3D Render Order

Description

There are various reasons why sorting of objects to render is required:

- **Transparency**: Transparent objects must be rendered from back to front, while opaque objects must be rendered from front to back.
- **Performance**: Front to back order or advanced techniques like state sorting provide the best performance
- **Other**: Custom rank, Skybox, other background geometry, dialogs, pop-ups, etc.

Render Order: Involved Classes

Involved Render Order Classes

Candera::RenderOrder	Defines the sequence of nodes to render per scene and camera. Sorting is done according to following criteria: 1. Rank of render order bins: Candera::RenderOrder::SetBinRank. 2. Sorting of Nodes within bin (see below). Default bins are 'opaque' and 'transparent'. If no assignment is set, nodes are automatically assigned to opaque or transparent, according to the Node's transparency settings (RenderMode state).
Candera::RenderOrderBin	RenderOrder maintains a set of RenderOrderBins. A RenderOrderBin: • Contains nodes that have rendering enabled • **Candera::RenderOrderBin::SetOrderCriterion:** Sorts nodes according to applied Candera::OrderCriterion • **Candera::RenderOrderBin::SetRank:** defines the rank or order of render order bin. Nodes with smaller rank are ordered before nodes with greater rank. • Distinction between camera dependent and camera independent bins

Candera::OrderCriterion	Nodes within a bin are sorted according to an OrderCriterion. Predefined for **camera dependent bins:** • Candera::DistanceToCameraOrderCriterion (sorting from front to back for opaque bin) • Candera::ReverseDistanceToCameraOrderCriterion (sorting from back to front for transparent bin) Predefined for **camera independent bins, sorted by explicit rank values:** • Candera::RankOrderCriterion (lower render order rank is before higher render order rank.) • Candera::ReverseRankOrderCriterion (higher render order rank is before lower render order rank) • Candera::RenderStateOrderCriterion (takes in consideration the Node's Shader, Texture and RenderMode, whose changes are considered the most expensive)
Candera::Node	A node must be assigned to a bin, and if this bin is sorted by rank, the node must also specify it's rank inside the bin. • Candera::Node::SetRenderOrderBinAssignment: Defines bin assignment. If not set, then auto assignment is processed: Opaque or transparent, according to Node's RenderMode state • Candera::Node::SetRenderOrderRank: Defines render order rank that is utilized if the bin assigned to is sorted by rank
Candera::Scene	Candera::Scene::SetRenderOrder: One RenderOrder is exactly assigned to one Scene.
Candera::Renderer	Uses RenderOrder to render Nodes according to its defined sequence.

Default Render Order: Sorting by Distance to Camera

Default Render Order

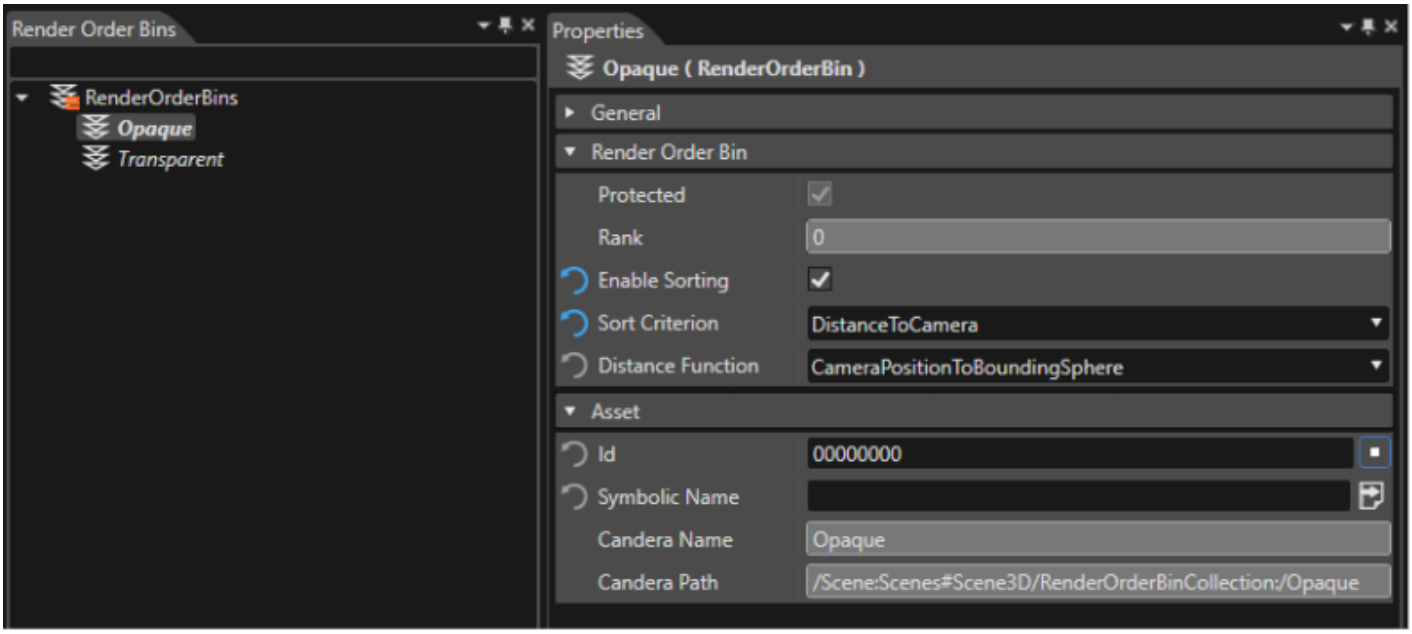
By default, all objects part of a scene are automatically sorted by Candera according to their render attributes into two predefined render order bins:

- **Opaque bin**: Opaque objects are rendered using Candera::DistanceToCameraOrderCriterion
- **Transparent bin**: Transparent objects are rendered using Candera::ReverseDistanceToCameraOrderCriterion

Both default bins are protected, which means that they are under control of Candera and cannot be deleted.

Predefined Render Order in SceneComposer

In SceneComposer, each 3D scene gets the predefined render order bins assigned by default:



Predefined Render Order in Candera

To apply the default behaviour with auto-assignment of transparent and opaque objects via Candera API, refer to following example:

```
// Create default render order with 2 default bins and max. number of 10 nodes each (opaque and
// Default rank for opaque bin is 10 and transparent bin is 20
static RenderOrder* renderOrder = RenderOrder::Create(10, 10);
// Assign render order to scene.
m_scene->SetRenderOrder(&renderOrder);
```

Fixed Render Order

To customize the default render order with a fixed render order, scene nodes must be assigned to custom render order bins sorted by explicit rank values.

- Using Candera::RankOrderCriterion, the scene will be rendered so that lower render order rank is before higher render order rank.
- When using Candera::ReverseRankOrderCriterion, higher render order rank is before lower render order rank.

Code Example: Fixed Render Order

As an example, create a render order with 3 bins and max. number of 10 nodes for O&T bin:

```
static RenderOrder renderOrder(3, 10, 10);
m_scene->SetRenderOrder(&renderOrder);
```

Create new render order bin with name "UserBin", rank "30", max. Number of 10 nodes:

```
renderOrder.CreateBin("UserBin", 30, 10);
```

Create dedicated order criterion which sorts nodes by its render order rank:

```
static RankOrderCriterion userCriterion;
```

Assign order criterion to UserBin:

```
renderOrder.SetBinOrderCriterion("UserBin", &userCriterion);
```

Assign nodes to dedicated bins:

```
m_mesh1->SetRenderOrderBinAssignment("UserBin");
m_mesh2->SetRenderOrderBinAssignment("UserBin");
```

Set dedicated render order rank for nodes:

```
m_mesh1->SetRenderOrderRank(2);
m_mesh2->SetRenderOrderRank(1);
```

Performance Optimized Render Order

Candera::RenderStateOrderCriterion implements a proof of concept OrderCriterion used for batching nodes to minimize render state changes. It takes into consideration the nodes shader, texture and render mode, whose changes are considered the most expensive.

- After sorting all the nodes using the Candera::OrderCriterionValue, rendering is done in batches of nodes with the same shader.

- Inside a batch of nodes with the same shader, the rendering will be done in batches of nodes with the same texture.
- Inside a batch of nodes with the same shader and texture, the rendering will be done in batches of nodes with the same blending enabled value.

The batching continues until the least expensive state change (CullingMode).

> - Usage of Candera::RenderStateOrderCriterion does not guarantee best batching algorithm, which should be implemented after benchmarking the target render device's state change costs.
> - RenderStateOrderCriterion groups the nodes only by their first appearance and texture. Nodes with multi-pass rendering (MultiPassAppearance), or with multiple texture units, most probably will break the batches and cancel some performance gains.
> - CANDERA_RENDER_STATE_CACHING_ENABLED should be defined to use Candera's render state caching mechanism.

Revision #9
Created 2023-02-21 06:21:26 UTC by Tsuyoshi.Kato
Updated 2026-08-19 02:18:31 UTC by Tsuyoshi.Kato