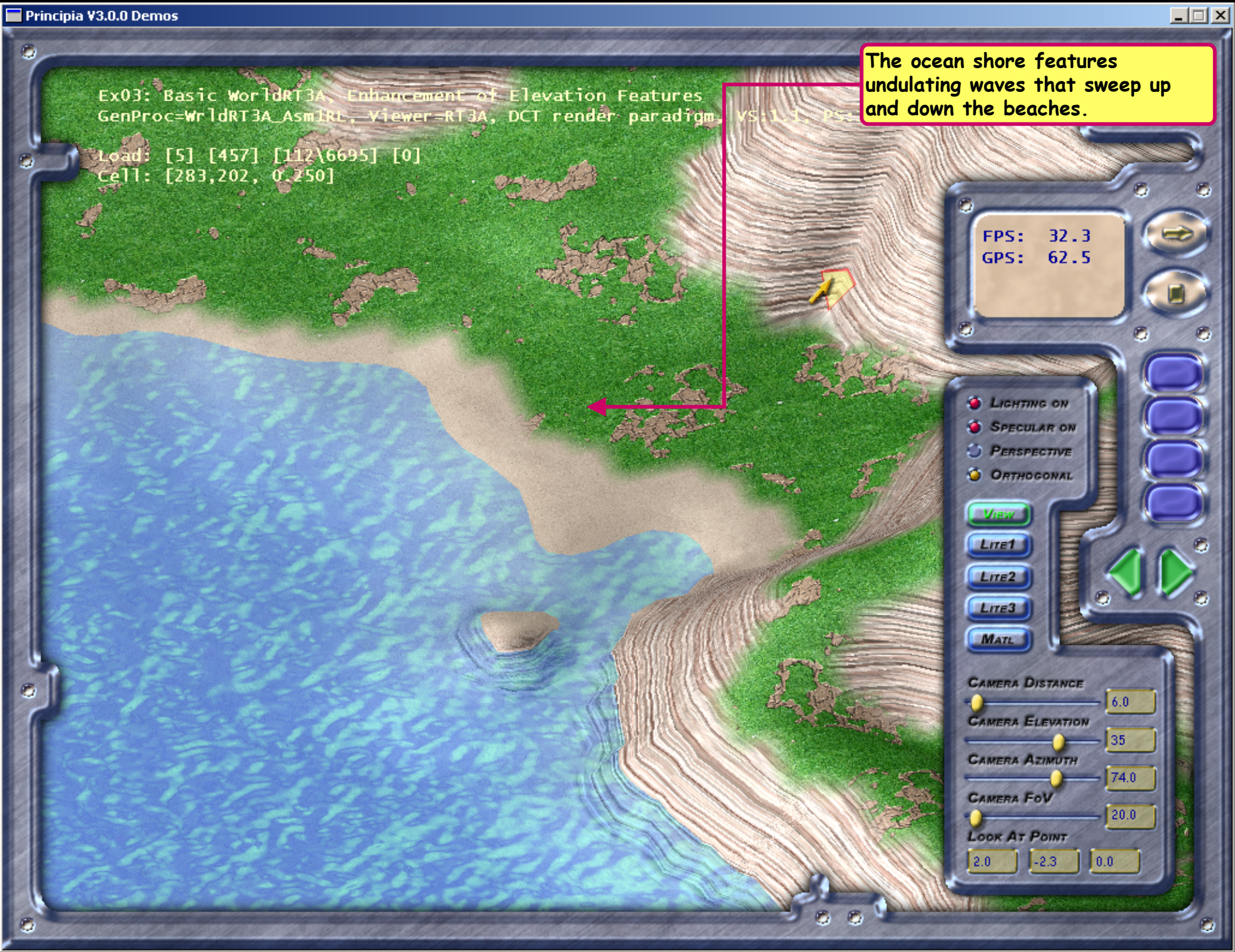
D108B:Ex02 – WorldRT3A

- **Textural appearance (contd):**
  - Even if compositing is better, surfaces still appear flat and repetitive. This will be addressed through bump and displacement mapping, and palletized materials in Ex03.
- **Riverbeds**
  - Some areas show noticeable jaggedness. This will be corrected procedurally later.
- **Junctions**
  - Junctions are smeared. This is a by-product of the grid structure, and will be addressed in Parts II and IV.
- **The intrinsic limitation of WorldRT3A is the uniform rectangular grid tile array.**
  - More advanced world designs use tile sets that are not RC constrained and naturally fit terrain features.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [4] [383] [57\6695] [0]
Cell: [258,425, 0.473]
FPS: 66.7
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
The land surface in Ex03 features scattered sharp-edged patches of grass and soil. Yet, this is achieved using a single texturing pass!
The richer surface detail in Ex03 is rendered using composite textures and per-pixel bump-mapping, on top of the traditional DCT blending. These are foundation techniques for terrain rendering in production.


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1
Load: [8] [492] [125\6695] [0]
Cell:
Our Ex03 terrain has rivers running along the eroded channels created previously. A major advance in Ex03 is water rendering. Water surface tile data is procedurally generated for animating rivers and oceans.
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
35
Camera Azimuth
150.0
Camera FoV
20.0
Look At Point
-9.2
3.7
0.0


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_AsmIRL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:
Load: [5] [457] [112\6695] [0]
Cell: [283,202, 0.250]
The ocean shore features undulating waves that sweep up and down the beaches.
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
35
Camera Azimuth
74.0
Camera FoV
20.0
Look At Point
2.0
-2.3
0.0


Principia V3.0.0 Demos
Rivers blend smoothly with the ocean where they open in terms of surface levels, depth, waves and even small-scale wave animation. This is achieved by proper shader design and by generating proper water surface reference data in a preliminary liquid geometry procedure.
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.
Load: [5] [459] [74\6695] [0]
Cell: [389,198, 0.176]
FPS: 62.5
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
4.2
Camera Elevation
35
Camera Azimuth
74.0
Camera FoV
20.0
Look At Point
6.1
-3.1
0.0


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:
Load: [12] [669] [191\6695] [0]
Cell: [225,189, 0.746]
Rivers fill their channels, merge and flow in a realistic manner. The visual representation of this is achieved using an extension of the small-scale bump-wave rendition in the liquid shader. The underlying data is generated in a preliminary FX_Liquid1A procedure.
GPS: 31.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
9
Camera Elevation
40
Camera Azimuth
-45.0
Camera FoV
20.0
Look At Point
-3.1
-2.3
0.0


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
Note the good appearance of river beds and animated water flow. The water cascades down the terrain gradient steps naturally. Remember, all this is procedurally generated, using a sequence of terrain enhancement steps that started in Ex02.
GPS: 31.3
On
Off
Dir
Point
Spot
Glob
View
Lite1
Lite2
Lite3
Matl
cAmb 85 85 85
cDif 155 155 155
cSpt 100 100 100
POS 0.0 0.0 0.0
DIR -1.00 0.00 -1.00
Coef 1.00 0.00 0.00
R,F 999 1.00
T,F 25 45


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A; DCT render paradigm, VS:1.1, PS:1.4
Load: [4] [382] [51\6695] [0]
Cell:
FPS: 62.5
GPS: 32.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
A close look shows that there is still room for improvement:
-Talus UV mapping is still jagged in some areas and results in off-scale relationship between land and talus surfaces.
-There are still jagged areas, even if their impact on lighting is less visible than in the regularly triangulated Ex02.
-Talus detail poor. There is a rather uniform layer banding and sloped geometry.
Such improvements will be addressed in the advanced world rendering chapters.

D108B:Ex03 – WorldRT3A

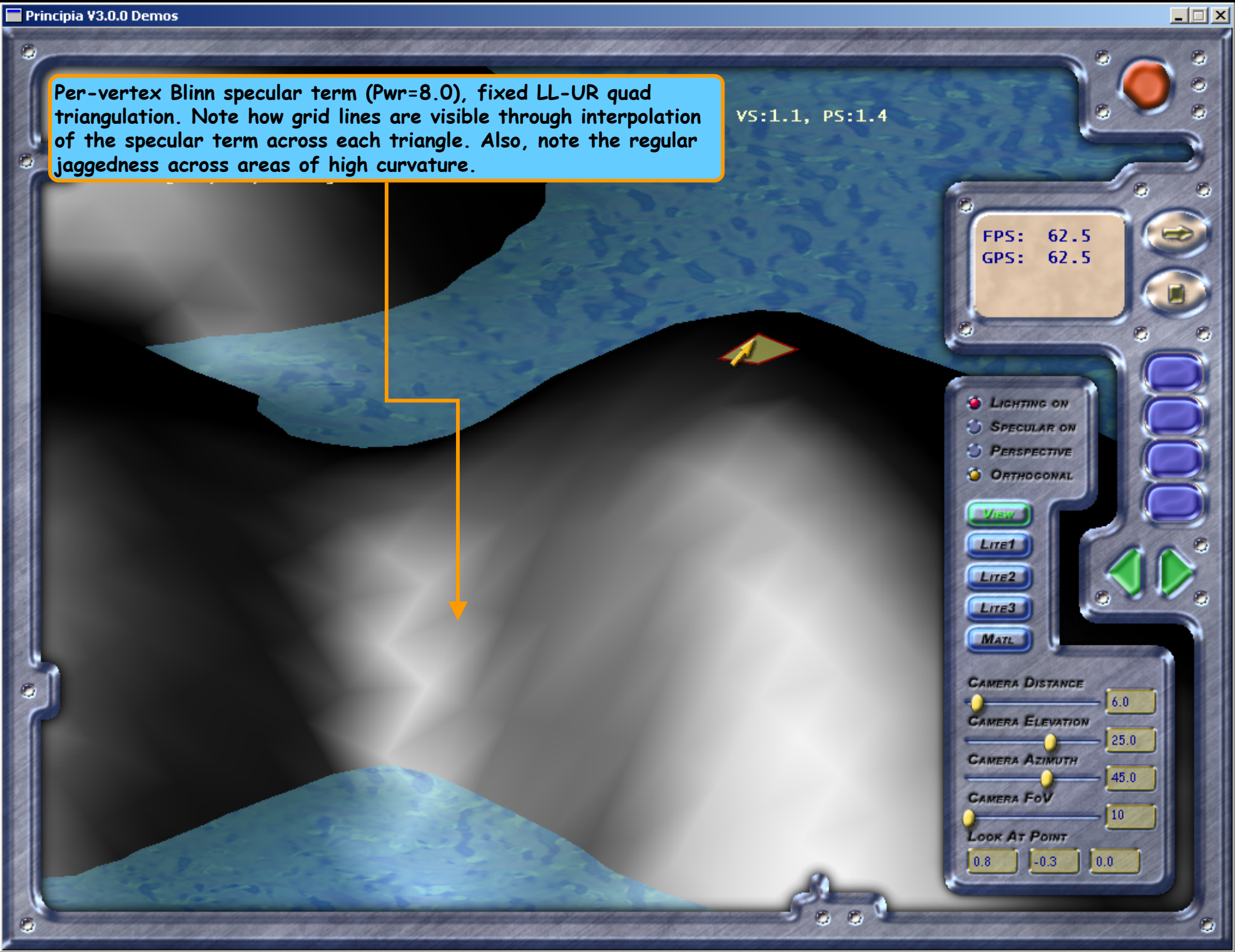
- **Objectives**
  - Further enhance the appearance of our Ex02 terrain by introducing several additional fundamental techniques.
- **Requirements and Implementation**
  - Enhanced water
  - Enhanced texturing
  - Enhanced visual interaction
- **D108B aims to present the foundation tools for worlds production. Our quest for further terrain visual enhancements will continue in Part II.**

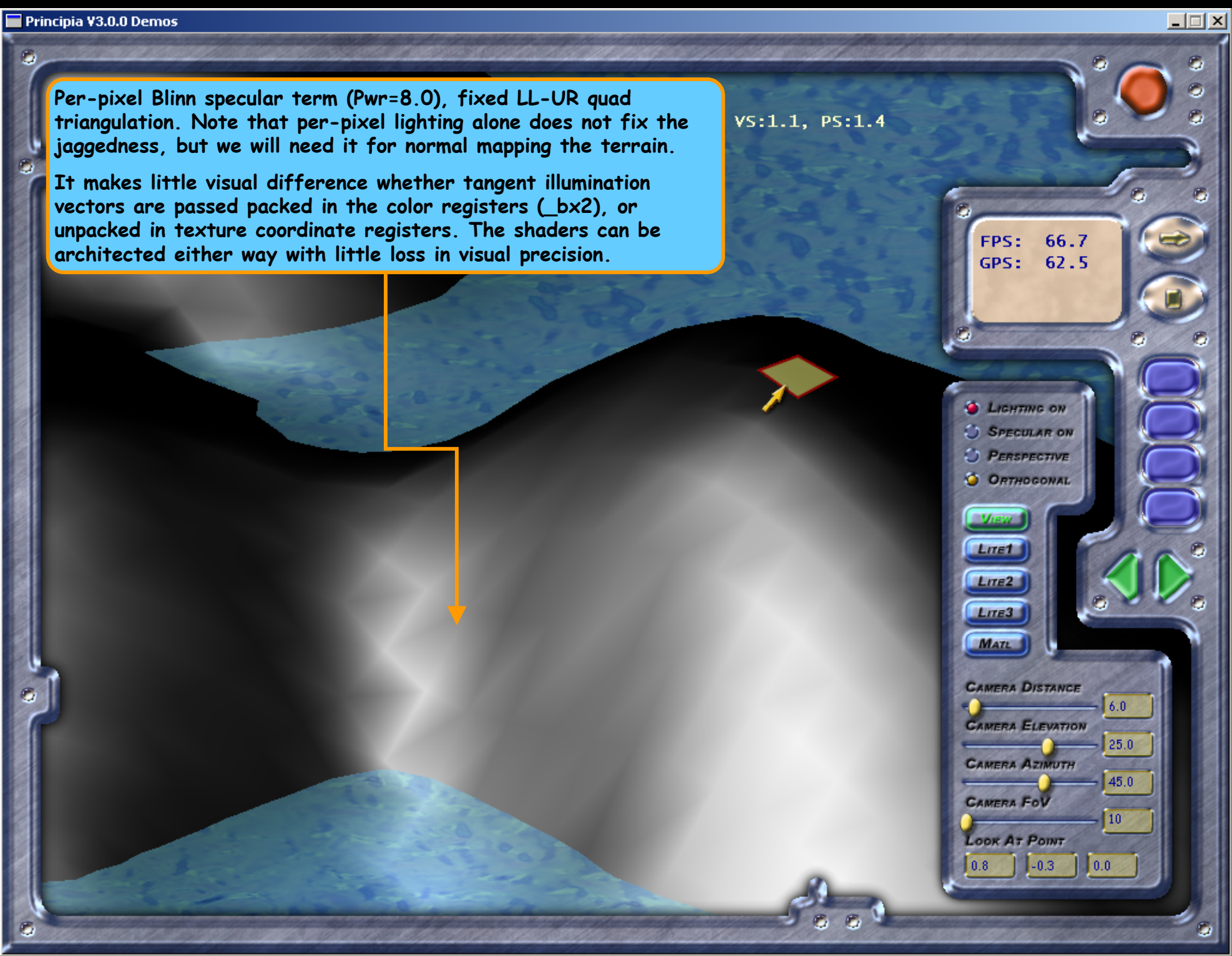
Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

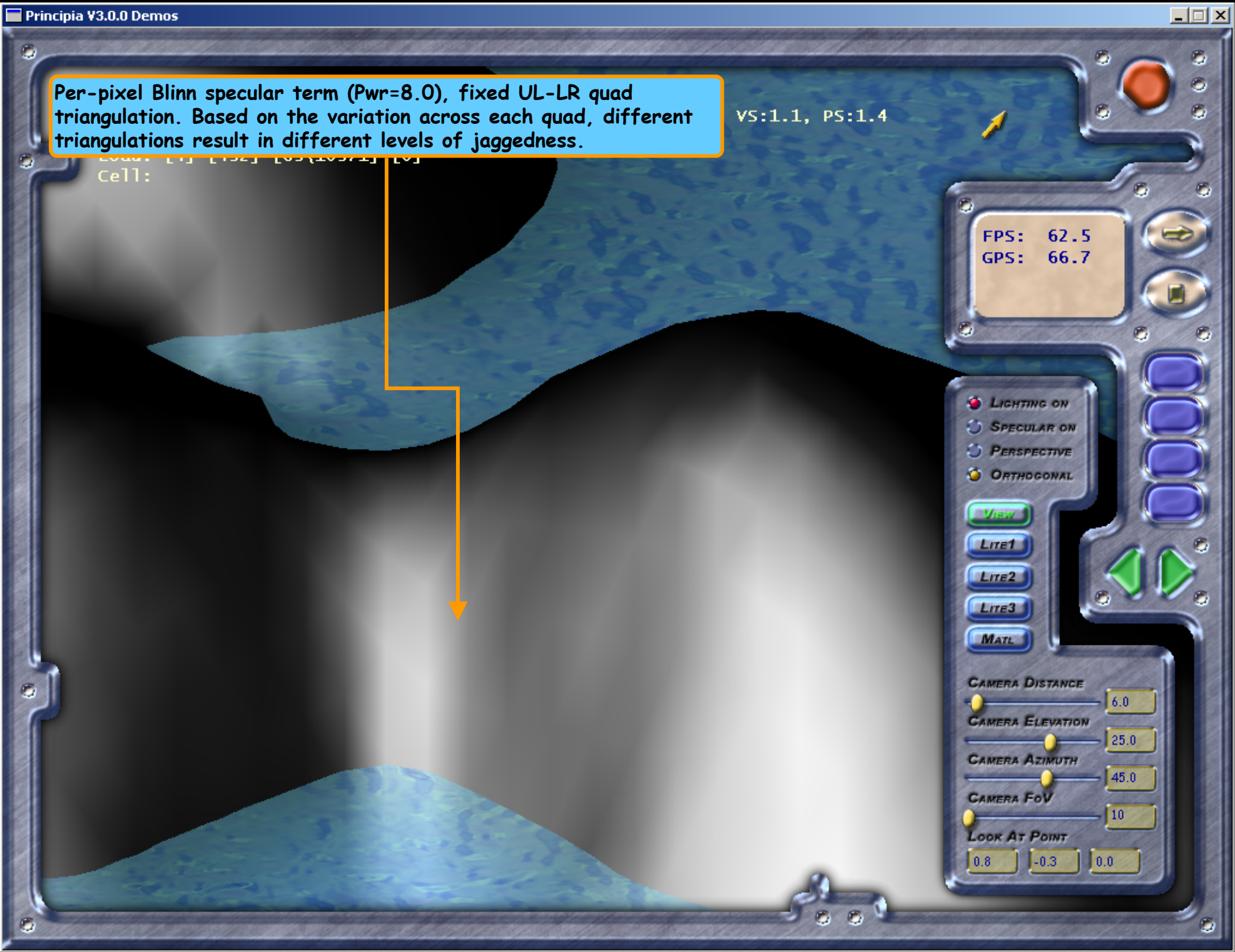
D108B:Ex03 – WorldRT3A

- **New world enhancements in Ex03**
  - Jaggedness-minimal triangulation of mesh
  - Bump-mapped terrain for small-scale detail
  - Palletized land textures to avoid patterning
  - Running rivers that open smoothly into the ocean
  - Undulating ocean waves
  - Prevent display of world “rim” discontinuity
- **Related enhancements <u>not</u> in Ex03 (covered in advanced world rendering chapters)**
  - Breaker waves on ocean shore along natural UV lines
  - Fine-scale riverbed features such as rocks and stones
  - Surface reflection off water edges
  - Automatic extension of tiles to fill “rim”

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Per-vertex Blinn specular term (Pwr=8.0), fixed LL-UR quad triangulation. Note how grid lines are visible through interpolation of the specular term across each triangle. Also, note the regular jaggedness across areas of high curvature.
VS:1.1, PS:1.4
FPS: 62.5
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
10
Look At Point
0.8
-0.3
0.0


Principia V3.0.0 Demos
Per-pixel Blinn specular term (Pwr=8.0), fixed LL-UR quad triangulation. Note that per-pixel lighting alone does not fix the jaggedness, but we will need it for normal mapping the terrain.
It makes little visual difference whether tangent illumination vectors are passed packed in the color registers (_bx2), or unpacked in texture coordinate registers. The shaders can be architected either way with little loss in visual precision.
VS:1.1, PS:1.4
FPS: 66.7
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
10
Look At Point
0.8
-0.3
0.0


Principia V3.0.0 Demos
Per-pixel Blinn specular term (Pwr=8.0), fixed UL-LR quad triangulation. Based on the variation across each quad, different triangulations result in different levels of jaggedness.
VS:1.1, PS:1.4
Cell:
FPS: 62.5
GPS: 66.7
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
10
Look At Point
0.8
-0.3
0.0


Principia V3.0.0 Demos
Several quad triangulation control methods are available in Principia. Here, a triangulation control based on normal vector gradient automatically adjust the triangulation of each quad to minimize the apparent jaggedness over the terrain.
Note that automatic triangulation control precludes the use of a single vertex stream index table instance, and thus increases somewhat vertex bandwidth and storage requirements.
FPS: 62.5
GPS: 66.7
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
8.9
-5.8
0.0

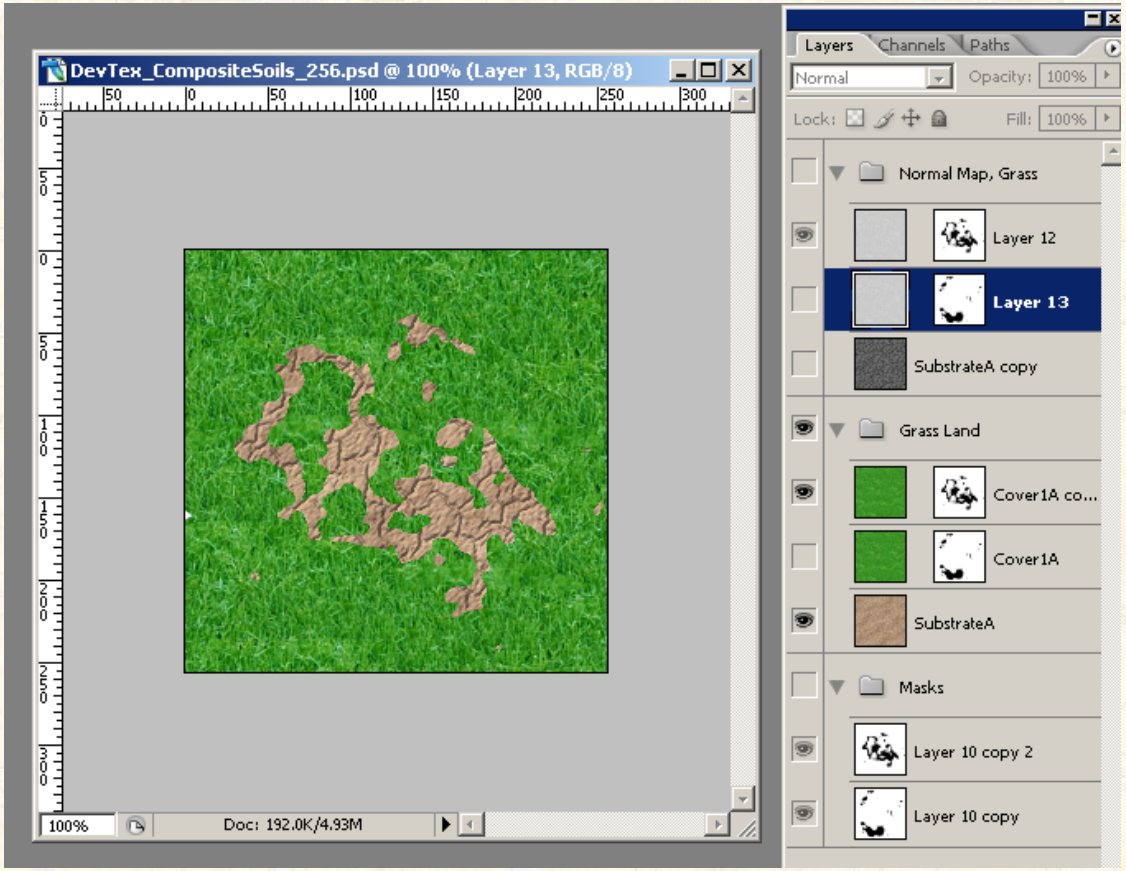
D108B:Ex03 – WorldRT3A

- **Terrain texturing considerations**
  - Texture compositing reduces the number of DCT layers required to achieve a given level of textural richness.
  - Bump-mapping complements texture compositing, but its visual gains may have to be balanced against GPU resources utilization.
  - Composited textures may need to be palletized to avoid the creation of regular pattern repetition. Palletized textures in turn require careful tiling design.
- **Terrain bump-mapping**
  - Ex03 shows a minimal 3DCT/2bump-map design that still requires a PS20-capable GPU and utilizes 8 texture slots. The texture sizes in Ex03 have been reduced to 256x256 to maintain frame rate.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Texture compositing. Proper design of blend masks is the crux of textural compositing.**



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Per-pixel bumped diffuse-specular DCT-blending terrain vertex and pixel shaders (3DCT blend).**

```
struct VS_INPUT
{   // Inbound vertex buffer stream
    float3 vPosition        : POSITION ;
    float3 vNormal          : NORMAL   ;
    float2 vTexCoord0       : TEXCOORD0; // Tile-based system
    float2 vTexCoord1       : TEXCOORD1; // Natural UV system
    float2 vTexCoord2       : TEXCOORD2; // World DCT system
};

struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ;
    float4 vDiffuse         : COLOR0   ;
    float4 vSpecular        : COLOR1   ;
    float2 vTexCoord0       : TEXCOORD0;
    float2 vTexCoord1       : TEXCOORD1;
    float2 vTexCoord2       : TEXCOORD2;
    float2 vTexCoord3       : TEXCOORD3;
    float3 vLite            : TEXCOORD4;
    float3 vHalf            : TEXCOORD5;
    float2 vTexCoord6       : TEXCOORD6;
    float2 vTexCoord7       : TEXCOORD7;
};

//*************************************************************************//
//  Shader principal operations                                            //
//*************************************************************************//
VS_OUTPUT VS_Ex03_LG1(const VS_INPUT In) { VS_OUTPUT Out = (VS_OUTPUT) 0;

    /* Position output, clip space (world multiply strictrly unneeded) */
    wPos            = mul(float4(In.vPosition,1.0f), mWrld);
    Out.vPosition   = mul(wPos, mViewProj);

    /* Tangent coordinate system vectors in world space (strinctly unneeded) */
    wNorm           = mul  (In.vNormal  , (float3x3) mWrld);
    wNorm           = normalize(wNorm);
    wTang.x         =   wNorm.y + wNorm.z;
    wTang.y         =   wNorm.z - wNorm.x;
    wTang.z         = - wNorm.x - wNorm.y;
    wTang           = normalize(wTang);
    wBnrm           = cross(wNorm, wTang);
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Per-pixel bumped diffuse-specular DCT-blending terrain vertex and pixel shaders (3DCT blend).**

```
. . .

    /* Transformation matrix, world to tangent space */
    mW2T             = float3x3(wTang, wBnrm, wNorm);
    mW2T             = transpose(mW2T);

    /*  View and light reflection vectors in world space */
    wLite            = normalize(-vLDir);
    wView            = vEPos - wPos.xyz;
    wView            = normalize(wView);
    wHalf            = wLite+wView;
    wHalf            = normalize(wHalf);

    /* Lambert diffuse and ambient per-vertex color output */
    LdotN            = saturate(dot(wNorm, wLite));
    Out.vDiffuse.rgb  = vLDiff*LdotN + vLAmbi;
    Out.vDiffuse.a    = float1(1.0);

    /* Blinn specular per-vertex color output */
    HdotN            = saturate(dot(wNorm, wHalf));
    Out.vSpecular.rgb = /*vLSpec**/pow(HdotN, 8.0 /*vMPwr*/);
    Out.vSpecular.a   = float1(1.0);

    /* Light and half vectors in tangent space */
    tLite            = normalize(mul(wLite, mW2T));
    tHalf            = normalize(mul(wHalf, mW2T));

    /* Compress the vectors to the [0,1] range for pixel shader use */
    Out.vLite        = tLite;
    Out.vHalf        = tHalf;

    /* Texture coordinates output for 4-texture blend */
    Out.vTexCoord0.xy = In.vTexCoord2.xy                          ;// World DCT system
    Out.vTexCoord1.x  = 6.0*In.vTexCoord1.x+0.020*In.vPosition.x  ;// Natural UV system, wrapped cliff features
    Out.vTexCoord1.y  = 1.0*In.vTexCoord1.y                       ;// Natural UV system, wrapped cliff features
    Out.vTexCoord2.xy = In.vTexCoord0.xy                          ;// Tile-based system, flatland
    Out.vTexCoord3.xy = In.vTexCoord0.xy                          ;// Tile-based system, flatland
    Out.vTexCoord6.xy = Out.vTexCoord1.xy                         ;// Natural UV system
    Out.vTexCoord7.xy = In.vTexCoord0.xy                          ;// Tile-based system, flatland

return Out; }
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex02 – WorldRT3A

- **Per-pixel bumped diffuse-specular DCT-blending terrain vertex and pixel shaders (3DCT blend).**

```
ps.2.0
;------------------------------------------------------------------------;
;                                                                        ;
;   Per-pixel bump-mapped terrain illumination, with three channel       ;
;   DCT blending. Half-vector Blinn specular form in tangent space       ;
;                                                                        ;
;   t0  - DCT texture                                                    ;
;   t1  - base texture                                                   ;
;   t2  - base texture                                                   ;
;   t3  - base texture                                                   ;
;   t4  - light incidence vector in tangent space (alt in v0)            ;
;   t5  - half vector in tangent space (alt in v1)                       ;
;   t6  - bump normal texture packed in [01]                             ;
;   t7  - bump normal texture packed in [01]                             ;
;                                                                        ;
;------------------------------------------------------------------------;

def c0,  0.5,  0.5,  1.0,  1.0    ; unit normal packed in [01]
def c5, -0.5, -0.5, -0.5, -0.5    ; unpacking coefficient for PS20
def c6,  2.0,  2.0,  2.0,  2.0    ; unpacking coefficient for PS20
def c7,  0.0,  0.0,  1.0,  1.0    ; unit vector

;------------------------------------------------------------------------;
;   Per-pixel bump-mapped terrain illumination, with three channel       ;
;------------------------------------------------------------------------;

dcl      v0                       ; alt channel for packed lite vector
dcl      v1                       ; alt channel for packed half vector

dcl      t0                       ; dct texture
dcl      t1                       ; base texture, rock
dcl      t2                       ; base texture, land
dcl      t3                       ; base texture, sand
dcl      t4                       ; lite vector, unpacked
dcl      t5                       ; half vector, unpacked
dcl      t6                       ; bump texture, rock
dcl      t7                       ; bump texture, land

dcl_2d   s0                       ; sampler, smoot DCT
dcl_2d   s1                       ; sampler, point base tex
dcl_2d   s2                       ; sampler, point base tex
dcl_2d   s3                       ; sampler, point base tex
dcl_2d   s6                       ; sampler, point bump tex
dcl_2d   s7                       ; sampler, point bump tex
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Per-pixel bumped diffuse-specular DCT-blending terrain vertex and pixel shaders (3DCT blend).**

```
;-------------------------------------------------------------------------;
;   Data load in shader                                                   ;
;-------------------------------------------------------------------------;
texld    r0, t0, s0              ; dct load
texld    r1, t1, s1              ; base load
texld    r2, t2, s2              ; base load
texld    r3, t3, s3              ; base load
texld    r6, t6, s6              ; bump load
texld    r7, t7, s7              ; bump load
mov      r4, t4                  ; (r4) has (L)
mov      r5, t5                  ; (r5) has (H)

;-------------------------------------------------------------------------;
;   Textural DCT composition                                              ;
;-------------------------------------------------------------------------;
mul      r1, r1, r0.g            ; base composition
mad      r1, r2, r0.r, r1        ; base composition
mad      r1, r2, r0.b, r1        ; base composition
mad      r1, r3, r0.a, r1        ; r1 holds base color

mul      r6, r6, r0.g            ; normal composition, bumped
mad      r6, r7, r0.r, r6        ; normal composition, bumped
mad      r6, r7, r0.b, r6        ; normal composition, bumpless
mad      r6, c0, r0.a, r6        ; r6 holds normal in 01 range
add      r6, r6, c5              ; normal unpacking
mul      r6, r6, c6              ; normal unpacking

;-------------------------------------------------------------------------;
;   Per-pixel bumped illumination                                         ;
;-------------------------------------------------------------------------;
dp3_sat  r2, r6, r4              ; r2 holds (N.L)
dp3_sat  r3, r6, r5              ; r3 holds (N.H)
mul      r3, r3, r3              ; r3 holds (N.H)^2
mul      r3, r3, r3              ; r3 holds (N.H)^4
mul      r3, r3, r3              ; r3 holds (N.H)^8

;-------------------------------------------------------------------------;
;   Illumination compositing                                              ;
;-------------------------------------------------------------------------;
mul      r2, r2, c2          ; (D) x CD
add      r2, r2, c1          ; (D) x CD + CA
mul      r3, r3, c3          ; (S) x CS
mad      r0, r1, r2, r3      ; (Amb+Dif)*Base+Spc
mov      oC0, r0             ; output
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DX1 render paradigm, VS:1.1, PS:1.4
Load: [8] [720] [181\10371] [0]
Cell: [186,236, 0.421]
FPS: 66.7
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
-2.4
-0.7
0.0
Bumped specular component with 256-size textures.


Principia V3.0.0 Demos
Ex03: Basic World RT3A, Enhancement of Elevation Features
GenProc=World RT3A Asmbl, Viewer=RT3A, DCT render paradigm, VS=1.1, PS=1.4
Load: [10] [864] [163\1037] [01]
Cell: [194,234, 0.561]
FPS: 62.5
GPS: 66.7
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
-2.1
-1.0
0.0
Bumped diffuse component with 256-size textures.


Principia V3.0.0 Demos
Terrain bump-mapping works best when the illumination is at a strong angle with the view vector, such as in this overhead view with lateral light incidence.
FPS: 66.7
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
57.0
Camera Azimuth
45.0
Camera FoV
20.0
Look At Point
-4.2
7.5
0.0

D108B:Ex03 – WorldRT3A

- **Water surface design considerations:**
  - Is the water surface isostatic (at same height) ?
  - Are rivers present ?
  - Is the riverbed explicitly carved or simulated ?
  - Is the water bottom visible everywhere ?
  - Are there geometric waves ?
  - Are there textural wave breakers ?
  - Is the river flow texturally shown ?
- **These considerations drive:**
  - Water surface data structure
  - Generating procedures
  - Assembly set-up
  - Shader design

D108B:Ex03 – WorldRT3A

- **Isostatic surface implications**
  - Simple surfacing
  - Better suited to manual terrain carving
  - Needs single surface data item in vertex stream
  - Limits world visual richness
  - Faster (tiles can be stream-sourced)
- **Non-isostatic surface implications**
  - Considerable richness and data complexity
  - Better suited for simulated terrain
  - Tiles cannot be stream sourced
  - Tiles may have to be shaped (occluded)
- **Ex03 features non-isostatic river surfaces and isostatic ocean surface with geometric waves.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Principia river generation automation levels**
  - Fully physical simulation
  - Manually assisted riverbed carving
- **Principia manually assisted river carving levels**
  - Manual specification of river trace, everything else simulated using erosion.
  - Manual specification of river trace and profile parameters, procedural carving of riverbed.
  - Manual specification of river trace and exact riverbed shape, exact carving of riverbed.
  - Manual vs. automated waterfall generation.
- **The more automation, the longer the CPU time and the harder to control the final result.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water surface data streams**
  - Surface elevation (Ex03)
  - Local depth (Ex03)
  - Riverinity (Ex03, fraction of resemblence to river)
  - Logic value controlling tile shaping (Ex03, carving)
  - Wave UV coordinates (Ex03)
  - Geometric wave amplitude (not in Ex03)
  - Object placement sockets (not in Ex03)
- **Several Principia procedures take terrain data and generate water surface streams. Ex03 uses the simplest FX_Liquids1A in natural fill mode.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water surface procedural data generation. This data stream will be used for tile production.**

```
##define (S_RIVRTRACE) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_RiverTrace.png"
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )

##define (PROC_WATERGEN_EX03) as <PROCWRL_LIQUID_1A>
   #tag Method           =  (   PROC_LQGEN_WATERMAP )
   #tag inFcn.Elevation  =  (                F_ELEV )
   #tag inFcn.RivrDepth  =  (                  NONE )
   #tag inMap.RivrTrace  =  (           S_RIVRTRACE )

   #tag Com.OceanLevel   =  (                  0.15 ) 'Ocean level
   #tag Com.LandScale    =  (      25.2 ,      25.2 ) 'Land XY size
   #tag Com.SplineMap    =  (                   YES ) 'Spline-based UV mapping flag
   #tag Com.SplineDiv    =  (                    10 ) 'Spline-based UV mapping precision
   #tag Com.SplineItr    =  (                     3 ) 'Spline-based UV mapping precision

   #tag River.MinSize    =  (                    10 ) 'River size minimum feature
   #tag River.MinAper    =  (                     5 ) 'River node search span
   #tag River.MaxAper    =  (                     5 ) 'River node search span
   #tag River.MaxJoin    =  (                     9 ) 'River node sequencing distance
   #tag River.RiverEdge  =  (                  0.03 ) 'River trace threshold
   #tag River.RiverTerm  =  (            0xFF0000FF ) 'River trace ending color code
   #tag River.UMax       =  (                   8.0 ) 'River longitudinal texturing max
   #tag River.USpan      =  (                  24.0 ) 'River longitudinal texturing unit
   #tag River.VSpan      =  (                  12.0 ) 'River transverse texturing unit

   #tag Rsurf.UpLevel    =  (                  0.22 ) 'Water shaping control elevation
   #tag Rsurf.ExPixel    =  (                     2 ) 'Water shaping pixel extension
   #tag Rsurf.ProfSpan   =  (                    10 ) 'Water surface shape, profile analysis span
   #tag Rsurf.ProfFilter =  (                     0 ) 'Water surface shape, profile filter passes
   #tag Rsurf.ProfDnFilH =  (                 0.000 ) 'Water surface shape, geometry bottom uplift, hilevel
   #tag Rsurf.ProfUpFilH =  (                 0.035 ) 'Water surface shape, geometry top downlift, hilevel
   #tag Rsurf.ProfDnFilL =  (                -0.050 ) 'Water surface shape, geometry bottom uplift, lolevel
   #tag Rsurf.ProfUpFilL =  (                 0.050 ) 'Water surface shape, geometry top downlift, lolevel
   #tag Rsurf.ElevFilter =  (                     1 ) 'Water surface shape, elevation filter passes
   #tag Rsurf.SurfFilter =  (                     2 ) 'Water surface shape, surfaceability filter passes
   #tag Rsurf.ArcWatrH   =  (                  35.0 ) 'Water surface shape, arclength high level point
   #tag Rsurf.ArcWatrL   =  (                  15.0 ) 'Water surface shape, arclength low point
   #tag Rsurf.ArcTerm    =  (                  10.0 ) 'Water surface shape, arclength terminator
   #tag Rsurf.SlopeWatrH =  (                  15.0 ) 'Water surface shape, slope high level point
   #tag Rsurf.SlopeWatrL =  (                  25.0 ) 'Water surface shape, slope low point
   #tag Rsurf.SlopeTerm  =  (                  35.0 ) 'Water surface shape, slope terminator
   #tag Rsurf.TermTrig   =  (                -0.500 ) 'Water surface shape, termination trigger level
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water surface procedural data generation. This data stream will be used for tile production.**

```
. . .
   #tag Ocean.MinSize    =  (                  64 ) 'Ocean characterization, minimum size for real ocean
   #tag Ocean.ShoreLen   =  (                  64 ) 'Ocean characterization, minimum shore length for real ocean
   #tag Ocean.MinAper    =  (                   2 ) 'Ocean characterization, shore extraction aperture
   #tag Ocean.MaxAper    =  (                   6 ) 'Ocean characterization, shore extraction aperture
   #tag Ocean.BreakL     =  (                  32 ) 'Ocean characterization, shore extraction break length
   #tag Ocean.RiverExcl  =  (                   3 ) 'Ocean characterization, shore riverine exclusion distance
   #tag Ocean.RivMerge   =  (                10.0 ) 'Ocean characterization, shore-river merge grid units axial span
   #tag Ocean.RivFan     =  (                 4.0 ) 'Ocean characterization, shore-river merge grid units side span
   #tag ExDat.WaterLevel = "Files_Media/Ex03_ExMap_WaterLevel.dat"
   #tag ExDat.WaterDepth = "Files_Media/Ex03_ExMap_WaterDepth.dat"
   #tag ExDat.WaterRiver = "Files_Media/Ex03_ExMap_WaterRiver.dat"
   #tag ExDat.WaterLogic = "Files_Media/Ex03_ExMap_WaterLogic.dat"
   #tag ExDat.WaterFlowU = "Files_Media/Ex03_ExMap_WaterFlowU.dat"
   #tag ExDat.WaterFlowV = "Files_Media/Ex03_ExMap_WaterFlowV.dat"

##run (PROC_WATERGEN_EX03)

##label  (SKIP_WATERGEN)
##define (F_WATR_LEVEL) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag DMap      =  "Files_Media/Ex03_ExMap_WaterLevel.dat"
. . .

##define (F_WATR_DEPTH) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag DMap      =  "Files_Media/Ex03_ExMap_WaterDepth.dat"

##define (F_WATR_RIVER) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag DMap      =  "Files_Media/Ex03_ExMap_WaterRiver.dat"

##define (F_WATR_LOGIC) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); z=LMap2(x,y); r=Gt(z,0.0,1.0,0.0);  Ret(r);"
   #tag DMap      =  "Files_Media/Ex03_ExMap_WaterLogic.dat"

##define (F_WATR_UFLOW) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag DMap      =  "Files_Media/Ex03_ExMap_WaterFlowU.dat"

##define (F_WATR_VFLOW) as <FUNCTION_S>
   #tag Operation =  " Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag DMap      =  "Files_Media/Ex03_ExMap_WaterFlowV.dat"
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water vertex shader with small-scale wave TC production, wave generation and R-data stream.**

```
//************************************************************************//
//  Shader configuration constants: remember z is negative under water    //
//************************************************************************//
// Base and depth bias RGBA water properties
static float4  WaterLvl  = float4(  .15,  .15,  .15,  .15);
static float4  ClrZsrfc  = float4(  0.0,  0.3,  0.7,  0.6);
static float4  ClrZdeep  = float4(  0.0,  0.2,  0.8,  0.8);

// Wave speed, scale and motion coefficients
static float4  WavSScl   = float4( 0.0000025 , 0.0000025 , 8.0, 8.0);
static float4  WavMove   = float4( 1.1,  1.3 , -1.2, -1.4);
static float2  RivFlow   = float2(   0.00005 ,   0.00005 );

//************************************************************************//
//  Shader input and output structures                                    //
//************************************************************************//
struct VS_INPUT
{   // Inbound vertex buffer stream
    float3 vPosition        : POSITION    ;
    float2 vFactor          : BLENDWEIGHT ;// Depth and riverinity data
    float2 vTexCoord0       : TEXCOORD0   ;// World coordinate
    float2 vTexCoord1       : TEXCOORD1   ;// Natural coordinate
};

struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ;  // Clip space position
    float4 vDiffuse         : COLOR0   ;  // Diffuse color for water depth
    float2 vTexCoord0       : TEXCOORD0;  // Tex sampling coordinates
    float2 vTexCoord1       : TEXCOORD1;  // Tex sampling coordinates
    float2 vTexCoord2       : TEXCOORD2;  // Tex sampling coordinates
    float3 wLite            : TEXCOORD3;  // Light vector, world space
    float3 wView            : TEXCOORD4;  // View vector, world space
    float3 wOcean           : TEXCOORD5;  // Water scalar rendition factors
};

//************************************************************************//
//  Shader principal operations                                           //
//************************************************************************//
VS_OUTPUT VS_Ex02_LW1(const VS_INPUT In)
{   /* Output declaration and initialization */
    VS_OUTPUT Out     = (VS_OUTPUT) 0;
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water vertex shader with small-scale wave TC production, wave generation and R-data stream.**

```
   /* Position output, clip space with macro wave animation */
    mPos              = float4(In.vPosition.xyz, 1.0);
    xWave.z           = 0.008*sin(0.0012*vTime.x + 5.0*mPos.x) + 0.004*sin(0.0007*vTime.x - 7.0*mPos.y);
    mPos.z           += xWave.z*(1.0-In.vFactor.y);
    wPos              = mul(mPos, mWrld);
    Out.vPosition     = mul(wPos, mViewProj);

    /*  Lite and view vectors in world space. The light vector is not really used here */
    tLite              = normalize(-vLDir);
    tView             = vEPos - wPos.xyz;
    tView             = normalize(tView);

    /* Scalar outputs of light, view and riverinity vectors */
    Out.wLite         = tLite;
    Out.wView         = tView;
    Out.wOcean        = float3(1.0-In.vFactor.y,1.0-In.vFactor.y, 1.0-In.vFactor.y) ;

    /* Diffuse color and transparency based on water depth */
    Out.vDiffuse      = In.vFactor.xxxx/WaterLvl;
    Out.vDiffuse      = ClrZsrfc + (ClrZdeep-ClrZsrfc)*Out.vDiffuse;
    Out.vDiffuse.a   *= Out.vDiffuse.a;

    /* Wave adjustment vector */
    xWave             = WavSScl.xxyy * vTime;
    xWave             = WavMove * xWave;
    xFlow             = RivFlow*vTime;

    /* Texture sampling coordinates for the moving wave bumpmap trains */
    Out.vTexCoord0.x  = xWave.x + WavSScl.z*In.vTexCoord0.x;
    Out.vTexCoord0.y  = xWave.y + WavSScl.z*In.vTexCoord0.y;
    Out.vTexCoord1.x  = xWave.z + WavSScl.w*In.vTexCoord0.x;
    Out.vTexCoord1.y  = xWave.w + WavSScl.w*In.vTexCoord0.y;
    Out.vTexCoord2.x  = frac( xWave.x + In.vTexCoord1.x + xFlow.x);
    Out.vTexCoord2.y  = frac( xWave.y + In.vTexCoord1.y );

    /* Output stream */
    return Out;
}
//*************************************************************************//
//  Shader principal operations                                            //
//*************************************************************************//
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water pixel shader. River flow is implemented using TC animation in the natural U-direction. Wave caustics is implemented using animated bump blending for rivers and oceans.**

```
ps.2.0
;------------------------------------------------------------------------;
;                                                                        ;
;   Per-pixel environment-reflective, fresnel-tinted, alpha-depth        ;
;   small-scale animated water rendition with river-ocean balance        ;
;   using translation-offset bumpmap blending for wave animation.        ;
;                                                                        ;
;------------------------------------------------------------------------;
def c0,  2.0, -1.0,  0.5,  1.0    ; auxiliary constants
dcl      v0                       ; water base color + depth alpha
dcl      t0                       ; water bump texture, packed
dcl      t1                       ; water bump texture, packed
dcl      t2                       ; water bump texture, packed
dcl      t3                       ; lite vector, unpacked
dcl      t4                       ; view vector, unpacked
dcl      t5                       ; riverinity data, unpacked
dcl_2d   s0                       ; sampler, water bump texture
dcl_2d   s1                       ; sampler, water bump texture
dcl_2d   s2                       ; sampler, water bump texture
dcl_cube s3                       ; sampler, environment map
dcl_2d   s4                       ; sampler, fresnel map

;------------------------------------------------------------------------;
;   Wave bumpmap data load                                               ;
;------------------------------------------------------------------------;
texld    r0, t0, s0               ; water bump texture offset 0 load
texld    r1, t1, s1               ; water bump texture offset 1 load
texld    r2, t2, s2               ; water bump texture offset 2 load
lrp      r1, t5, r1, r2           ; water ocean-river balancing

;------------------------------------------------------------------------;
;   Normal calculation and scaling                                       ;
;------------------------------------------------------------------------;
mad      r0, r0, c0.r, c0.g       ; unpack normal map 0
mad      r1, r1, c0.r, c0.g       ; unpack normal map 1
add      r0, r0, r1               ; map blending
mul      r0, r0, c0.b             ; map blending, r0=(N)
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Water pixel shader. The blend between rivers and ocean is done upfront at the N-blend level. The water coloration derives from Fresnel-modulated spherical environment map sampling.**

```
;-------------------------------------------------------------------------;
;   Environment reflection and Fresnel sampling vectors                   ;
;-------------------------------------------------------------------------;
dp3      r3.rgb,  r0, t4          ;(N.V)   term
add      r4.rgb,  r3, c0.g        ;(N.V-1) term
add      r3.rgb,  r3, r3          ;(2N.V)  term
mad      r3.rgb,  r0, r3, -t4     ;r3=(R)=2N(V.N)-V
mul_sat  r4.rgb,  r4, r4          ;r4=(1-N.V)^2 Fresnel term

;-------------------------------------------------------------------------;
;   Environment reflection and Fresnel map sampling                       ;
;-------------------------------------------------------------------------;
texld    r3, r3, s3               ; Environment cube map lookup
texld    r4, r4, s4               ; Fresnel color term lookup

;-------------------------------------------------------------------------;
;   Final color and alpha generation                                      ;
;-------------------------------------------------------------------------;
mad      r0.rgb,  r3  , r4 , v0   ; Color = Map x Fresnel + Depth Base
mov      r0.a  , v0.a ;           ; Alpha = Depth Alpha
mov      oC0   , r0               ; Output

;-------------------------------------------------------------------------;
;   Shader end                                                            ;
;-------------------------------------------------------------------------;
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

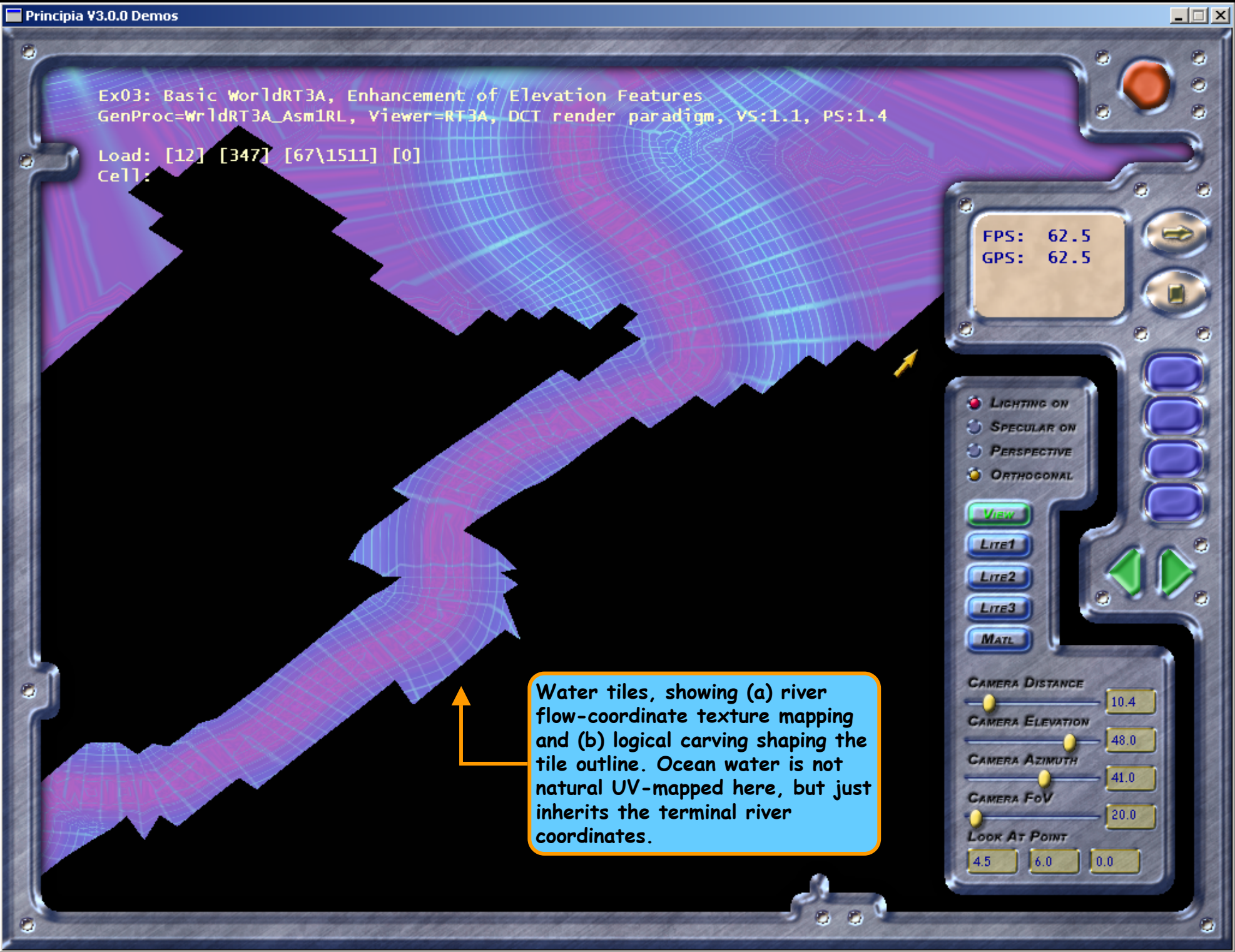
D108B:Ex03 – WorldRT3A

- **Water material set-up. It is important to enable wrapping for the environment texture.**

```
#tag SA        =  (              SS_ADDRESSU ,        SA_WRAP ,   3 ) 'Cube map must wrap!
#tag SA        =  (              SS_ADDRESSV ,        SA_WRAP ,   3 ) 'Cube map must wrap!
#tag SA        =  (              SS_ADDRESSU ,      SA_CLAMP ,   4 )
#tag SA        =  (              SS_ADDRESSV ,      SA_CLAMP ,   4 )
#tag TX        =  (         SW_BMAPWATER , 0 )
#tag TX        =  (         SW_BMAPWATER , 1 )
#tag TX        =  (         SW_BMAPWATER , 2 )
#tag TX        =  (       SW_ENVMAPWATER , 3 )
#tag TX        =  (      SW_FRESMAPWATER , 4 )
#tag VShader   =  (       VS_LAYER_WATR_EX03 )
#tag PShader   =  (       PS_LAYER_WATR_EX03 )
#tag Material  =  (       MSHD_STDCAMLIT , - )
```

- **It is remarkable that the shader that renders animated flowing rivers merging with animated oceans is a minor derivation of the fundamental bump-wave shader introduced in Chapter 1.04.F!**
- **The art of rendering water is really in the procedural generation of vertex reference data.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [12] [347] [67\1511] [0]
Cell:
FPS: 62.5
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
10.4
Camera Elevation
48.0
Camera Azimuth
41.0
Camera FoV
20.0
Look At Point
4.5
6.0
0.0
Water tiles, showing (a) river flow-coordinate texture mapping and (b) logical carving shaping the tile outline. Ocean water is not natural UV-mapped here, but just inherits the terminal river coordinates.


Principia V3.0.0 Demos
The FX_Liquid1A procedure, used to generate and map water surface properties, distinguishes between rivers and oceans. Here, the water shader is modified to display the color-coded "riverinity" coefficient, which determines to what extent the water behaves like a river or an ocean.
eatures
paradigm, VS:1.1, PS:1.4
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
5.5
Camera Elevation
59.0
Camera Azimuth
-97.0
Camera FoV
20.0
Look At Point
4.2
5.8
0.0

D108B:Ex03 – WorldRT3A

Tile texture palletization

- Consists in using multiple randomly-distributed isomorphs of textures while rendering a layer.
- The ASM_RT3A procedure implements palletization by scattering the tiles from a given terrain layer into multiple sub-layers, each having a separate material from the palette. The original layer is not rendered.

```
IN WORLD:

   #tag Layer_Decl    =  (   L_TILES_WRK ,    "TILES_WORKSET" ,                - ,          - , 1024 )
   #tag Layer_Decl    =  (   L_TILES_G1N ,  "TILES_GROUND_P1" ,  M_LAYER_GRNDPAL1 ,          - , 1024 )
   #tag Layer_Decl    =  (   L_TILES_G2N ,  "TILES_GROUND_P2" ,  M_LAYER_GRNDPAL2 ,          - , 1024 )
   . . .
   #tag Layer_Attr    =  (   L_TILES_WRK ,      LWT_HIDE ,           NONE ,         NONE )
   #tag Layer_Attr    =  (   L_TILES_G1N ,          NONE ,  LWO_VISITEST ,         NONE )
   #tag Layer_Attr    =  (   L_TILES_G2N ,          NONE ,  LWO_VISITEST ,         NONE )
   #tag Layer_Attr    =  (   L_TILES_W1N ,          NONE ,  LWO_VISITEST ,         NONE )
```

```
IN ASSEMBLER:

   #tag Wrld_Layer_Scatter =  (            L_TILES_WRK ,           L_TILES_G1N ,      NONE )
   #tag Wrld_Layer_Scatter =  (            L_TILES_WRK ,           L_TILES_G2N ,      NONE )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Palletized material set-up for DCT-blend bump-mapped composited texture base terrain. Note the constant mapping to shader registers and the TC index addressing map.**

```
   #tag RS        =  (       RS_ALPHABLENDENABLE ,              0 )
   #tag RS        =  (             RS_SRCBLEND ,     FA_SRCALPHA )
   #tag RS        =  (            RS_DESTBLEND ,  FA_INVSRCALPHA )
   #tag SA        =  (            SS_MAGFILTER ,       SA_LINEAR ,   0 ) ' Linear sample DCT for smooth blend
   #tag SA        =  (            SS_MINFILTER ,       SA_LINEAR ,   0 ) ' Linear sample DCT for smooth blend
   #tag SA        =  (            SS_MIPFILTER ,       SA_LINEAR ,   0 ) ' Linear sample DCT for smooth blend
   #tag SA        =  (            SS_MAGFILTER ,       SA_NONE   ,   1 ) ' No sampling of base texture
   . . .
   #tag SA        =  (            SS_MIPFILTER ,       SA_NONE   ,   7 ) ' No sampling of base texture
   #tag TS        =  (          TS_TEXCOORDINDEX ,              0 ,   0 )
   . .
   #tag TS        =  (          TS_TEXCOORDINDEX ,              7 ,   7 )
   #tag VShader   =  (        VS_LAYER_GRND_EX03 )
   #tag PShader   =  (        PS_LAYER_GRND_EX03 )
   #tag Material  =  (      MSHD_STDCAMLIT , - )
   #tag Material  =  (     MSHD_STDPERPIXEL , - )

##define (M_LAYER_GRNDPAL1) as <MATERIAL_3A>
   #tag Material  =  (   M_LAYER_GRNDCOMMON , - )
   #tag TX        =  (         S_DCT_GROUND , 0 )  ' Terrain control texture
   #tag TX        =  (            SG_ROCK , 1 )  ' Terrain base texture G
   #tag TX        =  (           SG_LAND_A , 2 )  ' Terrain base texture R+B
   #tag TX        =  (            SG_SAND , 3 )  ' Terrain base texture A
   #tag TX        =  (            SB_ROCK , 6 )  ' Terrain bump texture G
   #tag TX        =  (           SB_LAND_A , 7 )  ' Terrain bump texture R+B

##define (M_LAYER_GRNDPAL2) as <MATERIAL_3A>
   #tag Material  =  (   M_LAYER_GRNDCOMMON , - )
   #tag TX        =  (         S_DCT_GROUND , 0 )  ' Terrain control texture
   #tag TX        =  (            SG_ROCK , 1 )  ' Terrain base texture G
   #tag TX        =  (           SG_LAND_B , 2 )  ' Terrain base texture R+B
   #tag TX        =  (            SG_SAND , 3 )  ' Terrain base texture A
   #tag TX        =  (            SB_ROCK , 6 )  ' Terrain bump texture G
   #tag TX        =  (           SB_LAND_B , 7 )  ' Terrain bump texture R+B
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [5] [444] [97\6695] [0]
Cell: [334,178, 0.220]
FPS: 32.3
GPS: 62.5
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
51.0
Camera Azimuth
45.0
Camera FoV
If a single composite texture is tiled across the terrain, the UV random alternation method of Ex02 may not be enough to avoid repetitive patterning. This can be solved by using texture palletization during the tile assembly process.


Principia V3.0.0 Demos
Ex03: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [6] [504] [214\8203] [0]
Cell: [198,255, 0.254]
FPS: 32.3
GPS: 31.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
9.1
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
This kluged render shows the macro-scale wave trains created in the vertex shader, as well the two-tone land palletization achieved by scattering land base tiles between two layers with different material palettes.

D108B:Ex03 – WorldRT3A

- **World assembly procedures:**
  - Generate tile geometry for different layers by converting elevation maps, carving maps and specialized channel data maps to vertex data.
  - Generate accessory world data structures such as type grids or visibility structures.
  - Place predefined objects in the world object layers and validate the overall world data contents.
- **No procedural generation per se in assembler:**
  - This is done by a suite of specialized map generation procedures for different layer types (land, water…)
  - This enables the same assembly procedure to generate a wide variety of world designs, but requires a well-thought and complex process.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Putting it all together: Input data maps, generated from elemental terrain procedures.**

```
##define (S_ELEV) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_Elevation.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (F_ELEV) as <FUNCTION_S>
   #tag Operation =  "Arg(x,y); z=LMap2(x,y); Ret(z);"
   #tag ZMap      =  (           S_ELEV )
   #tag ZMin      =  (            0.000 )
   #tag ZMax      =  (            2.200 )

##define (S_WRAPU) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_WrapU.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (S_WRAPV) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_WrapV.png"
   #tag Definer   =  (        DEF_16BIT )
   #tag LoadNow   =  (                1 )

##define (S_DCT_GROUND) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_DCTGround_C.bmp"
   #tag Alpha     =  "Files_Media/Ex03_InMap_DCTGround_X.bmp"
   #tag Definer   =  (       DEF_TEX )
   #tag LoadNow   =  (             1 )

##define (S_DCT_WATER) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_DCTWater_C.bmp"
   #tag Alpha     =  "Files_Media/Ex03_InMap_DCTWater_X.bmp"
   #tag Definer   =  (       DEF_TEX )
   #tag LoadNow   =  (             1 )

##define (S_RIVRTRACE) as <SURFACE>
   #tag File      =  "Files_Media/Ex03_InMap_RiverTrace.png"
   #tag Definer   =  (     DEF_TEX )
   #tag LoadNow   =  (           1 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

Putting it all together: Tile vertex structures.

```
##define (VTD_LAYER_GRND) as <DC_GVTXDEF3>
   #tag IsFVF     =  (            NO )
   #tag IsPixl    =  (            NO )
   #tag IsNorm    =  (           YES )
   #tag IsDiff    =  (            NO )
   #tag IsSpec    =  (            NO )
   #tag IsTexN    =  (             3 )
   #tag IsSkin    =  (             0 )
   #tag IsBndx    =  (            NO )
   #tag IsSprt    =  (            NO )
   #tag Memory    =  (   GMEM_DEVICE )
   #tag IndexFmt  =  (       INDEX16 )
   #tag IsDynamic =  (             - )
   #tag IsWrtonly =  (             - )

##define (VTD_LAYER_WATR) as <DC_GVTXDEF3>
   #tag IsFVF     =  (            NO )
   #tag IsPixl    =  (            NO )
   #tag IsNorm    =  (            NO )
   #tag IsDiff    =  (            NO )
   #tag IsSpec    =  (            NO )
   #tag IsTexN    =  (             2 )
   #tag IsSkin    =  (             2 )
   #tag IsBndx    =  (            NO )
   #tag IsSprt    =  (            NO )
   #tag Memory    =  (   GMEM_DEVICE )
   #tag IndexFmt  =  (       INDEX16 )
   #tag IsDynamic =  (             - )
   #tag IsWrtonly =  (             - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

New ground tile layer structure:

- Tiles are collected and operated on (partition, visitest) in a working layer that is not renderable.
- The same tiles are also placed in two renderable but non-operated layers that focus on material mgmt.

```
##define (OP_VISITEST) as <OP_VISITEST_3R>
   #tag Method    = (          NONE )
   #tag TstObjVS  = (            NO )
   #tag SetObjVS  = (            NO )

##define (OP_PARTFIX) as <OP_PARTSECT_2R>
   #tag Frequency = (          NONE )

##define (L_TILES_GWK) as <WORLDLAYER_3R>
   #tag Name      = "Ground tiles, holding/working layer"
   #tag Type      = (  LWT_NORENDER )
   #tag MatInit   = (          NONE )
   #tag MatTerm   = (          NONE )
   #tag Operation = (    OP_PARTFIX )
   #tag Operation = (   OP_VISITEST )
   #tag Count     = (          1024 )

##define (L_TILES_GD1) as <WORLDLAYER_3R>
   #tag Name      = "Ground tiles, display layer for texture palette A"
   #tag Type      = (      LWT_TILE )
   #tag Type      = (    LWT_PARTED )
   #tag MatInit   = ( M_LAYER_GPAL1 )
   #tag MatTerm   = (          NONE )
   #tag Count     = (          1024 )

##define (L_TILES_GD2) as <WORLDLAYER_3R>
   #tag Name      = "Ground tiles, display layer for texture palette B"
   #tag Type      = (      LWT_TILE )
   #tag Type      = (    LWT_PARTED )
   #tag MatInit   = ( M_LAYER_GPAL2 )
   #tag MatTerm   = (          NONE )
   #tag Count     = (          1024 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

Putting it all together: World assembly

```
##define (L_GRID) as <WORLDLAYER_3R>
   #tag Name      = "Water tiles, holding and display layer"
   #tag Type      = (      LWT_HIDE )
   #tag Type      = (      LWT_TILE )
   #tag Type      = (    LWT_PARTED )
   #tag Link      = (   L_TILES_WTR )
   #tag MatInit   = (  M_LAYER_GRID )
   #tag MatTerm   = (           NONE )
   #tag Count     = (              8 )

. . .

##define (WRLD_EX03) as <WORLD_RT3A>

   #tag Sector_Mode   =  (  WPART_RECT2D )
   #tag Sector_Dim    =  (  12 ,  12 , 1 ) ' Should be tile multiple
   #tag Sector_Band   =  (          0.01 ) ' Sector safety guardband

   #tag Layer_Decl    =  (       L_PROPS ,            "PROPS" ,                - ,  M_STDNULL ,    8 )
   #tag Layer_Decl    =  (   L_TILES_INI ,       "TILES_INIT" ,                - ,           - ,    4 )
   #tag Layer_Decl    =  (   L_TILES_GWK ,               NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_GD1 ,               NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_GD2 ,               NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (   L_TILES_WTR ,               NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (        L_GRID ,               NONE ,   NONE ,     NONE )
   #tag Layer_Decl    =  (       L_WCURS ,          "WCURSOR" ,                - ,  M_STDNULL ,    4 )

   #tag Layer_Item    =  (       L_PROPS ,      O_AXISYS )
   #tag Layer_Item    =  (   L_TILES_INI ,     O_TILEKNX )
```

We changed how tiles are processed in the layer definition script – no code. Now you see the power of Principia !

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

Putting it all together: Assembly procedure.

```
##define (PROC_WGEN_EX03) as <PROCWRL_ASM_RT3A>

   #tag Tile_Layers        =  (                                2 )
   #tag Tile_Numbers       =  (  0 ,          72 ,           72 ) 'Tile size, count
   #tag Tile_SideSize      =  (  0 ,        0.35 ,         0.35 ) 'Tile size, scale per
   #tag Tile_Vertices      =  (  0 ,           8 ,            8 ) 'Tile size, vertices per
   #tag Tile_Definer       =  (  0 ,             VTD_LAYER_GRND ) 'Tile vertex stream definer
   #tag Tile_Kinex         =  (  0 ,        NONE ,            - ) 'Tile per kinex prototype
   #tag Tile_ElevationF    =  (  0 ,                     F_ELEV ) 'Tile elevation function
   #tag Tile_CarvingF      =  (  0 ,                       NONE ) 'Tile structural fill function
   #tag Tile_Tri_Method    =  (  0 ,           MESHTRIANG_MNDN ) 'Tile triangulate flat: across lowest gradient
   #tag Tile_UV0_Method    =  (  0 ,         PROC_UVSYS_RNDTILE ) 'Tile coordinate: intrinsic
   #tag Tile_UV1_Method    =  (  0 ,          PROC_UVSYS_IMPORT ) 'Tile coordinate: natural terrain import
   #tag Tile_UV2_Method    =  (  0 ,           PROC_UVSYS_WORLD ) 'Tile coordinate: world DCT
   #tag Tile_UV0_Source    =  (  0 ,           - ,              - ) 'Tile coordinate data source
   #tag Tile_UV1_Source    =  (  0 ,     S_WRAPU ,      S_WRAPV ) 'Tile coordinate data source
   #tag Tile_UV2_Source    =  (  0 ,           - ,              - ) 'Tile coordinate data source
   #tag Tile_ChannelXN     =  (  0 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelYN     =  (  0 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelZN     =  (  0 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelW1     =  (  0 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelW2     =  (  0 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelW3     =  (  0 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_BoxOverride   =  (  0 ,        0.00 ,         0.00 ) 'Use natural Z given water design
   #tag Tile_RimLineGen    =  (  0 ,           1 ,            1 ) 'Rim data structure extension flag

   #tag Tile_Numbers       =  (  1 ,          72 ,           72 ) 'Tile size, count
   #tag Tile_SideSize      =  (  1 ,        0.35 ,         0.35 ) 'Tile size, count
   #tag Tile_Vertices      =  (  1 ,           6 ,            6 ) 'Tile size, vertices per
   #tag Tile_Definer       =  (  1 ,             VTD_LAYER_WATR ) 'Tile vertex stream definer
   #tag Tile_Kinex         =  (  1 ,        NONE ,            - ) 'Tile per kinex prototype
   #tag Tile_ElevationF    =  (  1 ,                F_WATR_LEVEL ) 'Tile elevation function
   #tag Tile_CarvingF      =  (  1 ,                F_WATR_LOGIC ) 'Tile structural fill function
   #tag Tile_Tri_Method    =  (  1 ,                           - ) 'Tile triangulate flat: default
   #tag Tile_UV0_Method    =  (  1 ,           PROC_UVSYS_WORLD ) 'Tile coordinate: intrinsic
   #tag Tile_UV1_Method    =  (  1 ,          PROC_UVSYS_IMPORT ) 'Tile coordinate: world DCT
   #tag Tile_UV0_Source    =  (  1 ,           - ,              - ) 'Tile coordinate data source
   #tag Tile_UV1_Source    =  (  1 , F_WATR_UFLOW , F_WATR_VFLOW ) 'Tile coordinate data source
   #tag Tile_ChannelXN     =  (  1 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelYN     =  (  1 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelZN     =  (  1 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_ChannelW1     =  (  1 ,                F_WATR_DEPTH ) 'Tile scalar vertex data override
   #tag Tile_ChannelW2     =  (  1 ,                F_WATR_RIVER ) 'Tile scalar vertex data override
   #tag Tile_ChannelW3     =  (  1 ,                       NONE ) 'Tile scalar vertex data override
   #tag Tile_BoxOverride   =  (  1 ,        0.00 ,         0.00 ) 'Use natural Z given water design
   #tag Tile_RimLineGem    =  (  1 ,           1 ,            1 ) 'Rim data structure extension flag
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Putting it all together: Assembly procedure with specific enablement of:**
  - Water surface data channels
  - Palletization scatter of tiles
  - Rim visibility management structure generation

```
#tag Grid_TileSource    =  (                        0 )
#tag Grid_ElevationF    =  (        F_GRIDHEIGHT_EX03 )
#tag Grid_TypeExport    =  (             S_WTYPE_EX03 )
#tag Grid_TypeImport    =  (                     NONE )
#tag Grid_TypeAsmMap    =  (             S_DCT_GROUND ,   TC_CLIFF ,  TC_CLIFF ,  TC_FLATS ,          - )
#tag Grid_TypeAsmMap    =  (              S_DCT_WATER ,   TC_WATER ,         - ,         - ,          - )

@@@@ Wrld_Layer_Map     =  (  TileIndex+++WrldLayerID ,      Dct+DctTileViewableEvaluationFlag+EvalCode ,
   CoverDct+CoverViewableEvaluationCodeForTile )
#tag Wrld_Layer_Map     =  (  0 ,         L_TILES_WRK ,           S_DCT_GROUND ,        NO , 0xFFFFFFFF ,                  NONE
   , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Map     =  (  1 ,         L_TILES_W1N ,            S_DCT_WATER ,        NO , 0xFFFFFFFF ,                  NONE
   , NONE , NONE ,  0x00FFFFFF )
#tag Wrld_Layer_Scatter =  (              L_TILES_WRK ,            L_TILES_G1N ,      NONE )
#tag Wrld_Layer_Scatter =  (              L_TILES_WRK ,            L_TILES_G2N ,      NONE )
```

- **By adding the LoD and layering concepts introduced in Ex03, we can have a production grade terrain generated in minutes!**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex03 – WorldRT3A

- **Ex03 final assessment**
  - We have made a lot of progress in visual quality since Ex01, and introduced several fundamental techniques for third-person terrain generation.
  - These will be taken a step further in Chapter 2.XX which will show how to reach even more advanced stages in third-person world visual richness.
- **Advanced 3rdP terrain techniques (for later)**
  - Separable tiling of different terrain features
  - Unstructured tiles and meshes
  - Hierarchical detail generation
  - Reflections, refractions and athmospherics
  - Advanced lighting, texturing and LoD resolution

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Chapter 1:08
Core Worlds
Principia v3.0
Reference Series
(c) Western Star
Entertainment Ltd, 2003-2006


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1
Load: [3] [1945] [295\13633] [3][89][3]
Cell: [412,247, 0.245]
The most visible enhancement of our Ex04 world is that objects cast soft shadows, without noticeable penalty in frame rate.
Moreover, we have many diverse objects: 3D solids, billboards, and skinned animated characters with complex rendering.
FPS: 32.3
GPS: 66.7
CLK: 158.2
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
39.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
8.4
-0.4
0.6


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS
Load: [3] [1945] [296\13634] [3][90][3]
Cell: [423,258, 0.265]
Character object can be selected by clicking on them, and then directed to walk to another location by means of another click. Selected walking characters will display their path (with automatic way-stations), which curve to avoid obstacles.
FPS: 62.5
GPS: 32.3
CLK: 274.2
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
39.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
8.4
-0.4
0.6


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1,
Load: [3] [1945] [296\13634] [3][89][3]
Cell: [410,218, 0.211]
The cursor is composed of different elements that change based on context. For instance, when a character is selected, the cursor "exact destination" arrow turns red.
WorldRT3A cursors can be configured to point at the center of cells (as here) or at terrain vertices. The choice depends on the gameplay logic you wish to have.
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
39.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
8.4
-0.4
0.6


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [3] [2176] [108\13634] [0][79][0]
Cell: [344,180, 0.217]
FPS: 66.7
GPS: 62.5
CLK: 454.9
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Pressing the UNVA button activates a special world PLACER component, which enables the user to create objects by clicking on desired world location.
When the cursor is over a terrain type which enables object creation, the placer will display an object icon and change the cell cursor to indicate the necessary footprint. If the cursor is over an invalid terrain type, the placer icon will disappear.
If the footprint interferes with existing obstacles or an unacceptable type of terrain, the footprint will change appearance to indicate that the object cannot be placed there.


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [3] [2178] [110\13638] [2][85][1]
Cell: [355,189, 0.179]
FPS: 32.3
GPS: 62.5
CLK: 523.4
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
3.5
Camera Elevation
39.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
4.9
-2.9
0.6
Walking characters do not simply walk into each other. They will yield and adjust their paths dynamically to avoid that. Once they are clear of each other, the yielding character will resume its original route.


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [2] [1337] [276\13642] [1][82][0]
Cell: [184,275, 0.321]
FPS: 31.3
GPS: 25.0
CLK: 557.4
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
5.0
Camera Elevation
49.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
-3.1
1.1
0.6
The developer can select between many different path-finding methods that Principia offers at the time of world generation. In this variant of Ex04, we have enabled dynamic grid-cell based path-finding (as opposed to the faster but less flexible hierarchical domain used otherwise).

D108B:Ex04 – WorldRT3A

Objectives

- Introduce fundamental shadow rendering and character management techniques in our world.

Requirements and Implementation

- Objects cast shadows on world tiles (but not on each other or themselves, nor do tiles cast shadows).
- Mobile characters that can be interactively created, selected and moved by the user; and dynamically adjust motion to avoid collision.

Notes

- Ex04 introduces several new fundamentals, and shows how easy is to create a ***GAME ENGINE*** with Principia!

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Major new techniques introduced in Ex04:**
  - Fast and simple shadow mapping technique for approximate object-on-ground shadows.
  - Layer structuring to separate object computational operations from multi-pass rendering operations.
  - Layer structuring to manage and render different types of objects (billboards, solids, skinned chars).
  - Interactive world cursor and placer components, and their integration with interactive character objects.
  - Object path-finding and collision resolution.
- **Each of these is an important new addition to the fundamental techniques set from Ex01-03.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Ex04 topics roadmap (and these are basic!)**
  - Shadow mapping algorithm
  - Shadow mapping operations (Psm and VrtSetup)
  - Shadow mapping with billboards
  - Shadow mapping shader design
  - Shadow mapping refinements (guard bands...etc).
  - World layer construction, overall and operations
  - Character object design, animation and controls
  - Character user code, routing
  - Character user code, path display
  - Character user code, collision resolution
  - World cursor/viewer design
  - World placer/viewer design
  - World generation procedure with path-finding

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Shadow mapping for WorldRT3A:**
  - There are several different ways to implement shadow maps in WorldRT3A … but here, forget all that you saw on this subject in Chapters 3 and 4!
  - Simple shadow requirements for Ex04: objects cast shadows on terrain, but not on each other. There is no self-shadowing.
  - In this simple case, there is no need to track depth! Just render objects as solid white on shadow map as seen from light. When rendering terrain, sample shadow map. If (sample‡0x00), pixel is in shadow.
- **The resulting shadow map algorithm is very fast and provides decent visual results.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Shadow mapping algorithm in Ex04:**
  - In a first layer group, perform all computatonal operations on objects e.g. animation, partitioning, visitesting...etc.
  - Render objects in the world view. This helps early z-culling when rendering the world tiles later.
  - Clear shadow map (SM) and set up the light-view camera (LVC) so that all object visible in the world view fill the LVC view-port to its maximum extent.
  - Render all shadow casting objects on SM using LVC so as to just fill their silhouette with depth (no materials ...etc).
  - When rendering tiles, sample SM as usual. If there is non-zero depth in the SM, the tile pixel is in shadow.
  - This is much simpler and faster !!! Just be careful not to have objects below tile level (they will cast shadows).

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Shadow algorithm: more simplifications!

- If objects do not shadow each other, the only information needed when sampling the PSM is whether it contains a zero (no shadow) or non-zero (in shadow)!
- Therefore, when rendering the PSM, one can just fill the SM silhouettes with ones! This also allows the use of XRGB render targets (instead of FP32) for PS1X platforms.
- Alpha testing of billboard objects is much simpler when rendering on a XRGB format surface.
- If depth recording is still needed (for self-shadowing or for various filtering FX) one can still use the XRGB format and encode the depth using the techniques from Chapter 1.04.
- Performance/appearance can be optimized by balancing PSM size against multiple PSM sampling and averaging.
- The resulting PSM algorithm here is much faster and more robust than those seen in Chapter 1.04.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Shadow algorithm implementation: New tools!**
  - PsmSetup2A operation
  - VrtSetup2A operation
  - Billboard object orientation control variables
- **PsmSetup2A operation**
  - Automatically calculates all camera parameters for the view-from-light rendering onto the shadow map.
- **VrtSetup2A operation**
  - Sets up a new camera for rendering and application processing of camera-dependent calculations.
  - Assigns values to global Principia variables used to control individual component behavior.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
FPS: 31.3
GPS: 66.7
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
This diagnostic set-up shows the view from the shadow map light, as automatically calculated by the PsmSetup2A operation, which frees the user from the burden of calculating the parameters of shadow mapping light spaces and tracking what is visible or not. Only objects visible in the normal view are rendered in the shadow map.

D108B:Ex04 – WorldRT3A

- **Billboard object orientation controls**
  - Billboards are always oriented to face the camera. In Principia billboard objects such as <Billboard2F>, the orientation calculation may be done on the CPU and stored for later use depending on global control vars.
  - Using PSM shadows with billboards may require this orientation to be redone upon every view change.
  - <OrientExec> triggers a Billboard2F::Orient() calculation during *renders*. Set this to 0x0 if the orientation matrix has been already computed in an earlier layer to prevent unnecessary overhead when re-rendering object.
  - <OrientIndx> enables to choose between two distinct billboard orientation matrices A and B. In addition, <OrientIndx> is used to choose between different billboard shapes, enabling the shadow to have a distinct silhouette!

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [3] [1667] [330\11756] [0]
Cell: [54,151, 1.137]
FPS: 32.3
GPS: 25.0
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
-9.5
-6.0
0.0
The billboard/shadow can be rendered in different control sequences to achieve a particular billboard-to-shadow orientations, for different object types.
Here, the billboard is seen full-face-on from both the camera and the light standpoints. This is useful for full-body items (e.g. trees, etc) and is best sequenced as:
• World Camera Space Visitest (OrientIndex=0)
• World Camera Space Render (Orient Index=0, Exec=0)
• Light PSM Space Render (Orient Index=1, Exec=1)

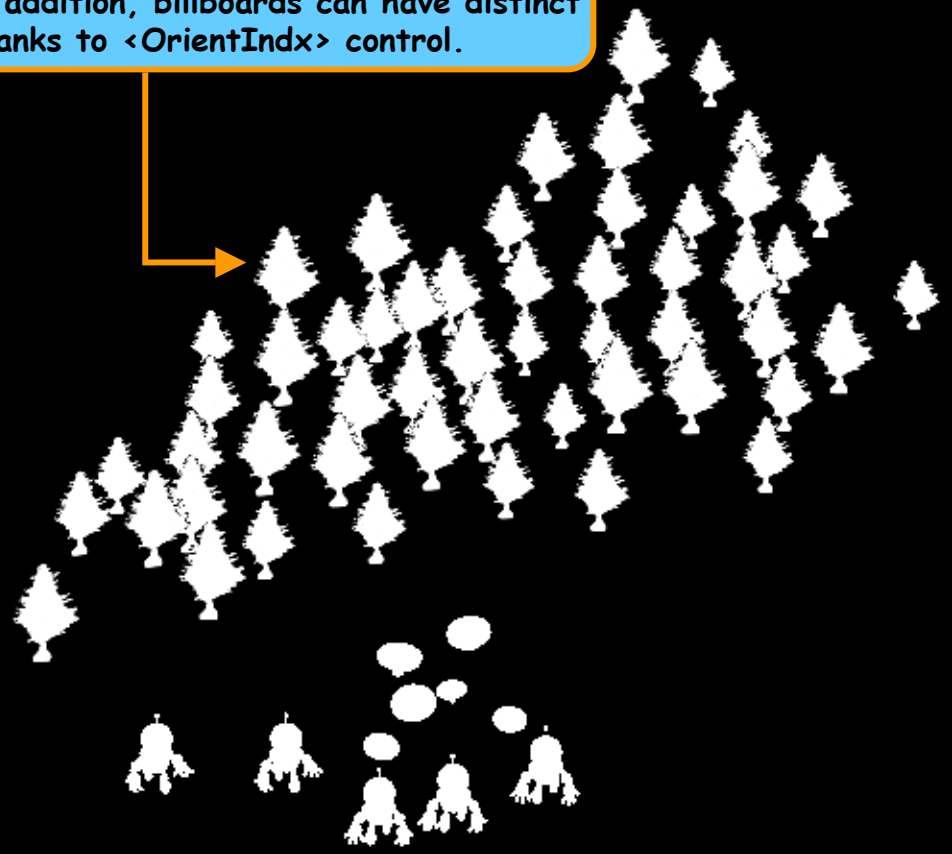

Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [3] [1667] [330\11756] [0]
Cell: [78,154, 1.097]
FPS: 62.5
GPS: 66.7
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
6.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
-9.5
-6.0
0.0
A simple change in the control sequence makes the billboards render in world and PSM space under the same orientation, thereby resulting in a thin-panel look:
• World Camera Space Visitest (OrientIndex=0)
• World Camera Space Render (Orient Index=0, Exec=0)
• Light PSM Space Render (Orient Index=0, Exec=0)

Principia V3.0.0 Demos

Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4

The terrain-only shadow map algorithm in Ex04 enables to use a simple RGB map, with simple 0-1 in-shadow state. This leads to significant speed gains.

As this diagnostic view of the PSM shows, casting objects are rendered in pure white silhouette as seen from the light. In addition, billboards can have distinct shadow outlines thanks to <OrientIndx> control.



FPS: 32.3
GPS: 62.5

Lighting on
Specular on
Perspective
Orthogonal

View
Lite1
Lite2
Lite3
Matl

Camera Distance 6.0
Camera Elevation 25.0
Camera Azimuth 45.0
Camera FoV 25.0
Look At Point -2.2 2.2 0.6

D108B:Ex04 – WorldRT3A

- **Billboard object construction for fixed function rendering with orientation-specific control.**

```
@@This variable determines whether the Bbd::Orient() function will be called before renders
##define <SYS_VARIABLE>   =  (   V_ORIENTEXE ,   VARTYPE_INT  ,          1 )
@@This variable determines which orientation alternative to use when rendering/visitesting
##define <SYS_VARIABLE>   =  (   V_ORIENTIDX ,   VARTYPE_INT  ,          0 )


##define (G_TREEPINE) as <VERTEXSET_3A>
   #tag File      =  "Files_Media\Com_Mesh_TreePineN_BillboardXY.vsf"
   #tag Definer   =  ( VTD_FVF_FILE )

##define (S_OBJ_TREEPINE_WRL) as <SURFACE>
   #tag File      =  "..\Demo_Common\Files_Media\Objects\Tex_WorldObject_TreePine_C.bmp"
   #tag Alpha     =  "..\Demo_Common\Files_Media\Objects\Tex_WorldObject_TreePine_X.bmp"
   #tag Definer   =  (       DEF_TEX )

##define (S_OBJ_TREEPINE_PSM) as <SURFACE>
   #tag File      =  "..\Demo_Common\Files_Media\Objects\Tex_WorldObject_TreePine_X[psm].bmp"
   #tag Alpha     =  "..\Demo_Common\Files_Media\Objects\Tex_WorldObject_TreePine_X[psm].bmp"
   #tag Definer   =  (       DEF_TEX )

##define (MX_TREEPINE_WRL) as <MATERIAL_3A>
   #tag TX        =  ( S_OBJ_TREEPINE_WRL , 0 )

##define (MX_TREEPINE_PSM) as <MATERIAL_3A>
   #tag TX        =  ( S_OBJ_TREEPINE_PSM , 0 )

##define (R_TREEPINE_WRL) as <MESH_3A>
   #tag Component =  (        0 ,    MX_TREEPINE_WRL ,  G_TREEPINE )

##define (R_TREEPINE_PSM) as <MESH_3A>
   #tag Component =  (        0 ,    MX_TREEPINE_PSM ,  G_TREEPINE )

##define (O_TREEPINE) as <BILLBOARD_2F>
   #tag Global    =  (          NO )
   #tag FixSize   =  (         YES )
   #tag BbdAxis   =  (  0.000 ,   0.000 ,   1.000 )
   #tag OrientExec =  (  V_ORIENTEXE ) 'will be set by a layer operation
   #tag OrientIndx =  (  V_ORIENTIDX ) 'will be set by a layer operation
   #tag Position  =  (   "Decl(M);" )
   #tag Facets    =  (   0 ,  R_TREEPINE_WRL ,    NONE ) 'render for normal world view
   #tag Facets    =  (   1 ,  R_TREEPINE_PSM ,    NONE ) 'render for psm bw view
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Rendering implementation highlights

- Three types of objects populate the world of Ex04: CPU-set billboards, solid 3D objects and skinned 3D characters. Each of these uses different rendering designs.
- World tiles are rendered as per Ex03, with per-vertex illumination, natural UV maps for slopes and water flow animation, pre-variegated DCT mapping of tile textures with UV flipping...etc.
- Shaders for rendering on the PSM can be greatly simplified. In fact, the world can be rendered on PS1.4 hardware if shadow smoothing is sacrificed.
- Several acceleration avenues have not been explored here, namely (a) internal computation of TC data for world tiles, (b) geometry phasing for PSM renders, and (c) one-pass MRT object renders.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World tiles, vertex shader structure

- Per vertex lighting, natural UV systems for slopes and animated water, light-view clip space output.

```
//*************************************************************************//
//  Shader data registers, variables and static constants                  //
//*************************************************************************//
##include "..\Demo_Common\Files_Media\Shaders\VS_Std_Includes.shd"
float4x4 mLVP      : register (c56);  // Api: world to camera projective space transform

float4   wPos      ;                  // Wrk: Position vector, world space
float3   wNorm     ;                  // Wrk: Normal vector, world space
float3   wLite     ;                  // Wrk: Light incidence vector, world space
float3   wView     ;                  // Wrk: View vector, world space
float3   wHalf     ;                  // Wrk: Halfway vector, world space
float3   LdotN     ;                  // Wrk: Scalar product
float3   HdotN     ;                  // Wrk: Scalar product

//*************************************************************************//
//   Shader input and output structures                                    //
//*************************************************************************//
struct VS_INPUT
{   // Inbound vertex buffer stream
    float3 vPosition        : POSITION ; // Tile vertex real position
    float3 vNormal          : NORMAL   ; // Tile normal vector
    float2 vTexCoord0       : TEXCOORD0; // Tile TC tile-internal system
    float2 vTexCoord1       : TEXCOORD1; // Tile TC natural-UV system
    float2 vTexCoord2       : TEXCOORD2; // Tile TC world-DCT system [XS]
};

struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ; // Transformed position
    float4 vDiffuse         : COLOR0   ; // Diffuse color [XS]
    float4 vSpecular        : COLOR1   ; // Speclar color [XS]
    float3 vTexCoordW       : TEXCOORD0; // TC world DCT system (xy+n.w)
    float4 vPosLps          : TEXCOORD1; // TC light projective space
    float2 vTexCoordA       : TEXCOORD2; // TC tile internal system (natUV)
    float2 vTexCoordB       : TEXCOORD3; // TC tile internal system (tile, [XS])
    float2 vTexCoordC       : TEXCOORD4; // TC tile internal system (tile, [XS])
};
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World tiles, vertex shader operations

- Note light-view clip space position output.

```
//--------------------------------------------------------------------------//
//   Shader principal operations                                            //
//--------------------------------------------------------------------------//
VS_OUTPUT VS_Ex04_GroundWithShadows(const VS_INPUT In) { VS_OUTPUT Out = (VS_OUTPUT) 0;

    /* Position world-view-projection transform and position output [XS] */
    wPos             = mul(float4(In.vPosition,1.0f), mWrld);
    Out.vPosition    = mul(wPos, mViewProj);

    /* Normals world transform [XS] */
    wNorm            = mul(In.vNormal, (float3x3) mWrld);
    wNorm            = normalize(wNorm);

    /* Common light, view and half vectors in world space */
    wLite            = normalize( -vLDir           );
    wView            = normalize( vEPos - wPos.xyz  );
    wHalf            = normalize( 0.5*(wLite+wView) );
    LdotN            = saturate ( dot(wNorm, wLite) );
    HdotN            = saturate ( dot(wNorm, wHalf) );

    /* Lambert diffuse and Blinn specular per-vertex color output */
    Out.vDiffuse     = float4( vLDiff*LdotN + vLAmbi , 1.0 );
    Out.vSpecular    = float4( vLSpec*pow(HdotN, vMPwr), 1.0 );

    /* DCT and shadow map blend TC output */
    Out.vTexCoordW.xy = In.vTexCoord2.xy;                          //World DCT system
    Out.vTexCoordW.z  = LdotN;                                     //World shadow map facing index
    Out.vTexCoordA.x  = 6.0*In.vTexCoord1.x+0.020*In.vPosition.x;  //Natural UV system, wrapped cliff features
    Out.vTexCoordA.y  = 1.0*In.vTexCoord1.y;                       //Natural UV system, wrapped cliff features
    Out.vTexCoordB.xy = In.vTexCoord0.xy;                          //Tile-based system, flatland
    Out.vTexCoordC.xy = In.vTexCoord0.xy;                          //Tile-based system, flatland

    /* Vertex position in light projection space */
    Out.vPosLps      = mul(wPos, mLVP);

return Out;}
//--------------------------------------------------------------------------//
//   Shader principal operations                                            //
//--------------------------------------------------------------------------//
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World tiles, pixel shader structure:

- Note the clamped point sampling of the PSM!

```
//*************************************************************************//
//  Shader application data, variables and config constants section        //
//*************************************************************************//
uniform float2    SmapTC0                           ;// Shadow map projective TC, raw
uniform float2    SmapTCB                           ;// Shadow map projective TC, biased
uniform float4    DepthLSM                          ;// Depth from shadow map // !!!RGB
uniform float1    Sfactor1                          ;// Shadowing factor
. . .
static  float2    SMSDELA = {0.00195313,  0.00260417} ;// Shadow multisample offsets
static  float2    SMSDELB = {0.00195313, -0.00260417} ;// Shadow multisample offsets
static  float1    SVALUE = 0.400                    ;// Shadow strength

//*************************************************************************//
//  Shader textures and samplers                                           //
//*************************************************************************//
sampler TS0 = sampler_state { // DCT texture sampler
  texture   = Tex0;
  mipfilter = LINEAR;
  . . .
  addressv  = WRAP;
};

sampler TS1 = sampler_state { // Shadow map depth sampler
  texture   = Tex1;
  mipfilter = POINT;
  . . .
  addressv  = CLAMP;
};

sampler TS2 = sampler_state { // Base color sampler
  texture   = Tex2;
  mipfilter = LINEAR;
  . . .
  addressv  = WRAP;
};

sampler TS3 = sampler_state { // Base color sampler
. . .
sampler TS4 = sampler_state { // Base color sampler
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **World tiles, pixel shader operations (with shadow map 4-tap smoothing)**

```
PS_OUTPUT PS_Ex04_GroundWithShadows(VS_OUTPUT In){  PS_OUTPUT Out = (PS_OUTPUT)0;

    /* Sample terrain DCT and tile textures */
    Cdct       = tex2D(TS0, In.vTexCoordW);
    C1         = tex2D(TS2, In.vTexCoordA);
    C2         = tex2D(TS3, In.vTexCoordB);
    C3         = tex2D(TS4, In.vTexCoordC);

    /* Blend color assembly according to DCT */
    C1          = C1*Cdct.gggg;
    C1         += C2*Cdct.rrrr;
    C1         += C2*Cdct.bbbb;
    C1         += C3*Cdct.aaaa;

    /* Pixel depth sampling from shadow map */
    SmapTC0   = 0.5*(In.vPosLps.xy / In.vPosLps.w) + float2(0.5,0.5);
    SmapTC0.y = 1.0-SmapTC0.y;

    /* Depth test: is pixel visible from camera but not from light? */
    Sfactor1  = 1.0;
    SmapTCB   = SmapTC0 + SMSDELA;
    DepthLSM  = tex2D(TS1, SmapTCB);
    if (DepthLSM.r>0.00) Sfactor1 = SVALUE;

    . . .

    /* Depth test: is pixel visible from camera but not from light? */
    Sfactor4  = 1.0;
    SmapTCB   = SmapTC0 - SMSDELB;
    DepthLSM  = tex2D(TS1, SmapTCB);
    if (DepthLSM.r>0.00) Sfactor4 = SVALUE;

    /* Shadow factor filtering, fast central average */
    Sfactor1  = 0.25*(Sfactor1+Sfactor2+Sfactor3+Sfactor4);

    /* Pixel color generation. Note shadow reduction in specular areas */
    Out.Color      = C1*In.vDiffuse;
    Out.Color.rgb *= Sfactor1.rrr;
    Out.Color.rgb += In.vSpecular*Sfactor1.rrr;

return Out;}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Billboard pixel shader for PSM output.

- Because we use a separate billboard faced with the BW silhouette for PSM renders, there is no need to kill the color output of the billboard pixel shader.

```
//-----------------------------------------------------------------------------//
//  Shader application control constants and texture  bindings                 //
//-----------------------------------------------------------------------------//
sampler  Tex0      ; // Api texture stage0

//-----------------------------------------------------------------------------//
//  Shader input and output structures                                         //
//-----------------------------------------------------------------------------//
struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ;
    float2 vTexCoord0       : TEXCOORD0;
};

struct PS_OUTPUT
{   // Extant pixel shader stream
    float4 Color            : COLOR0   ;
};

//-----------------------------------------------------------------------------//
//  Shader principal operations (PSM render, billboards)                       //
//-----------------------------------------------------------------------------//
PS_OUTPUT Shader_Ex04_smpPS_ObjBbd (VS_OUTPUT In) { PS_OUTPUT Out = (PS_OUTPUT)0;

    /* Pixel color generation */
    Out.Color      = tex2D(Tex0, In.vTexCoord0);

return Out; }
//-----------------------------------------------------------------------------//
//  Shader principal operations                                                //
//-----------------------------------------------------------------------------//
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Solid 3D pixel shader for PSM output.

- Since we need to output only a b/w silhouette, the vertex and pixel shader can be greatly simplified!

```
//*************************************************************************//
//  Shader input and output structures                                     //
//*************************************************************************//
struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION ;
    float4 vDiffuse         : COLOR0   ;
//  float2 vDepth           : TEXCOORD0;
};

struct PS_OUTPUT
{   // Extant pixel shader stream
    float4 Color            : COLOR0   ;
};

//*************************************************************************//
//  Shader principal operations (PSM render, solid3D objects)              //
//*************************************************************************//
PS_OUTPUT Shader_Ex04_smpPS_ObjS3d (VS_OUTPUT In) { PS_OUTPUT Out = (PS_OUTPUT)0;

//  /* For interacting shadows, output float32 depth */
//  Out.Color  = In.vDepth.x / In.vDepth.y;

    /* For simple on-off shadows, just mark the pixel */
    Out.Color  = In.vDiffuse;

return Out;}
//*************************************************************************//
//  Shader principal operations                                            //
//*************************************************************************//
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Skinned 3D vertex shader for PSM output.

- PSM renders can be lightning fast, especially if geometry phasing is used!

```
##include "..\Demo_Common\Files_Media\Shaders\VS_Std_Includes.shd"
float4x4 mBone[11] : register (c60);  // Api: bone matrices stack
float4   wPos      ;                  // Wrk: Position vector, world space

//-----------------------------------------------------------------------------//
//  Shader input and output structures                                         //
//-----------------------------------------------------------------------------//

struct VS_INPUT
{   // Inbound vertex buffer stream
    float3 vPosition        : POSITION      ;
    int4   vIndices         : BLENDINDICES0 ;
    float3 vNormal          : NORMAL        ;
    float2 vTexCoords       : TEXCOORD0     ;
};

struct VS_OUTPUT
{   // Extant processed vertex stream
    float4 vPosition        : POSITION  ;
    float4 vDiffuse         : COLOR0    ;
};

//-----------------------------------------------------------------------------//
//  Shader principal operations (PSM render, skinned objects)                  //
//-----------------------------------------------------------------------------//
VS_OUTPUT Shader_Ex04_smpVS_ObjSkinned (const VS_INPUT In) { VS_OUTPUT Out = (VS_OUTPUT) 0;

    /* Position world-view-projection transform and position output */
    wPos              = mul(float4(In.vPosition,1.0f), mBone[In.vIndices.x]);
    Out.vPosition     = mul(wPos, mViewProj);

    /* For simple on-off shadows on ground, just mark the pixel */
    Out.vDiffuse      = float4( 1.0, 1.0, 1.0 , 1.0);

return Out;}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Principia V3.0.0 Demos

Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4



FPS: 32.3
GPS: 62.5
CLK: 21.0

Lighting on
Specular on
Perspective
Orthogonal

View
Lite1
Lite2
Lite3
Matl

Camera Distance 6.0
Camera Elevation 25.0
Camera Azimuth 45.0
Camera FoV 25.0
Look At Point 0.0 0.0 0.6

When shadow casting objects render at the edges of the PSM (e.g. the object is partly outside the light view) there are non-null pixels at UV=0|1 on the PSM texture. Portions of terrain that have no shadows may sample these PSM pixels during the normal render, leading to shadow block artefacts.

You can overwrite the PSM edges with a 1-pixel wide null-value line (a guard band) to cure such artefacts. Here, we have rendered the guard band in white on a small PSM – to make it visible- and shown the typical resulting artefact.

D108B:Ex04 – WorldRT3A

Shadow map set-up

- Note the need to connect the PSM camera viewport data to a shader register in order to let the guardband vertex shader render it properly. Refer to D104 for PSM fundamentals.

```
##define (D_SMAP) as <DC_SRFDEF3>
   #tag Type      =  (       TEXTURE )
   #tag Format    =  (      X8R8G8B8 )
   #tag Memory    =  (   GMEM_DEVICE )
   #tag IsTarget  =  (           YES )
   #tag IsStencil =  (           YES ) 'May not be needed in some of the simplest PSM cases !!!

##define (S_SMAP) as <SURFACE>
   #tag Definer   =  (        D_SMAP )
   #tag Width     =  (           512 )
   #tag Heigt     =  (           384 )
@@@@tag Width     =  (          1024 )
@@@@tag Heigt     =  (           768 )
   #tag LoadNow   =  (             1 )

##define (FX_CLEAR_SMAP) as <FXTEXGPU_FILL_3A>
   #tag FillColor    =  (    0x00000000  ,           YES )
   #tag FillDepth    =  (           1.0  ,           YES )
   #tag FillStncl    =  (             0  ,            NO )
   #tag FillShader   =  (             -  ,             - )

##define (CAMERA_FROMLITE) as <CAMERA_3A>
   #tag Target           =  (                S_SMAP )
   #tag Viewport         =  (  - ,  - ,    - ,    - )
   #tag View.Perspec     =  (                      0 )
   #tag View.EyePos      =  (   0.0 ,   0.0 ,   0.0 )
   #tag View.LookPos     =  (   0.0 ,   0.0 ,   0.0 )
   #tag View.UpVec       =  (   0.0 ,   0.0 ,   0.0 )
   #tag View.Angle       =  (                   0.0 )
   #tag View.Aspect      =  (                   0.0 )
   #tag View.ZNear       =  (                   0.0 )
   #tag View.ZFar        =  (                   0.0 )
   #tag VsReg_View       =  (                    49 )

##define <SYS_VARIABLE>   =  (   V_LITVIEWPROJ ,   VARTYPE_MAT  , "Decl(M);" )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Guard band set-up

- Rendered as a simple group of 4 transformed sprites, using the common unit-geometry flat-render shaders from the STD_FLATRENDER module.
- Note the use of register 55 required by the vertex shader to pass the null color of the guardband (0x000000FF) and the exact pixel positioning of the lower and left guardband elements at (W-2, H-2).

```
##define <SYS_VARIABLE>   =  (   V_GUARDBANDCOLOR , VARTYPE_FSR ,  0.00 ,  0.00 ,  0.00 ,  1.00 )

##define (M_GUARDBAND) as <MATERIAL_3A>
   #tag RS        =  (               RS_FILLMODE ,        RA_SOLID )
   #tag RS        =  (               RS_CULLMODE ,     RA_CULLNONE )
   #tag RS        =  (                RS_ZENABLE ,               0 )
   #tag RS        =  (           RS_ZWRITEENABLE ,               0 )
   #tag RS        =  (        RS_ALPHATESTENABLE ,               0 )
   #tag RS        =  (       RS_ALPHABLENDENABLE ,               0 )
   #tag SA        =  (             SS_MAGFILTER ,       SA_LINEAR ,   0 )
   #tag SA        =  (             SS_MINFILTER ,       SA_LINEAR ,   0 )
   #tag SA        =  (             SS_MIPFILTER ,       SA_LINEAR ,   0 )
   #tag VShader   =  (        VS_IMG_SPRITECOLOR )
   #tag PShader   =  (                PS_STD_FLAT )
   #tag VsVar     =  (     55 , V_GUARDBANDCOLOR )

##define (R_GUARDBAND) as <MESH_3A>
   #tag Component =  (  0 ,  M_GUARDBAND ,  G_STD_FLATSPRITE )

##define (O_GUARDBAND1) as <GENOBJECT_3A>
   #tag Construct =  (  "Decl(M); Mul(512.0, 1.0, 0); Mov(0.0, 0.0, 0);Set(tM,0);" ,  R_GUARDBAND ,     NONE )

##define (O_GUARDBAND2) as <GENOBJECT_3A>
   #tag Construct =  (  "Decl(M); Mul(512.0, 1.0, 0); Mov(0.0, 382.0, 0);Set(tM,0);" ,  R_GUARDBAND ,     NONE )

##define (O_GUARDBAND3) as <GENOBJECT_3A>
   #tag Construct =  (  "Decl(M); Mul(1.0, 384.0, 0); Mov(0.0, 0.0, 0);Set(tM,0);" ,  R_GUARDBAND ,     NONE )

##define (O_GUARDBAND4) as <GENOBJECT_3A>
   #tag Construct =  (  "Decl(M); Mul(1.0, 384.0, 0); Mov(510.0 0.0, 0);Set(tM,0);" ,  R_GUARDBAND ,     NONE )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World layer overall construction:

- **(A) Object holding and processing layers. These layers contain the actual object sets, and perform basic visibility and partitioning assessment on them.**
  - Placing objects in layers that process them without rendering is a common technique in multi-pass environments. This separates one-time operations from multiple renders.
  - Shadow mapping requires only two passes, thus we can put (1) visitest+partition in one layer, (2) then an update cycle to do all animation and calculation together with a first-pass render, and (3) final render-only pass.
  - With more complex pass structure, we may have to put all computational operations in a processing layer, followed by pure rendering linked layers.
  - Billboards of the type used here introduce a twist to the operation sequencing, because of the need to re-calculate the orientation matrix at every render pass.
  - Layers holding moving objects need continuous partitioning.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Non-renderable object processing layers (A):

- Test object-pointer status and set partition codes. No animation or calculations done here though.

```
##define (OP_VISITEST) as <OP_VISITEST_3R>
   #tag Method    = (           NONE )
   #tag TstObjVS  = (             NO )
   #tag SetObjVS  = (             NO )

##define (OP_PARTFIX) as <OP_PARTSECT_2R>
   #tag Frequency = (           NONE )

##define (OP_PARTDYN) as <OP_PARTSECT_2R>
   #tag Frequency = (              3 )


##define (L_OBJOPERS_BBD) as <WORLDLAYER_3R>
   #tag Name      = "Holding and operations layer, billboard objects"
   #tag Type      = (  LWT_NORENDER )
   #tag MatInit   = (           NONE )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_PARTFIX )
   #tag Operation = (    OP_VISITEST )
   #tag Count     = (           4012 )

##define (L_OBJOPERS_S3D) as <WORLDLAYER_3R>
   #tag Name      = "Holding and operations layer, solid3D objects"
   #tag Type      = (  LWT_NORENDER )
   #tag MatInit   = (           NONE )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_PARTFIX )
   #tag Operation = (    OP_VISITEST )
   #tag Count     = (           4012 )

##define (L_OBJOPERS_SKN) as <WORLDLAYER_3R>
   #tag Name      = "Holding and operations layer, skinned characters"
   #tag Type      = (  LWT_NORENDER )
   #tag MatInit   = (           NONE )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_PARTDYN ) 'Because these objects can move between sectors
   #tag Operation = (    OP_VISITEST )
   #tag Count     = (           4012 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World layer overall construction:

➥ (B) Object collision detection and resolution layer.

- This layer holds only operations on object-holding layers to identify potential collisions between moving objects. It is also bound to a user procedure in the application, which resolves the found collisions by adjusting object paths and implementing yielding behavior.
- The OP_INTERTEST layer operation maintains lists of all visible objects that are in a potential collision situation based on criteria defined in the script. The collision lists of this operation are processed by the user application (see the Principia V3 Reference for detail).

```
##define (OP_INTERACT) as <OP_INTERTEST_3R>
   #tag Frequency    = (                1 )
   #tag InitBinSize  = (               32 )
   #tag ColliderSrc  = (    L_OBJOPERS_SKN ,    -  ,    -  )
   #tag GuardBands   = (   0.02  ,    0.00 )

##define (L_OBJ_INTERACT) as <WORLDLAYER_3R>
   #tag Name         = "Pure operation layer for resolving collisions and object interaction"
   #tag Type         = (  LWT_NORENDER )
   #tag MatInit      = (          NONE )
   #tag MatTerm      = (          NONE )
   #tag Operation    = (   OP_INTERACT )
   #tag Count        = (             4 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World layer overall construction:

(C) Normal view object render layers. There are two types of such normal view layers.

- Layers linked to the pre-processed world objects execute an update cycle for all visible partitioned objects. This updates animation and processes character signal-state-behavior interaction once per cycle.
- Layers that hold objects that are not part of the shadow map, hold the (1) object display path, (2) world placer display, and (3) world cursor display elements.
- Update or rendering layers are designed to minimize GPU state changes by lumping as many of the common material states in an up-front layer material. Thus, we have three groups of layers – billboards, 3D solids and 3D skinned animations. Each of these layers has a master material applied at the beginning that sets shaders...etc. Within the layer, there is an occasional texture or parameter change to reflect object variety, but no shader or mass material changes.
- When object accessories use different rendering technique from the main object (as with the robot and its “Selected” crown), make sure that materials and meshes are properly sequenced and don’t incur excessive state cost.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Normal view object render layers (C)

```
##define (L_OBJNRM_BBD) as <WORLDLAYER_3R>
   #tag Name      = "Shadow casting objects, normal view, billboards"
   #tag Type      = (      LWT_PARTED )
   #tag MatInit   = (   M_OBJNORM_BBD ) 'General prefix material for transparentBB in world
   #tag MatTerm   = (            NONE )
   #tag Link      = (  L_OBJOPERS_BBD ) 'Render processed billboard content
   #tag Count     = (              16 )

##define (L_OBJNRM_S3D) as <WORLDLAYER_3R>
   #tag Name      = "Shadow casting objects, normal view, solid3D objects"
   #tag Type      = (      LWT_PARTED )
   #tag MatInit   = (   M_OBJNORM_S3D ) 'General prefix material for solid3D in world
   #tag MatTerm   = (            NONE )
   #tag Link      = (  L_OBJOPERS_S3D ) 'Render processed solid3D content
   #tag Count     = (              16 )

##define (L_OBJNRM_SKN) as <WORLDLAYER_3R>
   #tag Name      = "Shadow casting objects, normal view, skinned characters"
   #tag Type      = (      LWT_PARTED )
   #tag MatInit   = (   M_OBJNORM_SKN ) 'General prefix material for skinned characters in world
   #tag MatTerm   = (            NONE )
   #tag Link      = (  L_OBJOPERS_SKN ) 'Render processed skinned characters
   #tag Count     = (              16 )

##define (L_OBJSPC_PATH) as <WORLDLAYER_3R>
   #tag Name      = "Path visualization objects, not part of shadow rendering"
   #tag Type      = (    LWT_NORENDER ) 'Cannot use HIDE because user function will not process
   #tag MatInit   = (   M_OBJNORM_S3D )
   #tag MatTerm   = (            NONE )
   #tag Operation = (     OP_VISITEST )
   #tag Count     = (              16 )

##define (L_OBJSPC_PLACER) as <WORLDLAYER_3R>
   #tag Name      = "Object placer construction show layer"
   #tag Type      = (               - ) 'The placer manages is own visibility
   #tag MatInit   = (   M_OBJNORM_S3D )
   #tag MatTerm   = (            NONE )
   #tag Count     = (              16 ) 'Do not place anything in this layer because it is managed by the placer

##define (L_OBJSPC_CURSOR) as <WORLDLAYER_3R>
   #tag Name      = "Object cursor layer"
   #tag Type      = (               - ) 'The cursor manages is own visibility
   #tag MatInit   = (            NONE )
   #tag MatTerm   = (            NONE )
   #tag Count     = (              16 ) 'Do not place anything in this layer because it is managed by the cursor
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World layer overall construction:

(D) Shadow map set-up layers and operations.

- A “shadow map begin” layer executes two key operations (1) calculates the camera-from-light settings to maximize the use of the shadow map surface and minimize pixel swimming, and (2) switches the GPU to render from this camera. This layer also applies all common material settings for shadow map rendering.
- A “shadow map end” layer restores the normal camera and restores normal material settings.
- Camera restoration is effected through the VrtSetup operation, which sets up the billboard on-render orientation-calculation control variables in a way that billboards are rendered in the proper orientation during their second pass over the shadow map.
- With a dynamic PSM camera, some shadow pixel swimming can occur. Extent stabilization is performed by the PsmSetup procedure. There are two options (a) use objects with extent snap, as shown here, or (b) use tile coverage.
- The guardband is placed in a trivial layer that directly holds its four components and purely renders them without any processing.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Shadow map set-up layers and ops (D)

```
@@This operation calculates the shadow map camera based on preprocessed object visibility
##define (OP_SMAPSETUP) as <OP_PSMSETUP_2A>
   #tag WRL_Camera  = (     CAMERA_ST3D )
   #tag PSM_Camera  = ( CAMERA_FROMLITE )
   #tag PSM_Target  = (          S_SMAP )
   #tag PSM_Layers  = (  L_OBJOPERS_BBD ) 'Design A for sizing with swim-free shadow pixels
   #tag PSM_Layers  = (  L_OBJOPERS_S3D ) 'Design A for sizing with swim-free shadow pixels
   #tag PSM_Layers  = (  L_OBJOPERS_SKN ) 'Design A for sizing with swim-free shadow pixels
@@@@tag PSM_Layers  = (    L_OPER_TILES ) 'Design B for sizing with swim-free shadow pixels
   #tag PSM_MatVar  = (   V_LITVIEWPROJ )
   #tag Light_Dir   = ( VL_L1DX , VL_L1DY , VL_L1DZ )
   #tag Lc_ExtScale = (            4.0 )
   #tag Lc_MinSize  = (  4.0 , 4.0, 0.0 )
   #tag Lc_RoundFix = (            0.1 ) 'Design A for sizing with swim-free shadow pixels

@@This operation sets the render target and camera for the shadow map
##define (OP_SMAPVIEW_BGN) as <OP_VRTSETUP_2A>
   #tag Camera      = (       CAMERA_FROMLITE )
   #tag CtrlVarI    = (     V_ORIENTEXE ,   1 )
   #tag CtrlVarI    = (     V_ORIENTIDX ,   1 )

@@This operation returns the render target and camera to the normal world
##define (OP_SMAPVIEW_END) as <OP_VRTSETUP_2A>
   #tag Camera      = (          CAMERA_ST3D )
   #tag CtrlVarI    = (     V_ORIENTEXE ,   0 )
   #tag CtrlVarI    = (     V_ORIENTIDX ,   0 )

@@This layer initiates shadow map rendering
##define (LSGEN_BGN)  as <WORLDLAYER_3R>
   #tag Name        = "Shadow map preparation operations"
   #tag MatInit     = (            NONE )
   #tag MatTerm     = (     M_SMAP_INIT )
   #tag Operation   = (    OP_SMAPSETUP )
   #tag Operation   = ( OP_SMAPVIEW_BGN )
   #tag Count       = (               4 )

@@This layer terminates shadow map rendering
##define (LSGEN_END)  as <WORLDLAYER_3R>
   #tag Name        = "Shadow map preparation operations"
   #tag MatInit     = (            NONE )
   #tag MatTerm     = (     M_SMAP_TERM )
   #tag Operation   = ( OP_SMAPVIEW_END )
   #tag Count       = (               4 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World layer overall construction:

- (E) Shadow map rendering layers. They only render the already processed and updated objects on the PSM, using the simplified 0-1 PSM shaders.

```
@@This layer renders Billboard objects on the shadow map
##define (LSGEN_OBJ_BBD)  as <WORLDLAYER_3R>
   #tag Name        = "Shadow map generation, object render, billboards (from lite view)"
   #tag Type        = (      LWT_PARTED )
   #tag Type        = (    LWT_PURENDER ) ' Just render the geometry, no updates (including animation)
   #tag MatInit     = (   M_OBJSMAP_BBD ) ' General material for transparent BB on SMAP
   #tag MatTerm     = (            NONE )
   #tag Link        = (  L_OBJOPERS_BBD ) ' Render processed billboard content
   #tag Count       = (               4 )

@@This layer renders Solid3D objects on the shadow map
##define (LSGEN_OBJ_S3D)  as <WORLDLAYER_3R>
   #tag Name        = "Shadow map generation, object render, solids3d (from lite view)"
   #tag Type        = (      LWT_PARTED )
   #tag Type        = (    LWT_PURENDER ) ' Just render the geometry, no updates (including animation)
   #tag MatInit     = (   M_OBJSMAP_S3D ) ' General material for solid3D on SMAP
   #tag MatTerm     = (            NONE )
   #tag Link        = (  L_OBJOPERS_S3D ) ' Render processed solid3D content
   #tag Count       = (               4 )

@@This layer renders Skinned objects on the shadow map
##define (LSGEN_OBJ_SKN)  as <WORLDLAYER_3R>
   #tag Name        = "Shadow map generation, object render, skinned characters (from lite view)"
   #tag Type        = (      LWT_PARTED )
   #tag Type        = (    LWT_PURENDER ) ' Just render the geometry, no updates (including animation)
   #tag MatInit     = (   M_OBJSMAP_SKN ) ' General material for skinned characters on SMAP
   #tag MatTerm     = (            NONE )
   #tag Link        = (  L_OBJOPERS_SKN ) ' Render processed skinned characters
   #tag Count       = (               4 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Layer material synopsis (E):

- PSM materials may change the graphic phase to low-res geomery and disabled material changes (not done here). Note the need to re-bind shader constants and optionally reset depth-stencil parameters.

```
@@This material sets common PSM render settings and clears the PSM render target
##define (M_SMAP_INIT) as <MATERIAL_3A>
   #tag RS        =  (                RS_LIGHTING ,              0 )
   #tag RS        =  (                 RS_ZENABLE ,              1 )
   #tag RS        =  (            RS_ZWRITEENABLE ,              1 )
   #tag RS        =  (               RS_DEPTHBIAS ,        0.00000 )
   #tag RS        =  (           RS_STENCILENABLE ,              0 )
   #tag RS        =  (               RS_FILLMODE ,       RA_SOLID )
   #tag RS        =  (              RS_SHADEMODE ,     RA_GOURAUD )
   #tag RS        =  (               RS_CULLMODE ,     RA_CULLCCW )
   #tag RS        =  (        RS_ALPHATESTENABLE ,              0 )
   #tag RS        =  (       RS_ALPHABLENDENABLE ,              0 )
   #tag TS        =  (                TS_COLOROP ,    TO_DISABLE  ,   1 )
   #tag TS        =  (                TS_ALPHAOP ,    TO_DISABLE  ,   1 )
   #tag TX        =  (                  NONE , 1 )
   #tag VShader   =  (                     NONE )
   #tag PShader   =  (                     NONE )
@@@@tag Phase     =  (       GRAFIC_PHASE_NOMATL )
   #tag FX        =  (             FX_CLEAR_SMAP ,              - ,   0 )
   #tag VsMat     =  (          4 ,      MAT_VPT ) 'Needed repeat because camera changes

@@This material restores default 3D world render settings after PSM stage
##define (M_SMAP_TERM) as <MATERIAL_3A>
   #tag RS        =  (                RS_LIGHTING ,              0 )
   #tag RS        =  (                 RS_ZENABLE ,              1 )
   #tag RS        =  (            RS_ZWRITEENABLE ,              1 )
   #tag RS        =  (               RS_DEPTHBIAS ,        0.00000 )
   #tag RS        =  (           RS_STENCILENABLE ,              0 )
   #tag RS        =  (               RS_FILLMODE ,       RA_SOLID )
   #tag RS        =  (              RS_SHADEMODE ,     RA_GOURAUD )
   #tag RS        =  (               RS_CULLMODE ,     RA_CULLCCW )
   #tag RS        =  (        RS_ALPHATESTENABLE ,              0 )
   #tag RS        =  (       RS_ALPHABLENDENABLE ,              0 )
   #tag TS        =  (                TS_COLOROP ,    TO_DISABLE  ,   1 )
   #tag TS        =  (                TS_ALPHAOP ,    TO_DISABLE  ,   1 )
   #tag TX        =  (                  NONE , 1 )
   #tag VShader   =  (                     NONE )
   #tag PShader   =  (                     NONE )
   #tag Phase     =  (       GRAFIC_PHASE_NORMAL )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Layer material synopsis (E):

- PSM rendering layer materials re-set shaders, and perform some small but vital changes to GPU states such as cull...etc.

```
##define (M_OBJSMAP_BBD) as <MATERIAL_3A>
   #tag RS         =  (                RS_FILLMODE ,        RA_SOLID )
   #tag RS         =  (                RS_CULLMODE ,     RA_CULLNONE )
   #tag RS         =  (                  RS_ZENABLE ,               1 )
   #tag RS         =  (             RS_ZWRITEENABLE ,               1 )
   #tag RS         =  (          RS_ALPHATESTENABLE ,               1 )
   #tag RS         =  (                 RS_ALPHAREF ,               0 )
   #tag RS         =  (               RS_ALPHAFUNC ,      CA_GREATER )
   #tag RS         =  (         RS_ALPHABLENDENABLE ,               0 )
   #tag RS         =  (                RS_SRCBLEND ,     FA_SRCALPHA )
   #tag RS         =  (               RS_DESTBLEND ,  FA_INVSRCALPHA )
   #tag SA         =  (                SS_ADDRESSU ,         SA_WRAP ,   0 )
   #tag SA         =  (                SS_ADDRESSV ,         SA_WRAP ,   0 )
   #tag SA         =  (                SS_MAGFILTER ,       SA_LINEAR ,   0 )
   #tag SA         =  (                SS_MINFILTER ,       SA_LINEAR ,   0 )
   #tag VShader    =  (                        NONE )
   #tag PShader    =  (             PS_EX04_PSMBBD )

##define (M_OBJSMAP_S3D) as <MATERIAL_3A>
   #tag RS         =  (                RS_FILLMODE ,        RA_SOLID )
   #tag RS         =  (                RS_CULLMODE ,      RA_CULLCCW )
   #tag RS         =  (                  RS_ZENABLE ,               1 )
   #tag RS         =  (             RS_ZWRITEENABLE ,               1 )
   #tag RS         =  (                RS_DEPTHBIAS ,         0.00000 )
   #tag RS         =  (          RS_ALPHATESTENABLE ,               0 )
   #tag RS         =  (         RS_ALPHABLENDENABLE ,               0 )
   #tag SA         =  (                SS_ADDRESSU ,         SA_WRAP ,   0 )
   #tag SA         =  (                SS_ADDRESSV ,         SA_WRAP ,   0 )
   #tag SA         =  (                SS_MAGFILTER ,       SA_LINEAR ,   0 )
   #tag SA         =  (                SS_MINFILTER ,       SA_LINEAR ,   0 )
   #tag TX         =  (                    NONE , 1 )
   #tag VShader    =  (             VS_EX04_PSMS3D )
   #tag PShader    =  (             PS_EX04_PSMS3D )
   #tag Material   =  (         MSHD_STDCAMLIT , - )

##define (M_OBJSMAP_SKN) as <MATERIAL_3A>
   . . .
   #tag VShader    =  (             VS_EX04_PSMSKN )
   #tag PShader    =  (             PS_EX04_PSMSKN )
   #tag Material   =  (         MSHD_STDCAMLIT , - )
   . . .
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Layer material synopsis (E): normal materials.

```
##define (M_OBJNORM_BBD) as <MATERIAL_3A>
   #tag RS         =  (                RS_FILLMODE ,        RA_SOLID )
   #tag RS         =  (                RS_CULLMODE ,     RA_CULLNONE )
   #tag RS         =  (                  RS_ZENABLE ,               1 )
   #tag RS         =  (             RS_ZWRITEENABLE ,               1 )
   #tag RS         =  (          RS_ALPHATESTENABLE ,               1 )
   #tag RS         =  (                 RS_ALPHAREF ,               0 )
   #tag RS         =  (               RS_ALPHAFUNC ,      CA_GREATER )
   #tag RS         =  (         RS_ALPHABLENDENABLE ,               1 )
   #tag RS         =  (                RS_SRCBLEND ,     FA_SRCALPHA )
   #tag RS         =  (               RS_DESTBLEND ,  FA_INVSRCALPHA )
   #tag SA         =  (                SS_ADDRESSU ,         SA_WRAP ,   0 )
   #tag SA         =  (                SS_ADDRESSV ,         SA_WRAP ,   0 )
   #tag SA         =  (               SS_MAGFILTER ,       SA_LINEAR ,   0 )
   #tag SA         =  (               SS_MINFILTER ,       SA_LINEAR ,   0 )
   #tag VShader    =  (                       NONE )
   #tag PShader    =  (             PS_EX04_NRMBBD )

##define (M_OBJNORM_S3D) as <MATERIAL_3A>
   #tag RS         =  (                RS_FILLMODE ,        RA_SOLID )
   #tag RS         =  (                RS_CULLMODE ,      RA_CULLCCW )
   #tag RS         =  (                  RS_ZENABLE ,               1 )
   #tag RS         =  (             RS_ZWRITEENABLE ,               1 )
   #tag RS         =  (                RS_DEPTHBIAS ,         0.00000 )
   #tag RS         =  (          RS_ALPHATESTENABLE ,               0 )
   #tag RS         =  (         RS_ALPHABLENDENABLE ,               0 )
   . . .
   #tag VShader    =  (             VS_EX04_NRMS3D )
   #tag PShader    =  (             PS_EX04_NRMS3D )
   #tag Material   =  (         MSHD_STDCAMLIT , - )

##define (M_OBJNORM_SKN) as <MATERIAL_3A>
   #tag RS         =  (                RS_FILLMODE ,        RA_SOLID )
   #tag RS         =  (                RS_CULLMODE ,      RA_CULLCCW )
   #tag RS         =  (                  RS_ZENABLE ,               1 )
   #tag RS         =  (                RS_DEPTHBIAS ,         0.00000 )
   #tag RS         =  (             RS_ZWRITEENABLE ,               1 )
   . . .
   #tag VShader    =  (             VS_EX04_NRMSKN )
   #tag PShader    =  (             PS_EX04_NRMSKN )
   #tag Material   =  (         MSHD_STDCAMLIT , - )
   #tag SA         =  (               SS_MAGFILTER ,       SA_LINEAR ,   0 )
   #tag SA         =  (               SS_MINFILTER ,       SA_LINEAR ,   0 )
   #tag SA         =  (               SS_MAGFILTER ,       SA_LINEAR ,   1 )
   #tag SA         =  (               SS_MINFILTER ,       SA_LINEAR ,   1 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World layer overall construction:

(F) Tile processing and rendering layers.

- Those are organized in essentially the same manner as in Ex03, except that the ground layer has to bind to shaders and textures for sampling and processing the PSM.
- Tiles are held, partitioned and visitested in a processing layer that holds all tiles in the world.
- Ground tiles are updated in two layers, each having a different texture palette, to minimize state changes.
- Water tiles are updated in a single layer with a unified shader design for all flowing and waving water.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Tile processing and rendering layers (F)

```
##define (L_TILES_PAL1) as <WORLDLAYER_3R>
   #tag Name      = "Ground tiles, display layer for texture palette A"
   #tag Type      = (      LWT_TILE )
   #tag Type      = (    LWT_PARTED )
   #tag MatInit   = (  ML_GRND_PAL1 )
   #tag MatTerm   = (          NONE )
   #tag Count     = (          1024 )

##define (L_TILES_PAL2) as <WORLDLAYER_3R>
   #tag Name      = "Ground tiles, display layer for texture palette B"
   #tag Type      = (      LWT_TILE )
   #tag Type      = (    LWT_PARTED )
   #tag MatInit   = (  ML_GRND_PAL2 )
   #tag MatTerm   = (          NONE )
   #tag Count     = (          1024 )

##define (L_TILES_WATR) as <WORLDLAYER_3R>
   #tag Name      = "Water tiles, holding and display layer"
   #tag Type      = (      LWT_TILE )
   #tag Type      = (    LWT_PARTED )
   #tag MatInit   = (       ML_WATR )
   #tag MatTerm   = (          NONE )
   #tag Operation = (    OP_PARTFIX )
   #tag Operation = (   OP_VISITEST )
   #tag Count     = (          1024 )

##define (L_OPER_TILES) as <WORLDLAYER_3R>
   #tag Name      = "Holding and operations layer, ground tiles"
   #tag Type      = (  LWT_NORENDER )
   #tag MatInit   = (          NONE )
   #tag MatTerm   = (          NONE )
   #tag Operation = (    OP_PARTFIX )
   #tag Operation = (   OP_VISITEST )
   #tag Count     = (          2096 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **The world put together! Note the settings for the path-finding tags and options.**

```
##define (WRLD_EX04) as <WORLD_RT3A>

   #tag Sector_Mode   =  (           WPART_RECT2D )
   #tag Sector_Dim    =  (         4 ,    4 ,   1 ) ' Should be tile multiple
   #tag Sector_Band   =  (                  0.03 ) ' Sector safety guardband
   #tag Tile_SetCodes =  (  0 ,              YES ) ' Use tile set #0 for cursor
   #tag Tile_SetCodes =  (  1 ,               NO ) ' Nus tile set #1 for cursor
   #tag Path_Design   =  (                     - ) ' Inherit from generator
   #tag Path_SrchOpts =  ( PATF_A8 | PATF_HDIST2I ) ' Path search options
   #tag Path_SrchCntl =  (                     - ) ' Automatic

   @@@@ Object processing, animation and assessment
   #tag Layer_Decl  =  (  L_OBJOPERS_BBD   ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJOPERS_S3D   ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJOPERS_SKN   ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJ_INTERACT   ,              NONE ,   NONE ,     NONE )

   @@@@ Object rendering, prior to SMAP (use opersets) and tiles (early-z gains)
   #tag Layer_Decl  =  (  L_OBJNRM_BBD     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJNRM_S3D     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJNRM_SKN     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJSPC_PATH    ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJSPC_PLACER  ,              NONE ,   NONE ,     NONE )

   @@@@ Shadow map generation (light space render of processed objects)
   #tag Layer_Decl  =  (  LSGEN_BGN        ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  LSGEN_OBJ_BBD    ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  LSGEN_OBJ_S3D    ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  LSGEN_OBJ_SKN    ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  LSGEN_GUARDBAND  ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  LSGEN_END        ,              NONE ,   NONE ,     NONE )

   @@@@ Tile processing and rendering
   #tag Layer_Decl  =  (  L_TILES_INI      ,       "TILES_INIT" ,     - ,           - ,      4 )
   #tag Layer_Decl  =  (  L_OPER_TILES     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_TILES_PAL1     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_TILES_PAL2     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_TILES_WATR     ,              NONE ,   NONE ,     NONE )
   #tag Layer_Decl  =  (  L_OBJSPC_CURSOR  ,              NONE ,   NONE ,     NONE )

   @@@@ Embedded objects
   #tag Layer_Item  =  (  L_TILES_INI      ,         O_TILEKNX )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **That's not all! In Ex04, snippets of user code have to be bound to specific layers:**
  - A user procedure ("Lex") bound to the character processing layer manages object states, selections and actions. *With Layer3R, user procedures are executed after operations and before updates*.
  - A user procedure ("Pat") bound to the special path display layer generates and shows the object paths when appropriate.
  - A user procedure ("Crs") bound to the object collision detection layer resolves the collisions. We shall review all these later.

```
BGN_PRINCIPIA_MAIN (PROD_NAME, PROD_VERS, ROOT_NAME, INST_FILE, CORE_DUMP){

   /* Bind the user custom code to particular application objects */
   . . .

   /* Bind user functions to specific world layers */
   Principia->BindUserProcMain ( "L_OBJOPERS_SKN"   , Lex_Ex04_Main   );
   Principia->BindUserProcMain ( "L_OBJSPC_PATH"    , Pat_Ex04_Main   );
   Principia->BindUserProcMain ( "L_OBJ_INTERACT"   , Crs_Ex04_Main   );

   /* Execute the application main Principia script */
   Principia->ParseFile(CONF_FILE);

END_PRINCIPIA_MAIN}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Character design**
  - Skinned interactive characters, integrated in a complex world, bring together many of the powerful features of Principia's components.
- **Character design aspects**
  - Object type implementation
  - Animation design
  - States and appearances
  - Behaviors
  - Interaction with world
  - Interaction with cursors and placers
  - State management

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Ex04 character implementation

- Biped1A :: AX_Object()+AX_Interactive()
- AX_Interactive() dual archetype: Implements states, signal-driven state management, path kinematics and behaviors without coding!
- Geometry speed/quality balance: Skinned geometry with 11-bone design. Biped1A() has state-dependent geometry configuration with mesh alternates and additives.
- Animation speed/quality balance: use Kinex_M mixed keyframe/interpolated animation.
- States: Walking/standing and selected/deselected. States can be managed from the Api or by signals.
- Behaviors: BX_GenStrideRT3A enables character to walk a path following the world surface in user-defined manner.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character definition: states and behaviors

- States must be binary superposable!
- Striding behavior (GenStrideRT3A) has two different alternates: walk and stand. At the end of a path, turn and loop back (an alternate where the character stops there sends a signal to the biped to assume a standing state from a walking state).
- The stride follows the world height (the path uses only X and Y). WorldRT3A designs work the smoothest when Z is elevation.

```
##define <NAME_VAL>       =  (    S_STND      ,  0x00000001 )
##define <NAME_VAL>       =  (    S_WALK      ,  0x00000002 )
##define <NAME_VAL>       =  (    S_SELECT    ,  0x00010000 )


##define (B_STRIDE) as <BHV_GENSTRIDE_RT3A>
   #tag StateStride    =  (     S_WALK ,    0.100 ,    1.200 ) 'State+stride length in world units+stride period in world clock
   #tag StateStride    =  (     S_STND ,    0.000 ,    1.200 ) 'State+stride length in world units+stride period in world clock
   #tag World_Heit     =  (        YES )
@@@@tag AtEnd_Stop     =  (        YES ,   CH_UNI ,  XO_STND , S_WALK )
@@@@tag AtEnd_Back     =  (         NO ,       - ,       - ,      - )
   #tag AtEnd_Stop     =  (         NO ,       - ,       - ,      - )
   #tag AtEnd_Back     =  (        YES ,       - ,       - ,      - )
   #tag AtEnd_Loop     =  (         NO ,       - ,       - ,      - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character definition: animation

- A KinexM component holds two sequences of keyframes: walk and stand cycle, and enables smooth interpolated transition between them without the need for connecting keyframes.
- The KinexM keyframe data is generated in Chapter 3.06, and is loaded from individual keyframe series files. Each keyframe holds the relative bone rotations for the 11-bone skinned mesh.

```
##define (K_ROBOT) as <KINEX_M>
   #tag File      =  "Files_Media\Ex04_Obj_Robot_Animation[Stnd30F].kx"
   #tag File      =  "Files_Media\Ex04_Obj_Robot_Animation[Walk30F].kx"
   #tag VsMat     =  (  60 ,  MAT_BT )

##define (AN_ROBOT) as <ANIMATOR_F>
   #tag Type      =  (     ANIM_SEQSTATE )
   #tag Clock     =  (         CLOCK_PMS )

   @States and keyframes for 'Stnd' animation
   #tag AState    =  (        S00 ,      0 )
   . . .
   #tag AState    =  (        S29 ,     29 )

   @States and keyframes for 'Walk' animation
   #tag AState    =  (        W00 ,     30 )
   . . .
   #tag AState    =  (        W29 ,     59 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character definition: animation, continued

- To enable a rapid animation response to changes in object state, there are multiple points where the animation cycles can be switched between each other.
- The changes in animation are signal-driven, and these signals will be managed by Principia and the Biped1A configuration.

```
@Motion chain, Walk cycle
#tag TransTMP      =  (   W00 ,   W01 ,   40 )
. . .
#tag TransTMP      =  (   W29 ,   W00 ,   40 )

@Motion chain, Stnd cycle
#tag TransTMP      =  (   S00 ,   S01 ,   40 )
. . .
#tag TransTMP      =  (   S29 ,   S00 ,   40 )

@Signalled transition between chains with timed interpolation
#tag TransSIG  =  (  W00 ,  S00 ,   700      , CH_UNI ,  XA_STND )
#tag TransSIG  =  (  W05 ,  S05 ,   700      , CH_UNI ,  XA_STND )
#tag TransSIG  =  (  W10 ,  S10 ,   700      , CH_UNI ,  XA_STND )
#tag TransSIG  =  (  W15 ,  S15 ,   700      , CH_UNI ,  XA_STND )
#tag TransSIG  =  (  W20 ,  S20 ,   700      , CH_UNI ,  XA_STND )
#tag TransSIG  =  (  W25 ,  S25 ,   700      , CH_UNI ,  XA_STND )

#tag TransSIG  =  (  S00 ,  W00 ,   700      , CH_UNI ,  XA_WALK )
#tag TransSIG  =  (  S05 ,  W05 ,   700      , CH_UNI ,  XA_WALK )
#tag TransSIG  =  (  S10 ,  W10 ,   700      , CH_UNI ,  XA_WALK )
#tag TransSIG  =  (  S15 ,  W15 ,   700      , CH_UNI ,  XA_WALK )
#tag TransSIG  =  (  S20 ,  W20 ,   700      , CH_UNI ,  XA_WALK )
#tag TransSIG  =  (  S25 ,  W25 ,   700      , CH_UNI ,  XA_WALK )

@Channel clears to prevent repeated commands from accumulating
#tag KillSIG   =  (         S01 ,    CH_UNI ,  XA_STND )
. . .
#tag KillSIG   =  (         W26 ,    CH_UNI ,  XA_WALK )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character definition: putting it all together

- Character has different appearance when walking or standing (implemented via AltMesh settings)
- Character has an additional mesh displayed at the root kinex when selected (selection “crown”, its rendering requires the root kinex to be defined so as it sets its base matrix at shader constant register #0).
- A generic base radius is defined to override the somewhat excessive natural pose radius. This is used for collision detection only. Screen state testing in Ex04 is done using the natural pose bounding. Principia offers very sophisticated object-pointer and object-object relationship management functions, but we will leave these alone until Section II.

```
@@Object specification
##define (O_ROBOT) as <BIPED_1A>

   #tag IniState  =  (          S_STND )
   #tag MeshAnim  =  (       K_ROBOT   ,      AN_ROBOT )
   #tag BodyMesh  =  (        S_STND   ,  R_ROBOT_STND )
   #tag BodyMesh  =  (        S_NORM   ,  R_ROBOT_STND )
   #tag Accessory =  (          S_SELECT ,  R_ROBOT_MARK ,      - )

   #tag Exclusive =  (  S_STND , S_WALK  )
   #tag RootKinex =  ( "Decl(M); Set(tM,0);" ,   NONE )
   #tag Behavior  =  (          B_STRIDE ,     S_STND )
   #tag Behavior  =  (          B_STRIDE ,     S_WALK )
   #tag Radii     =  (             0.10 ,          - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character definition: putting it all together

- Mouse-over-character changes the pointer (trivial signal)
- Left-click-over-character sends an XO_SELECT signal that sets the character state as selected. Reception of XO_WALK/XO_STND signals sets the state to walking or standing.
- Upon entering a walk or stand state, the appropriate animation signal is sent to the proper component to change the animation track as well. The AX_Interactive() biped archetype manages signals automatically.
- The signalling channel CH_UNI is defined to track sender IDs and to not allow duplicate signals in the same cycle. The latter is important to avoid impatient clicks from transforming into a mess.

```
@@Object specification

   . . .

   #method        =  (           ON_OVER ,        CH_CRS ,          X_OVER ,                  - )
   #method        =  (           ON_LCLK ,        CH_UNI ,       XO_SELECT ,                  - )
   #method        =  (     EXE_SETSTATE ,        CH_UNI ,       XO_SELECT ,   S_SELECT ,      - )
   #method        =  (     EXE_SETSTATE ,        CH_UNI ,         XO_WALK ,   S_WALK ,        - )
   #method        =  (     EXE_SETSTATE ,        CH_UNI ,         XO_STND ,   S_STND ,        - )

   #method        =  (     ON_UPONSTATE ,        CH_UNI ,         XA_WALK ,   S_WALK   ,      - )
   #method        =  (     ON_UPONSTATE ,        CH_UNI ,         XA_STND ,   S_STND   ,      - )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character definition: meshes

- The body meshes are straightforward. They use the same skinned geometry with different texture maps. Shaders and other GPU settings are not set per-object, but at the layer material level.
- The “selection crown” mesh however has to “unset” the layer material, since that mesh is not skinned nor it uses the same lighting model. This add-mesh sets its master material, then the texture material with which to render the geometry, and finally it restores the layer material.
- Since few characters have their “crowns” active at the same time, the performance impact from this add-mesh design is negligible.

```
@@Meshes
##define (R_ROBOT_STND) as <MESH_3A>
   #tag Component =  (   0 ,   MX_ROBOT_STND ,   G_ROBOT_BODY )

##define (R_ROBOT_WALK) as <MESH_3A>
   #tag Component =  (   0 ,   MX_ROBOT_WALK ,   G_ROBOT_BODY )

##define (R_ROBOT_MARK) as <MESH_3A>
   #tag Component =  (   0 ,   M_OBJNORM_S3D ,          NONE )
   #tag Component =  (   0 ,   MX_ROBOT_MARK ,   G_ROBOT_MARK )
   #tag Component =  (   0 ,   M_OBJNORM_SKN ,          NONE )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Character design look-ahead: Using Alt and Add-meshes with large crowds of characters.**
  - If the Alt/Add-meshes have distinct materials from the base mesh, excessive material change count will clobber performance.
  - If the character displays a shadow, then all Alt/Add meshes will display shadows too (as in here). There are two design options to manage these problems:
  - Option 1: Set the Add/Alt-meshes as individual objects on their own layer, managed by the user code. In this method, Add meshes will not have shadows unless their layer is explicitly designed for that purpose.
  - Option 2: Vertex-index the Add/Alt meshes to superpose over the base skinning design. Reserve a channel in the character kinex and texture maps to render them.
  - Sort layer objects by variety when there is significant material overhead. This will be covered later.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Character design look-ahead: Layer integration

- When using mouse methods with objects that rely on pointer state change (and derive from AX_Interactive), you must have a layer that invokes a screen state test, such as a layer with an OP_VISITEST operation.
- Interactive biped characters have a lot of calculations and animations. It is essential to do those separately from rendering. With multipass renders that are more complex than the PSM of Ex04, you will need to use a general scheduling operation instead of the simple OP_VISITEST.
- With large biped crowds, you there is increased overhead from processing signals locally per-object. It may be faster to process signals in a Api-bound layer procedure, instead of using the AX_Interactive() native signaling.
- When performance squeeze is an absolute must, consider doing the above and deriving your own custom objects from the native Principia code.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

User code, character processing layer:

- Many of the character actions cannot (and should not) be implemented at the object level. Instead, they must be managed in a procedure bound to the object processing layer. Here:
- (a) Figure out if there is a selected character. If clicks have caused more than one character from turning selected (remember, that feature is implemented at the local object level), use past state and viewer distance to make the most suitable character selected, and deselect the others. You need only change the state, the object will take care of its proper display.
- (b) If by chance we are in placer mode, do not allow clicks to be interpreted as selection or routing commands.
- (c) If an object is selected, manage the cursor display to indicate whether the cursor location is reachable or not.
- (d) A left click outside a selected object on a reachable location makes this object find a path to that location and walk there.
- (e) A left click on a selected and walking object causes it to stop.
- (f) A right click makes all selected objects to deselect.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

User code, character processing layer:

```
void Lex_Ex04_Main (void* VArg)
BGN_ACTOR_CODE{

   . . . Initializations and definitions .. .

   /* Find out where is the mouse with relation to objects, and how many objects are currently selected */
   NbSel = 0x00;
   NbOvr = 0x00;
   BiSel = NULL;
   BiOvr = NULL;
   for (i=0; i<Count; i++) begin{
       Biped = Crowd[i];
       if (BITMASK_TEST(Biped->InteractionFlag, INTER_OVER)) then{
          BiOvr = Biped;
          NbOvr++;
       endif}
       if (BITMASK_TEST(Biped->State, Ssel)) then{
          BiSel = Biped;
          NbSel++;
       endif}
   endfor}

   /* If there is more than one object currently selected, deselect all which did not */
   /* just change state. This ensures new selections replaces old ones automatically. */
   /* Note this does not ensure there will be a selection, hence viewer-dist fallback */
   if (NbSel>0x01) then{
      BiSel = NULL;
      for (i=0; i<Count; i++) begin{
          Biped = Crowd[i];
          if (BITMASK_TEST(Biped->State, Ssel)==BITMASK_TEST(Biped->StatePast, Ssel)) then{
             BITMASK_DEL(Biped->State, Ssel);
          }else{
             BiSel = Biped;
          endif}
       endfor}
       if (BiSel==NULL) then{
       if (STANDARD_ITEMTYPE_TEST(Viewr->PtrOver_Object,CLSID_BIPED1A)) then{
          BiSel = (BX_Biped1A*) Viewr->PtrOver_Object;
          BITMASK_SET(BiSel->State, Ssel);
       endif}
       endif}
   endif}

   /* If there is a selected object, manage the cursor display based on whether it is over a */
   /* reachable area of the grid or not. Note this may take long based on pathfinding method */
   if ((BiSel) && (Viewr->Cursor_GridI!=INVALID)) then{
      if (World->GridP->Get(Viewr->Cursor_GridI, Viewr->Cursor_GridJ)) then{
         SigY->Send(0x00);
      }else{
         SigN->Send(0x00);
      endif}
   }else{
      SigY->Send(0x00);
   endif}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

User code, character processing layer:

- While logically intricate, this code snippet implements a very complex workflow with few lines. This makes your robots walk and all !!!

```
. . .
   /* A left click outside of any object, with object selected results in path-rerouting */
   /* and an optional activation of the behavior if the object is stopped and path is ok */
   if (InputInterface->TestPointerLC()) then{
   if (Api_PtrPosX<950) then{
   if (!BiOvr) then{
   if (BiSel) then{
      if (BiSel->Path==NULL) then{
         BiSel->Path = new WX_Path();
         BiSel->Path->Allocate(0x0F,0x00);
      endif}
      if (Viewr->Cursor_GridI!=INVALID) then{
         World->FindPath(BiSel->X, BiSel->Y, Viewr->Cursor_GridX, Viewr->Cursor_GridY, xP, yP, N, 0xFFFFFFFF, GRID_NODEREF, 0x00);
      endif}
      if (N>0x00) then{
         BiSel->Path->ReMake_FromPos(xP, yP, N);
         BiSel->PathSetOn (0x00, 0x01);
         World->Elevation (BiSel);
         if (BITMASK_TEST(BiSel->State,Sstn)) SigW->Send(BiSel->ID);
      endif}
   endif}
   endif}
   endif}
   endif}

   /* A left click on a currently selected object results in the object stopping if walking. The mouse pos test is used to */
   /* avoid placer activation to be misinterpreted as an object directive. In production, this would be handled via a PJAY */
   if (InputInterface->TestPointerLC()) then{
   if (Api_PtrPosX<950) then{
   if (BiSel) then{
   if (BiSel==BiOvr) then{
      if (BITMASK_TEST(BiSel->State,Swlk)) SigS->Send(BiSel->ID);
   endif}
   endif}
   endif}
   endif}

   /* A rite click outside of an object with an object selected results in deselection. */
   if (InputInterface->TestPointerRC()) then{
   if (BiSel) then{
      for (i=0; i<Count; i++) begin{
          Biped = Crowd[i];
          BITMASK_DEL(Biped->State, Ssel);
      endfor}
   endif}
   endif}

END_ACTOR_CODE}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Path generation and display

- The “Lex” user code snippet uses AX_Interactive() and WX_WorldRT3A() powerful built-in capabilities to make a path for the character to walk with a single, straightforward line of code:

```
World->FindPath(BiSel->X, BiSel->Y, Viewr->Cursor_GridX, Viewr->Cursor_GridY, xP, yP, N, 0xFFFFFFFF, GRID_NODEREF, 0x00);
```

- We will delve into path-finding in a moment. For now, this command creates a standard WX_Path() Principia component attached to our character, which is used by the GenStrideRT3A() behavior to make the character walk. We do not really care about these internals, but we want to display the path as a dotted line showing waypoints and all.
- A “Pat” user code snipped bound to a layer dedicated to path display, generates a renderable path object when appropriate, and controls it display as well. The renderable path object is a Principia generic particle BX_GenericP() object, seen in Chapter 7.
- The dedicated path layer is not hidden (because that would prevent the execution of the “Pat” user procedure). Instead, it is toggled between render and renderless states, with ordinary solid 3D shaders.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Path layer and object definition

- The path layer is not partitioned, but it has to be visi-tested. The path particle object has a box that is big enough so that it is always visible. Here, the performance hit is negligible and cuts detail beside our topic.

```
##define (L_OBJSPC_PATH) as <WORLDLAYER_3R>
   #tag Name      = "Path visualization objects, not part of shadow rendering"
   #tag Type      = (    LWT_NORENDER ) 'Cannot use HIDE because user function will not process
   #tag MatInit   = (   M_OBJNORM_S3D )
   #tag MatTerm   = (           NONE )
   #tag Operation = (     OP_VISITEST )
   #tag Count     = (              16 )


. . .


##define (MX_PATHMARK) as <MATERIAL_3A>
   #tag TX        =  (        S_ROBOT_MARK , 0 )

##define (G_PATHMARK) as <VERTEXSET_3A>
   #tag File      =  "Files_Media\Ex04_Obj_Robot_PathMarker.3ds"
   #tag Definer   =  ( VTD_DCL_FILE )
   #tag Indexed   =  (          YES )

##define (O_PATHMARK) as <GENOBJECT_3P>
   #tag MaxCount  =  (              256 )
   #tag Material  =  (       MX_PATHMARK )
   #tag Geometry  =  (        G_PATHMARK )
   #tag Animator  =  (                  - )
   #tag Kinex     =  (   "Decl(M); Set(tM,0);")
   #tag Box       =  (  -15.0, 15.0,   -15.0, 15.0,   -15.0, 15.0 )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

User code, path object generation and control:

```
void Pat_Ex04_Main (void* VArg)
BGN_ACTOR_CODE{ . . . Skip initializations . . .

   /* Determine whether there is a path to be displayed (a selected object with a path) */
   Path = NULL;
   for (i=0; i<Biset->N; i++) begin{
       Biped = Biset->ePtr[i];
       if (BITMASK_TEST(Biped->State, Ssel))
       if (Biped->Path)
       if (Biped->Path->N>0x01)
       Path = Biped->Path;
   endfor}

   /* Generate the path layer renderable object if it does not exist and if it is needed */
   if (Path) then{
   if (Paths->N==0x00) then{
      Pobj = new BX_Generic3P(Proto);
      Paths->Add(Pobj);
   endif}
   endif}

   /* Update the particle positions by analyzing the current path contents */
   if (Path) then{
   if (Paths->N) then{
      j  = 0x00;
      for (k=0; k<Path->N-1; k++) begin{
          XA = Path->X[k]; XB = Path->X[k+1];
          YA = Path->Y[k]; YB = Path->Y[k+1];
          P  = Path->S[k];
          N  = RoundInt( P / World->GridDX );
          N  = max(0x01, N);
          for (i=0; i<N; i++) begin{
              P = float(i-1)/float(N);
              X = XA + P*(XB-XA);
              Y = YA + P*(YB-YA);
              Z = World->Elevation(X,Y);
              pX[j] = X;
              pY[j] = Y;
              pZ[j] = Z + Zup;
              pS[j] = (i==0x00) ? ONE : HALF;
              j++;
          endfor}
      endfor}
      pX[j] = XB;
      pY[j] = YB;
      pZ[j] = World->Elevation(XB,YB) + Zup;
      pS[j] = ONE;
      j++;
      Pobj->SetParticles(pX,pY,pZ,pS,j);
   endif}
   endif}

   /* If there is a selected object with path, show the path layer. Else, hide it */
   if (Path) then{
      SigS->Send(0x00);
   }else{
      SigH->Send(0x00);
   endif}

END_ACTOR_CODE}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Viewer, Cursor, Placers …**
  - These are powerful programmable components that clearly do quite a lot behind the scenes here to implement the interactivity for creating and controlling characters.
  - Viewer: manages the cursor, placer and a host of other “atom-level” world interaction components we will visit in Section II. Besides its many functions, the viewer keeps track of which objects and tiles are interacting with the pointer.
  - Cursor: a component that indicates a set of cells under pointer scope with context-sensitive visuals
  - Placer: a component that enables the user to create objects in the world via the cursor.
- **The combination of viewer, cursor, placer and user code in Ex04 implements a very common interaction workflow for strategy games.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Viewer definition:

- Our viewer features a cursor and a placer
- It tracks objects in the BBD, S3D and SKN layers
- Left clicks outside objects and right clicks anywhere in the viewer generate signals that are used by our placer to implement standard placer methods such as “CREATE” or “CANCEL”

```
##define (VIEW_EX04) as <WORLDVIEWER_RT3A>
   #tag World          =  (     WRLD_EX04 )
   #tag Camera         =  (   CAMERA_ST3D )
   #tag Cursor         =  (       OBJ_CRS )
   #tag Placer         =  (    OBJ_PLACER )
   #tag PlateBtm       =  (      KJAY_CRS ,          - ,         - )
   #tag PlateTop       =  (          NONE ,          - ,         - )
   #tag ObjectTrack    =  (             L_OBJOPERS_BBD ,         - ,                - )
   #tag ObjectTrack    =  (             L_OBJOPERS_S3D ,         - ,                - )
   #tag ObjectTrack    =  (             L_OBJOPERS_SKN ,         - ,                - )
   #tag RimHaltView    =  (           YES )
   #method             =  (    ON_LCLKCEL ,     CH_WSG ,    X_PLON )
   #method             =  (    ON_RCLKCEL ,     CH_WSG ,    X_PLEX )
   #method             =  (    ON_RCLKOBJ ,     CH_WSG ,    X_PLEX )
   #method             =  (  EXE_DELLTYPE ,     CH_WRL ,    X_SHOW ,    L_OBJSPC_PATH ,    LWT_NORENDER )
   #method             =  (  EXE_SETLTYPE ,     CH_WRL ,    X_HIDE ,    L_OBJSPC_PATH ,    LWT_NORENDER )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Cursor3R for WorldRT3A/ViewerRT3A

- *Position datums:* The actual and detailed pointer-over-world location tracking is done in the ViewerRT3A component. This information is passed to the cursor and managed as two tracking data items:
  - XYZ cursor pointer position on the world grid. This position can be snapped to the middle of the nearest-to-viewer-containing cell (CELLREF mode) or nearest-to-pointer terrain vertex of a visible containing cell (NODEREF mode).
  - IJ index position on the world grid. If (IJ) is invalid, the cursor is either outside the viewer, or is over a terrain location that is deemed invalid (see below).
- *Cursor validity:* The cursor uses a standard Principia footprint type assessment to determine whether a given IJ position is valid. Cursors that are not valid are not displayed. This is not to be confused with invalid object creation or routing positions, over which the cursor here is displayed in a different form.
- *Cursor states:* Like an AX_Interactive() component, the cursor has states and methods to change them via signals.
- *Cursor modes:* The cursor has a cell mode (terrain grid cell blocks around IJ are highlighted) and an object mode (specific meshes are placed in a desired layer and rendered at XYZ). What is displayed can be linked to a given state – which is how our placer and Api implement cursor actions.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [4] [2054] [51\6709] [1][74][0]
Cell: [235,236, 0.206]
GRID_NODEREF cursors always point at terrain vertex nodes. This GRID_NODEREF illustration shows how an object and single-cell cursor images would be displayed. The cell displayed is the one that is overlaid by the mouse pointer. The yellow arrow object displayed is at the vertex that is closest to the mouse pointer and the camera. The yellow arrow shows the XYZ value pointed and returned by the cursor object. The grid cursor IJ index returned is the vertex IJ index on the world grid (and not the cell head IJ index)
FPS: 32.3
GPS: 25.0
CLK: 83.6
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
2.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
-0.5
-0.5
0.6


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [4] [2054] [46\6709] [1][74][0]
Cell: [235,236, 0.206]
FPS: 32.3
GPS: 62.5
CLK: 34.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
2.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
-0.3
-0.3
0.6
GRID_CELLREF cursors always point at the midpoint of terrain cells. This GRID_CELLREF illustration shows how an object and single-cell cursor images would be displayed in this case. The yellow arrow shows the XYZ value pointed and returned by the cursor object. The grid index IJ pointer is the head of the cell.
Do not confuse the GRID_*REF cursor mode with the exclusion mode of the cursor. The latter simply determines how neighbouring cells to the the grid IJ index pointer will be tested to determine whether this is a valid cursor value.


Principia V3.0.0 Demos
Ex04: Basic WorldRT3A, Enhancement of Elevation Features
GenProc=WrldRT3A_Asm1RL, Viewer=RT3A, DCT render paradigm, VS:1.1, PS:1.4
Load: [4] [2054] [49\6709] [1][74][0]
Cell: [233,233, 0.239]
FPS: 32.3
GPS: 66.7
CLK: 70.3
Lighting on
Specular on
Perspective
Orthogonal
View
Lite1
Lite2
Lite3
Matl
Camera Distance
2.0
Camera Elevation
25.0
Camera Azimuth
45.0
Camera FoV
25.0
Look At Point
-0.2
-0.2
0.6
Multi-cell cursors are basically extensions of a single-cell cursor, with cells added behind and ahead of the root cell. The cursor shown is thus a ::SetCellCover(-1,2,-1,2). Note that we count the root cell. The single-cell cursor is restored using a ::SetCellCover(0,1,0,1). Multi-cell cursors are typically used by object placers to indicate the required footprint of an object, and whether it is feasible or not.

D108B:Ex04 – WorldRT3A

Cursor definition:

- Our cursor has two binary superposed states: VALID (the pointed location can receive a placed object or not) and WKBLE (the location pointed can be reached or not).
- By default, the cursor is VALID and WKBLE, and default mesh forms and cell materials are used. Should one of these states be not true however, an alternate is displayed. Signals in the WRL channel generated by the placer (for VALID) and the Api for (WKBLE) select the cursor appearance.
- Locations over water invalidate the cursor and do not display it at all.

```
##define <NAME_VAL>       =  (  CST_VALID , 0x00000001 )  'Cursor state, valid node
##define <NAME_VAL>       =  (  CST_WKBLE , 0x00000002 )  'Cursor state, walkable node

##define (OBJ_CRS) as <WORLDCURSOR_3R>
   #tag State         =  (  CST_VALID | CST_WKBLE )
   #tag Grid_Mode     =  (  GRID_CELLREF )
   #tag Grid_Exclude  =  (      TC_WATER ,   0.00 , 0.00 , 0.00 , 0.00 , GRID_METRECT , GRID_NODEREF ,  0.001 )
   #tag Use_Cells     =  (           YES )
   #tag Use_Object    =  (           YES )
   #tag Cell_Build    =  (  VTD_FVF_FILE ,          NONE , L_OBJSPC_CURSOR )
   #tag Cell_Display  =  (             0 ,         M_CRX ,            NONE )
   #tag Cell_Display  =  (     CST_VALID ,         M_CRS ,            NONE )
   #tag Obj_Display   =  (             0 ,  O_CURSOR_RED , L_OBJSPC_CURSOR )
   #tag Obj_Display   =  (     CST_WKBLE ,  O_CURSOR_YEL , L_OBJSPC_CURSOR )
   #method            =  (  EXE_DELSTATE ,     CH_WRL ,    X_VALNOT ,  CST_VALID )
   #method            =  (  EXE_SETSTATE ,     CH_WRL ,    X_VALINS ,  CST_VALID )
   #method            =  (  EXE_SETSTATE ,     CH_WRL ,    X_WKBLE  ,  CST_WKBLE )
   #method            =  (  EXE_DELSTATE ,     CH_WRL ,    X_NONWK  ,  CST_WKBLE )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Placer3R for WorldRT3A

- Placer3R implements the common “click on this menu button to drag and click-create a particular item at a desired terrain location” workflow seen in most 3rd person games.
- Placer3R maps different signals into commands to take different object prototypes, display them in a “drag mode” and instance them in the world upon a “valid click”. The mapping is:
  - Placed object prototype-channel-signal
  - “Drag” or construction temporary object
  - Layer for placed object instancing
  - Standard footprint validation codes: (a) required terrain type that must be in the footprint, (b) excluded terrain type that must not be in the footprint, (c) footprint extent as per Ex01, and (d) footprint reference and masking modes as per Ex01.
  - Code to mark the grid type field over the footprint upon successful object placement.
- General control tags, such as “InvalOnOver” which temporarily turns off the placer if the cursor is over an existing world object.
- Methods to generate signals based on placer events, to place objects or to cancel the placer based on entrant signals. Note that the Placer3R::osValidInsert data member is a very useful indicator to help the Api track the placement validity state of Placer3R.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Placer definition:

- Because we want the UNVA button click to turn the placer (but not do anything else in the world, given that it overlaps the viewport), we translate this signal into another signal in the CH_WSH channel.
- CH_WSG is a *deferred, topmost* channel of the Principia system interface. That means that only the topmost control clicked generates an actual signal. Thus, a click on UNVA does not send any click signals to the world viewer, because the latter is “below” the button.
- Our placer puts characters on any 3x3 footprint that is over flat land. It sends signals to the cursor to indicate footprint validity and operation termination. The cursor interprets these signals to adjust its cell print and appearance. No user code is necessary!

```
##define (SJAY_PLACER) as <SJAY_2A>
   #tag ProcSig_Replace    =  ( CH_NAV , X_UNVA ,   CH_WSG , X_UNVA )

##define (OBJ_PLACER) as <WORLDOBJECTPLACER_3R>
   #tag World          =  (          WRLD_EX04 )
   #tag TempLayer      =  (    L_OBJSPC_PLACER )
   #tag InvalOnOver    =  (               YES )
   #tag Obj_Placer     =  (       O_ROBOT ,   CH_WSG ,   X_UNVA , O_CONSTRUCT , L_OBJOPERS_SKN , TC_FLATS , 0x80800080 , -
      1.0,2.0, -1.0,2.0, GRID_METRECT , GRID_NODEREF , 0.001 , 0x00000000 , - )
   #method             =  (    EXE_PLACER ,     CH_WSG ,     X_PLON )
   #method             =  (    EXE_CANCEL ,     CH_WSG ,     X_PLEX )
   #method             =  (   ON_VALIDINS ,     CH_WRL ,  X_VALINS )
   #method             =  (   ON_VALIDNOT ,     CH_WRL ,  X_VALNOT )
   #method             =  (  ON_INSFINISH ,     CH_WRL ,  X_VALINS )
   #method             =  (  ON_INSCANCEL ,     CH_WRL ,  X_VALINS )
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

World navigation signals and channels:

- Implementing a relatively complex left-click driven object workflow such as in Ex04 requires well-though channel design.
- Problem: Unrelated clickable controls may overlay the world viewer and make it signal unrelated clicks (or too many impatient clicks).
- Solution1: Use DEFERRED/USETOP/UNISIG channels for world clicks and world menu controls to avoid multiple actions result from a single click, or to avoid unrelated control click from causing undesired world event.
- Solution2: Use a signal translation SJAY which introduces a 1-cycle delay. This solution is effective when combined with Solution1.

World signalling design principles:

- Use viewer object event queues and automated depth sorting to discriminate target objects.
- Process object events in user code (not as interactive objects) for massive crowds to save time, and also to avoid duplicate selections.
- Use dedicated channels for placers, world cursors and object communication
- Remember that the order of signal definition is important. Signals are processed in the order they are defined. Thus, the last of mutually exclusive signals (such as state change) trumps!

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

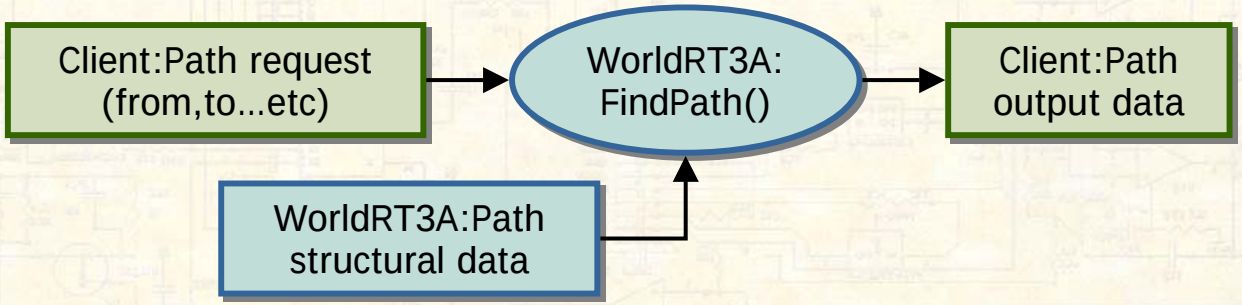
Path finding:

- Essential client operation for all world-based games!
- There is no single generic path-finding method that works across the variety of game worlds that one may design.
- Principia has several highly-efficient built-in path-finding methods for different world classes. These are meant to be used by the client code managing character movement, and user-selected by setting a simple script flag.
- Principia also offers the ability to provide an overriding custom-built path-finding method that is specific to the application world. The built-in methods code can be used as a template to support such custom designs.
- Different path-finding methods require different supporting data structures. These are generated procedurally or, for custom-designed path-finding, by the client code.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Path-finding real-time process:



Path-finding data structures:

- Output path data: SX_Set2A<flt,flt> of path nodes in world space. Easily converts to Principia WX_Path data items.
- Structural data: set of reference data structures describing terrain connectivity and passability. Specific to the method used. Automatically generated by Principia's procedures. Must be maintained by the client if the environment changes during gameplay in a way affecting path-finding.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **WorldRT3A path finding basic designs I:**
  - *Dynamic grid navigation (DGN) design*: Unlike the above, there is no predefined routing graph. The passable grid is explored using a selection of A* algorithms to find the route between two points in real time.
  - *Static route map (SRM) design*: A road network is predefined on the world, with a cloud of collection points to place off-road characters on a near road. The SRT graph is used to connect any two points with a fixed roadmap.
  - *Simple variable domain (SVD) design*: The world grid is tiled with a quilt of rectangular domains that define the passable features. The SVD domain graph is used to find routes dynamically between any two points.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

WorldRT3A path finding basic designs II:

- *Structured tile navigation (STN) design*: The world is partitioned in large uniform rectangular tiles. A path request is first resolved in tile space, followed by fine-resolution refinements using the DGN method.
- *Custom user (USR) design*: Principia invokes a user-defined path-finding procedure to resolve the path request in grid index space, followed by a conversion to XYZ space.

Path finding set-up:

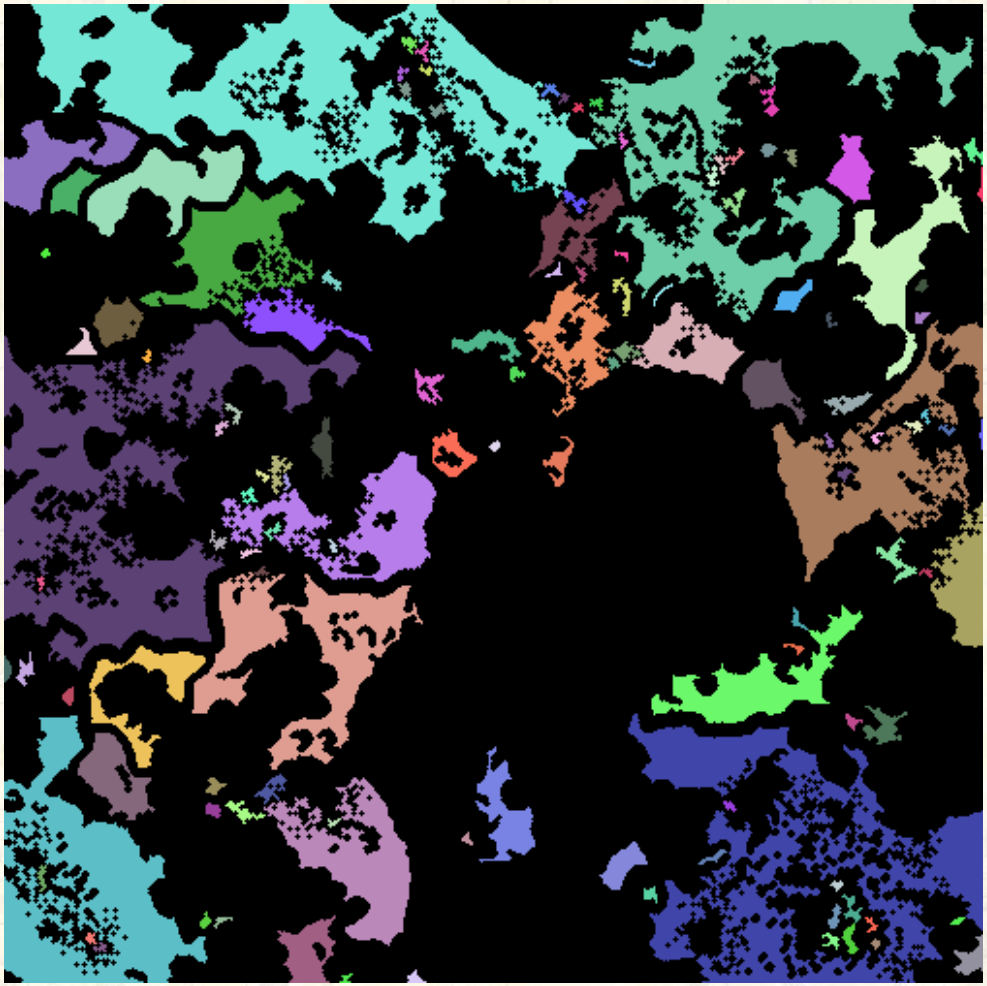
- Script, world generation: define pathfinding method so that the supporting data structures can be generated for you.
- Script, world definition: define pathfinding method and options to be used by WorldRT3A:FindPath().

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

DGN design

- Isolated passable features marked with non-null code on GridP.
- Slowest path finding, but most flexible foundation for dealing with dynamic obstacles and dynamic pathing.
- Design method is selected upon world generation. Pathing data is transferred to world from generator procedure or loaded from file.

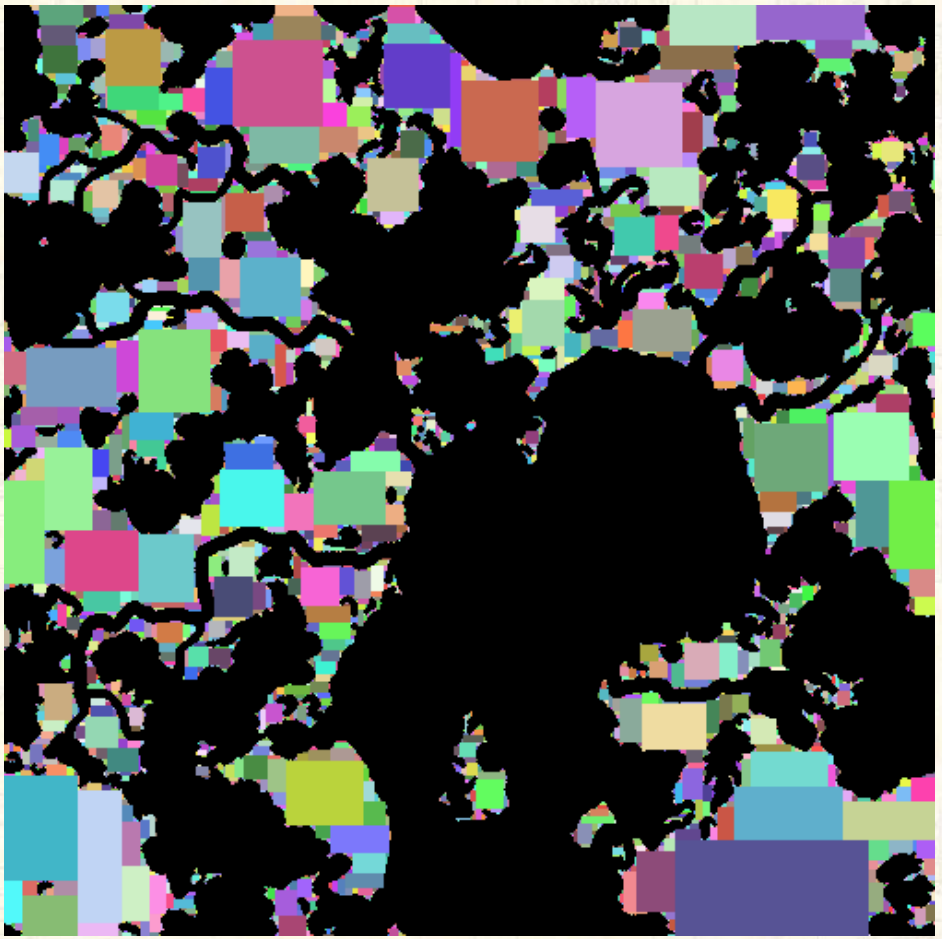


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

SVD design

- Quilt of “walkable” rectangular regions, derived using GridT markings and Principia topology procedures.
- Allows lightning-fast and very optimal route finding of any length with or without waypoints or bridges.
- Good for diverse terrain with fixed obstacles.
- Bad for terrain with many small obstacles or many small navigability changes.



Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- **Why so many different designs? Consider the number of important path design variables:**
  - *Fractal scale range of passable regions:* Is this a forest with discrete trees ("foam-like" passable grid") versus an open single domain plain? Will there be *hierarchical LoD* path-finding?
  - *Topological complexity of passable regions:* How many regions there are? Are there any bridges or portals? Dynamic waypoints? Are there any canyons or other bottlenecks?
  - *Morphology of passable regions and paths:* Is this a city grid with rectangular paths? Or an open plain with random movement?
  - *Movement cost variability:* Are there different types of terrain with variable cost of movement? For different moving units?
  - *Dynamism of obstacles and paths:* Do obstacles move? Can they appear or disappear during game play (e.g. tree harvesting)? Will the paths have to be re-calculated once motion begins?
  - *Nature and number of movers*: Will units have to move in group? Will motion be fully constrained or partly biased to roads?
  - *Clearance and collision:* Do movers have different clearances? Are movers treated as obstacles with *dynamic pathing*, or do we use static paths with *collision detection and resolution*?

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Basic built-in path-finding configuration tags:

- Path_Design: Activates switches for utilizing different method components for path-finding, and for pre-computing their supporting data structures. For the simpler designs introduced in Section I, the main switches to consider are PATF_SVD, PATF_SRM, PATF_DGN, PATF_STN or PATF_USR.
- Path_SrchCntl: Control values for the path-finding process: (1) maximum number of node iterations or null for automatic, (2) TBD.
- Path_Options: Lists a series of options for the particular path method.

Basic built-in path-finding options:

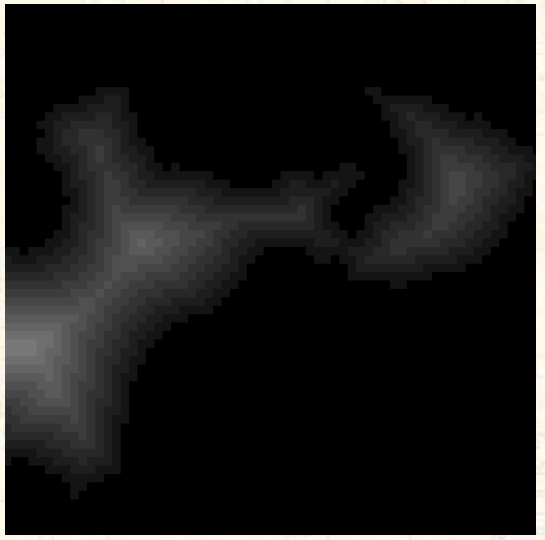
- PATF_A8: for direct grid methods, consider all 8 neighbouring vertices. The default is to consider only 4 vertices, leading to “blockier” paths.
- PATF_HDIST2I: for direct grid methods, use a scalar squared integer distance cost function (instead of the default Manhattan integer distance).
- PATF_HASHMAP: for A*-derived methods, use hash lookup tables. This increases memory roughly two-fold and yields significant speed gains.
- PATF_LAHEAD: for domain-graph methods, use a look-ahead topological optimization to generate paths across the domains.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

- Api note: Principia offers an unprecedented array of hi-perf topologic operations that can be called by any user procedure to enhance built-in capabilities:
  - Feature extraction
  - Domain subdivision (top)
  - Node scoring (internality score, bottom)
  - Graph operations (e.g. A*)
- Path finding, and its closely related operation – collision resolution – usually require some degree of user code to fully implement in production.





Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Collision management elements:

- Path-finding: Trace a path from A to B. Static or dynamic.
- Collision detection: Find objects that may collide and need action.
- Collision resolution: Adjust object paths and motion to avoid collision. Typically implemented in the user code with help of the powerful Principia Api. Dynamic.

Collision complexities:

- Interplay between dynamic path finding and collision resolution. In Ex04, path-finding is static, and collision resolution operates by modifying fully-formed paths and stopping objects.
- Collisions between group of objects and group path-finding.
- Dynamic obstacles causing collisions with between moving and fixed objects. Ideally, moving-to-fixed clearances are resolved during path-finding, and moving-to-moving clearances are managed by collision resolution (as here)

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Generating the world:

- Much of our procedure is the same as in Ex01-Ex03. The main novelty is the tags for generating path-finding pre-computed data structures and saving them in graphic form.

```
##define (PROC_WGEN_EX04) as <PROCWRL_ASM_RT3A>

   #tag Tile_Layers        =  (                              2 )
   . . .

   #tag Grid_TileSource    =  (                          0 )
   #tag Grid_ElevationF    =  (         F_GRIDHEIGHT_EX04 )
   #tag Grid_TypeExport    =  (              S_WTYPE_EX04 )
   #tag Grid_TypeImport    =  (                      NONE )
   #tag Grid_TypeAsmMap    =  (             S_DCT_GROUND ,   TC_CLIFF ,  TC_CLIFF ,  TC_FLATS ,          - )
   #tag Grid_TypeAsmMap    =  (              S_DCT_WATER ,   TC_WATER ,         - ,         - ,          - )

   #tag Path_Design        =  (      PATF_DGN )
@@@@tag Path_Design        =  (      PATF_SVD )
   #tag Path_PassYes       =  (      TC_FLATS ,   GRID_NODEREF ,   0.01 )
   #tag Path_PassNon       =  (      TC_WATER ,   GRID_NODEREF ,   0.01 )
   #tag Path_PassNon       =  (      TC_CLIFF ,   GRID_NODEREF ,   0.01 )
   #tag Path_PassNon       =  (      TC_LUMPS ,   GRID_NODEREF ,   0.01 )
   #tag Path_DomClearance  =  (          1.20 )
   #tag Path_DomMinSize    =  (            16 )
   #tag Path_ViewExport    =  (  S_PTYPE_EX04 )

##run (PROC_WGEN_EX04) with target (WRLD_EX04)
##save (S_WTYPE_EX04) in ("Files_Media/Ex04_ExMap_TerrainType_C.bmp") alpha in (Files_Media/Ex04_ExMap_TerrainType_X.bmp")
##save (S_PTYPE_EX04) in ("Files_Media/Ex04_ExMap_PathingType_C.bmp") alpha in (Files_Media/Ex04_ExMap_PathingType_X.bmp")
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Ex04 collision resolution logic:

- Use collision detection layer to identify pairs of moving objects that are about to interfere (based on radii+guard band).
- For newly found pairs, set the object that will make a detour (passer) and the object that will yield. An object that is fixed will always yield. The slower object of the pair will try to yield first.
- For the passer: modify its path by adding a point outside the collision sphere around the yielder. For the yielder: if it was moving, stop it. Once the passer is far enough, the yielder resume its motion on its original path.
- Mark passer and yielder using the high word of the generic AX_Object::StatusCode to prevent them from being re-injected in the resolution process at the next cycle. Clear the mark once they get far enough, to re-enable them to resolve other collisions.

Ex04 collision resolution limitations:

- Does not consider terrain type or other collisions when modifying the path of the passer. Not good for dense character situations.
- Does not consider dynamic obstacles.
- Does not handle collisions between three or more object simultaneously.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Collision detection layer

- Operates only on visible objects. Not very appropriate for non-moving object layers (although will work).
- Will not resolve collisions outside the viewport (in fact, what happens to objects that are not in scope is a subject that we have ignored so far. Production-grade pathfinding and object interaction resolution will be discussed in D108C:Ex02).
- Sector guard band ensures that objects that collide at the boundary between sectors will be detected.
- Object guard band helps detect collisions with moving objects sometime before they occur (so that the application can take early avoidance action) and helps plan avoidance. The guard band is the extension of radius for collision detection. It should be ~warning time x object speed. Setting it to zero will require immediate collision avoidance.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Collision detection user code:

- The resolution code section is short and sweet thanks to the world component built-in collision resolution methods.

```
void Crs_Ex04_Main (void* VArg)
BGN_ACTOR_CODE{

   . . . Initializations . . .

   /* Get a shortcut to the list of collisions detected and exit if there no need for action */
   ColN = ColOp->ColliderSet->N;
   ResN = ColOp->ColliderRes->N;
   if ((ColN==0x00)&&(ResN==0x00)) return;

   /* If both detected colliders are fresh, reroute them, mark them, and place them */
   /* in the resolved set. If the yielder object was moving, halt its motion by sig */
   for (k=0; k<ColN; k++) begin{
      ObjA = ColOp->ColliderSet->ePtr[k]; MovA = (AX_ObjInter*) ObjA->GetElement(OBJCID_BASEINTER);
      ObjB = ColOp->ColliderSet->aPtr[k]; MovB = (AX_ObjInter*) ObjB->GetElement(OBJCID_BASEINTER);
      TstA = ( BITMASK_TEST(ObjA->StatusCode, OBJSTAT_PPASS) || BITMASK_TEST(ObjA->StatusCode, OBJSTAT_YIELD) );
      TstB = ( BITMASK_TEST(ObjB->StatusCode, OBJSTAT_PPASS) || BITMASK_TEST(ObjB->StatusCode, OBJSTAT_YIELD) );
      if ((TstA==0x00)&&(TstB==0x00)) then{
         Rted = World->ColRes_PathReroute (MovA, MovB, ColOp->GBandObject, 0x00, NULL);
         if (Rted==0x02) then{
            // Can happen if stop points are near: Principia->TrapError("Static interference");
         ef (Rted=0x01) then{
            ColOp->ColliderRes->Add(ObjA, ObjB);
            if ( (BITMASK_TEST(ObjA->StatusCode, OBJSTAT_YIELD)) && (BITMASK_TEST(ObjB->StatusCode, OBJSTAT_YIELD)) ) then{
               Principia->TrapError("Path rerouting logical error");
            ef ( (BITMASK_TEST(ObjA->StatusCode, OBJSTAT_PPASS)) && (BITMASK_TEST(ObjB->StatusCode, OBJSTAT_PPASS)) ) then{
               Principia->TrapError("Path rerouting logical error");
            ef ( BITMASK_TEST(ObjB->StatusCode, OBJSTAT_YIELD) ) then{
               if (BITMASK_TEST(MovB->State, Swlk)) SigS->Send(ObjB->ID);
            ef ( BITMASK_TEST(ObjA->StatusCode, OBJSTAT_YIELD) ) then{
               if (BITMASK_TEST(MovA->State, Swlk)) SigS->Send(ObjA->ID);
            endif}
         }else{
            Principia->TrapError("Path rerouting failed");
         endif}
      endif}
   endfor}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B:Ex04 – WorldRT3A

Collision detection user code:

- **Despite its limitations, Principia enables us to handle collisions in few lines of code!**

```
   . . .

   /* If any two colliders in the resolved set have moved far enough */
   /* now, unmark them, restart the yielder and remove them from set */
   for (k=0; k<ResN; k++) begin{
       ObjA = ColOp->ColliderRes->ePtr[k]; MovA = (AX_ObjInter*) ObjA->GetElement(OBJCID_BASEINTER);
       ObjB = ColOp->ColliderRes->aPtr[k]; MovB = (AX_ObjInter*) ObjB->GetElement(OBJCID_BASEINTER);
       POINTDISTANCE_2D2_FAST(D,X,ObjA->X,ObjA->Y,ObjB->X,ObjB->Y);
       X = ObjA->RXY2 + ObjB->RXY2 + ColOp->GBandObject + ExtD*ExtD;
       if (D>X) then{
          if (BITMASK_TEST(ObjA->StatusCode, OBJSTAT_YIELD)) then{
             SigW->Send(ObjA->ID);
          endif}
          if (BITMASK_TEST(ObjB->StatusCode, OBJSTAT_YIELD)) then{
             SigW->Send(ObjB->ID);
          endif}
          BITMASK_DEL(ObjA->StatusCode, OBJSTAT_PPASS);
          BITMASK_DEL(ObjA->StatusCode, OBJSTAT_YIELD);
          BITMASK_DEL(ObjB->StatusCode, OBJSTAT_PPASS);
          BITMASK_DEL(ObjB->StatusCode, OBJSTAT_YIELD);
          ColOp->ColliderRes->ePtr[k] = NULL;
          ColOp->ColliderRes->aPtr[k] = NULL;
       endif}
   endif}
   ColOp->ColliderRes->RemoveAny(NULL, NULL, false);

END_ACTOR_CODE}
```


Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

D108B – WorldRT3A

- **It is time to take stock!**
  - **While none of the worlds seen in D108B is commercial production grade (yet), they cover the foundations for creating interactive 3D worlds:**
    - Layers and operations...
    - Terrain generation and tiles ...
    - Terrain rendering, DCT maps ...
    - Rendering of water and shadows ...
    - Viewers, cursors and placers...
    - Object selection and character integration...
    - Path finding and collision resolution...
  - **These start to add up to a solid basis for producing games and game engines!**
- **This will be pursued in D108C and Parts II-VI…**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005


Chapter 1:08
Core Worlds
Principia v3.0
Reference Series
(c) Western Star
Entertainment Ltd, 2003-2006

Chapter 8 – Performance Tricks

Rendering design

- Minimize shader changes. Most world contents should reuse few sets of illumination designs.
- Organize object layers based on material similarity. Do not incorporate materials in the objects, but place them at the head of the layer.
- Use multiple render detail settings in the world and object design. Partition tiles and use simple shaders in areas that do not require complex texture blending or VFX.

Data structure

- Calculate texture coordinates from position
- Do not generate XY data for all tiles. Use stream offsets and translate the XY portion of vertex data.
- Instance or assembly-bake large counts of identical objects such as trees or rocks.

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005

Chapter 8 – Developer Challenges

- **Create a third-person overhead view world with heavy atmospheric effects such as drifting fog, glowing swirling clouds, and reflective water. The challenge is on material layer management.**

Western Star Entertainment Ltd. PRINCIPIA Series (c) 2000-2005