colladaRenderUtil Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

1 Library Overview

Purpose and Features

The colladaRenderUtil library is the library that supports the rendering of COLLADA scenes.

This library can be used to easily display scenes defined within COLLADA on the screen.

Main Functions

The colladaRenderUtil library provides the following main features.

- Loading COLLADA
- Creating an instance of a COLLADA scene
- Setting the animation time of a scene instance
- Rendering a scene instance

Embedding in Program

Include colladarenderutil.h in source program.

When building the program, link libcolladarenderutil.a.

Sample Programs

Refer to the following sample programs that use the colladaRenderUtil library.

samples/sample_code/system/tutorial_collada_render_util/

This sample uses the colladaRenderUtil library to display COLLADA scenes.

Reference Materials

Refer to the following site for the COLLADA standard.

- "COLLADA - Digital Asset Exchange Schema for Interactive 3D"
  http://www.khronos.org/collada/

(The above reference destination has been confirmed as of November 17, 2011. Note that pages may have been subsequently moved or its contents modified.)

2 Usage

Basic Procedure

This section describes the basic procedure for rendering processing using the colladaRenderUtil library according to the following processing flow.

(1) Initialize `SimpleRenderer`
(2) Load COLLADA
(3) Create a scene instance
(4) Specify the animation time
(5) Render a scene instance
(6) Dispose of the scene instance
(7) Dispose of the loaded COLLADA
(8) Finalize `SimpleRenderer`

(1) Initialize `SimpleRenderer`

Call `colladaRenderUtil::SimpleRenderer::initialize()` to initialize `SimpleRenderer`.

```
SceGxmShaderPatcher *pShaderPatcher = …;
colladaRenderUtil::SimpleRenderer simpleRenderer;
simpleRenderer.initialize(pShaderPatcher);
```

(2) Load COLLADA

Call `colladaRenderUtil::SimpleRenderer::loadCollada()` to load the COLLADA file and save it in the Collada class.

```
colladaRenderUtil::Collada collada;
simpleRenderer::loadCollada(
        collada, "host0:graphics/model/astroBoy_walk.dae");
```

(3) Create a scene instance

Call `colladaRenderUtil::SimpleRenderer::instantiateVisualScene()` to create a scene instance.

```
colladaRenderUtil::InstanceVisualScene instanceVisualScene;
simpleRenderer.instantiateVisualScene(instanceVisualScene, collada);
```

(4) Specify the animation time

Call `colladaRenderUtil::SimpleRenderer::setTime()` to specify the animation time of the scene instance. Calling this function reproduces the scene at that time.

```
instanceVisualScene.setTime(0.1);
```

(5) Render a scene instance

Call `colladaRenderUtil::SimpleRenderer::render()` to render a scene instance.

```
// SceGxmContext *context;

// Set projection matrix and view matrix.
Vectormath::Aos::Matrix4 projectionMatrix = …;
Vectormath::Aos::Matrix4 viewMatrix = …;
```

```
// Set point light position and color.
Vectormath::Aos::Vector3 lightPosition = …;
Vectormath::Aos::Vector3 lightColor = …;

// Set InstanceVisualScene to be rendered and matrix for conversion from scene
// coordinate system to world coordinates.
colladaRenderUtil::SimpleRenderer::SceneInfo sceneInfo;
sceneInfo.instanceVisualScene = &instanceVisualScene;
sceneInfo.localToWorld = ..; // Scene Root to World matrix.

// Render
simpleRenderer.render(constext,
                      projectionMatrix,
                      viewMatrix,
                      lightPosition,
                      lightColor,
                      &sceneInfo,
                      1);
```

(6) Dispose of the scene instance

If the scene instance is no longer needed, call `colladaRenderUtil::SimpleRenderer::disposeVisualScene()` to dispose of the scene instance.

```
simpleRenderer.disposeVisualScene(instanceVisualScene);
```

(7) Dispose of the loaded COLLADA

If the loaded COLLADA is no longer needed, call `colladaRenderUtil::SimpleRenderer::disposeCollada()` to dispose of `Collada`. Before disposing of `Collada`, all `InstanceVisualScene` created with this `Collada` must be disposed of.

```
simpleRenderer.disposeCollada(collada);
```

(8) Finalize `SimpleRenderer`

If `SimpleRenderer` is no longer needed, call `colladaRenderUtil::SimpleRenderer::finalize()` to perform finalization. Before performing finalization, all `Collada` created with this `SimpleRenderer` must be disposed of.

```
simpleRenderer.finalize ();
```

Obtaining COLLADA Data

An application can obtain COLLADA data loaded with `SimpleRenderer`. Currently, geometries and images are the types of data that can be obtained. The procedures for obtaining each of these are described below.

Obtaining a Geometry

(1) Initialize COLLADA and load with `SimpleRenderer`

```
colladaRenderUtil::Collada colladaMesh;
colladaRenderUtil::SimpleRenderer simpleRenderer;
colladaMesh.initialize( "app0:graphics/mesh/temple.dae",
    simpleRenderer.m_loaderConfig);
```

(2) Obtain the geometries library from COLLADA

```
const colladaRenderUtil::collada::geometry::LibraryGeometries
    *libraryGeometries;
libraryGeometries = colladaMesh.getLibraryGeometries();
```

(3) Obtain the number of geometries from the geometries library

```
uint32_t numGeometries;
numGeometries = libraryGeometries->getNumGeometries();
```

(4) Obtain a geometry from the geometries library (index specification)

```
const colladaRenderUtil::collada::geometry::Geometry *geometry;
geometry = libraryGeometries->getGeometry(0);
```

(5) Obtain a mesh from the geometry

```
const colladaRenderUtil::collada::geometry::Mesh *mesh;
mesh = geometry->getMesh();
```

(6) Obtain triangles from the mesh (index specification)

```
const colladaRenderUtil::collada::geometry::Triangles *meshTriangles;
meshTriangles = mesh->getTriangles(0);
```

(7) Obtain the stride of the vertex data from the triangles

```
uint32_t strideCount;
strideCount = meshTriangles->getStride();
```

(8) Obtain the position offset value, normal offset value, and tex offset value from the triangles.

```
uint32_t positionOffset;
uint32_t normOffset;
uint32_t texOffset;
positionOffset =
    meshTriangles->getSemanticOffset(
    colladaRenderUtil::collada::SEMANTIC_POSITION);
normOffset =
    meshTriangles->getSemanticOffset(
    colladaRenderUtil::collada::SEMANTIC_NORMAL);
texOffset =
    meshTriangles->getSemanticOffset(
    colladaRenderUtil::collada::SEMANTIC_TEXCOORD);
```

Obtaining an Image

(1) Initialize COLLADA and load with `SimpleRenderer`

```
colladaRenderUtil::Collada colladaMesh;
colladaRenderUtil::SimpleRenderer simpleRenderer;
colladaMesh.initialize(
    "app0:graphics/mesh/temple.dae", simpleRenderer.m_loaderConfig);
```

(2) Obtain the images library from COLLADA

```
colladaRenderUtil::collada::Image::LibraryImages *libraryImages;
libraryImages = colladaMesh.getLibraryImages();
```

(3) Obtain the number of images from the images library

```
uint32_t numImages;
numImages = libraryImages->getNumImages();
```

(4) Obtain an image from the images library (index specification)

```
colladaRenderUtil::collada::Image::Image *image;
image = libraryImages->getImage(0);
```

3 Creating Data

This section describes how to create data to be handled by the colladaRenderUtil library. To export COLLADA data, use Maya 2009 and COLLADA for Maya 3.05C (http://sourceforge.net/projects/colladamaya/).

(The above reference destination has been confirmed as of November 17, 2011. Note that pages may have been subsequently moved or its contents modified.)

Exporting COLLADA Files

(1) Load plugins in Maya 2009

In Maya 2009, select **Window > Settings and Preferences > Plugin Manager**, and load the following files.

- COLLADA.mll
- COLLADAMaya.mll

(2) Export the COLLADA files

Select **File > Export All**.

Select **Options**, select **COLLADA exporter** for the file type, and export the files.

Using an Independent Shader

By default, a shader is provided when using `SimpleRenderer`, but an application can also use a shader created independently. This procedure is described below.

(1) Create and assign a Colladafx Shader material

Create a “Colladafx Shader” material and assign it to an object.

(2) Configure the shader

Display the properties of the created “Colladafx Shader” material, specify the “Vertex Program” and “Fragment Program” files, and configure the “Uniform” and “Varing” parameters.

(3) Export the COLLADA files

Export the COLLADA files using the procedure described in the “Exporting COLLADA Files" section.

(4) Position the gxp file

The output COLLADA file has a path to the cg file, as shown below.

```
Description example:
<profile_CG platform="PC-OGL">
<include sid="include" url="./shader/texture_shader_v.cg"/>
<include sid="include2" url="./shader/texture_shader_f.cg"/>
```

The compiled gxp file must reside in the same directory where the cg file is located. In the above examples, ./shader/texture_shader_v.gxp and ./shader/texture_shader_f.gxp must be prepared beforehand.

Note
The path of the shader in the COLLADA file must be a path usable by the target.
Specify an absolute path that can be read even by the target or a relative path from the COLLADA file.

(5) Prepare a derivative class of colladaRenderUtil::ShaderParameterManager

The values specified with Maya are passed automatically to the Uniform and Varing parameters of the shader by `SimpleRenderer`. The value you may wish to modify of the Uniform parameter during game execution can be changed dynamically with `colladaRenderUtil::ShaderParameterManager`.

To configure the Uniform parameter value of the shader, you must first define the derivative class of `colladaRenderUtil::ShaderParameterManager`, and then define the mapping of the Uniform parameter name and value of the independent shader within the derivative class. The following is an example of the definition of the derivative class.

Description example:

```
class MyShaderParameterManager : public
colladaRenderUtil::ShaderParameterManager
{
public:

MyShaderParameterManager();
virtual ~MyShaderParameterManager();

virtual int setNodeToSceneRootMatrix(
        sce::Vectormath::Simd::Aos::Matrix4_arg nodeToSceneRootMatrix );

virtual int setSceneRootToWorldMatrix(
        sce::Vectormath::Simd::Aos::Matrix4_arg sceneRootToWorldMatrix );

virtual int setViewProjectionMatrix(
        sce::Vectormath::Simd::Aos::Matrix4_arg viewMatrix,
        sce::Vectormath::Simd::Aos::Matrix4_arg projectionMatrix );

virtual int getParameterId(
        ShaderStage stage, const char *parameterName, uint32_t
        componentCount );

virtual const float* getSource(
        ShaderStage stage, int index );

virtual int getTextureParameterId(
        ShaderStage stage, const char *parameterName);

virtual const SceGxmTexture* getSourceTexture(
        ShaderStage stage, int index );

private:
......
}
```

The example below uses an independent shader, so use this as a reference.

- $(SCE_PSP2_SDK_DIR)/target/samples/sample_code/system/tutorial_shooting_game_trc_compliant/

For details on each function, refer to the reference document.

(6) Specify an instance of the derivative class when initializing colladaRenderUtil::SimpleRenderer

By specifying an instance of the derivative class when initializing `colladaRenderUtil::SimpleRenderer`, the parameter value specified with the derivative class can be passed to the shader.

```
colladaRenderUtil::SimpleRenderer renderer;
MyShaderParameterManager *shaderParameterManager;
uint32_t gpuHeapSize = (32*1024*1024);

shaderParameterManager = new MyShaderParameterManager();
renderer.initialize(gpuHeapSize, shaderParameterManager);
```

4 Supported Functions

The following COLLADA functions are supported by colladaRenderUtil.

- Visual scenes and instance visual scenes
  Only node, instance_geometry, and instance_controller are supported. Among node conversion elements, only translate, rotate, scale, and matrix are supported.
- Animation controller
  Only animation targeting node conversion elements within scenes is supported.
- Image
  For `SimpleRenderer`, only .gxt and .tga formats are supported.
- Geometry
  Only mesh and triangles are supported.
- Effects and materials
  Only Phong, Blinn, and Lambert of technique_common are supported. Only 2D textures are supported.