# Candera Engine 3D

## OpenGL ES 3.0 Support

### RenderDevice for OpenGL ES 3.0

In order to support OpenGL ES 3.0 a new RenderDevice.cpp has been added, which implements RenderDevice.h with OpenGL ES 3.0 functions. Additionally GlInclude.h, GlTraceMapper and GlTypeMapper have been adapted to support the split of one RenderDevice into several ones for multiple GLES versions. GlTraces are now all displayed under Gles device (instead of Gles20).

For current device packages OpenGL ES 2.0 is still used. If it is supported by a device, usage of OpenGL ES 3.0 can easily be configured in Device.cmake. Simply include Common/OpenGLES30 instead of Common/OpenGLES20 and add the following line to the CMake file.

```
CgiAddDefinitions( -DCGIDEVICE_OPENGLES_30 )
```

### Synchronization for OpenGL ES 3.0

The OpenGL ES 3.0 render device uses core OpenGL ES 3.0 fences for synchronization, instead of EGL extension EGL_KHR_fence_sync.

### OpenGL ES 3.0 Support for Linux OpenGLAdapter

OpenGLAdapter for Linux hosts now also supports OpenGL ES 3.0 interface (based on GLX and OpenGL 3.0).

### OpenGL ES 3.0 ETC2/EAC Texture Compression

OpenGL ES 3.0 introduces ETC2/EAC texture compression as part of the core standard. This enables all OpenGL ES 3.0 capable devices to reduce VRAM and memory bandwidth usage per texture at unnoticeable reduction of image quality, allowing either higher framerates or more and bigger textures used simultaneously.

Candera now supports this feature in the OpenGL ES 3.0 render device. For this purpose Bitmap class has been changed (see Changes in Bitmap Enumerations). The new Bitmap::PixelFormat enumerator has to be used for texture uploading.

**See also:**

- Import Images for information how to import ETC2/EAC compressed images into SceneComposer.

---

## Cloning Interface

In accordance to basic Unified Cloning Interface, the following changes have been applied to the affected classes in Candera 3D:

- CloneInternal protected interface was removed.
- Clone(Traversing) has become deprecated.
- Copy constructor was implemented or cleaned up.
- Assignment operator was moved to the private section, and its implementation removed.
- Clone() const function was added.

The cloning interface was applied to:

- Node and its descendants: Billboard, Camera, Group, Light, LineList, LodNode, Mesh, MorphingMesh, PlanarShadow, PointSprite, ReflectionCamera, Scene, StereoCamera.
- All other objects that are linked to node and descendants through shared pointers: Appearance (and MultiPassAppearance), Material, RenderMode, Texture, AbstractShaderParamSetter, GenericShaderParamSetter, ShaderParamSetter, Projection, GenericProjection, OrthographicProjection, PerspectiveProjection.
- RenderOrder, as its lifetime is controlled by Scene.

Special considerations:

- General. Clone for shared objects returns a shared pointer to the base shared class.
- General. Descendants of Node implement deprecated "Node* Clone(Traversing)" for backward compatibility.
- Billboard. Bounding box and bounding sphere are copied from the base class, not recomputed.
- Camera. CameraRenderStrategy is not linked to the clone anymore.
- LodNode. Clone contains the same number of levels as the original, but no level information is stored.
- Node. CopyFrom is provided to copy transformations, appearance and other such properties from this node to another one.
- PlanarShadow. Clone loses association to light.

- PlanarShadow. As defined by the Copy constructor (but not the assignment operator, previously used for cloning), the id of the clone is different from the one of the original.
- ReflectionCamera. Clone loses association to source camera.
- Scene. Scene may be cloned. The clone does not receive a render order.
- StereoCamera. ProjectionListener may be copied.
- ShaderParamSetter. Clone does not hold any data since data is handled by the application.
- AssetLoader nodes. They don't support cloning.

### Deep Cloning Interface

In accordance to basic Tree Cloning Strategies, the following support is provided for deep cloning in 3D as follows:

- TreeCloner is a specialization of TreeClonerBase.
- DeepNodeCloneStrategy provides a generic node strategy that clones the delegates the cloning of appearance and derived Nodes to specialized strategies.
- DeepAppearanceCloneStrategy provides deep appearance cloning. It clones its constituents by resolving shared instances.
- DeepCompositeGroupCloneStrategy, DeepLodNodeCloneStrategy, DeepStereoCameraCloneStrategy and DeepReflectionCameraCloneStrategy resolve node binding by use of TreeMatch.
- DeepCameraCloneStrategy resolves the sharing of projections.
- DeepSceneCloneStrategy clones the RenderOrder.
- DeepTreeCloner is a wrapper of DeepNodeCloneStrategy that provides an interface similar to TreeCloner.

---

## CompositeGroup

CompositeGroup is a new Node type that is usually created by AssetLoader and which contain:

- a list of AnchorPoints: descendants of the CompositeGroup that can be accessed directly from the CompositeGroup instance.
- a list of Widgets that act on any node within the CompositeGroup subtree.
- a list of AnimationPlayers that act on any node within the CompositeGroup subtree.

---

## Picking Test: Consistent Behavior for deep and flat Picking

In both cases Node::IsPickIntersectingGeometry only returns true for a tested node if it is hit by the intersection ray AND Node::IsEffectiveIntersectionTestEnabled is true.

---

## DeviceObject Listener

DeviceObject now supports adding listeners that are notified before and after the object is uploaded and unloaded and before it is destroyed.

---

## Upload to Multiple Contexts

Device Objects of a Node (and its descendants) can be uploaded to all render target contexts belonging to the cameras the nodes are in scope of.

**See also:**

> Node::UploadAll(). Furthermore, logical operators for intersection and union of Scope masks have been added to Candera::ScopeMask.

---

## ExternalTextureImage

Added ExternalTextureImage which is a texture image that allows wrapping external created native texture objects.

---

## Car Paint Shaders added

The following reference shaders have been added:

- RefCarbon / RefTransLight1Carbon
  Creates a car paint with typical carbon look.

- RefFlopCarPaint / RefTransLight1FlopCarPaint
  Creates a car paint with shiny, anisotropic two color look.

- RefMetalFlakesCarPaint / RefTransLight1MetalFlakesCarPaint
  Creates a car paint with shiny, sparkling look produced by metal flakes within specular highlight.

- RefFlopMetalFlakesCarPaint / RefTransLight1FlopMetalFlakesCarPaint
  Creates a car paint with shiny, sparkling look produced by metal flakes within specular highlight combined with anisotropic two color look.

---

Revision #5
Created 2023-03-02 08:17:05 UTC by Kanai Tomoaki
Updated 2024-05-30 02:25:57 UTC by Tsuyoshi.Kato