# Candera Engine 3D

## Shader Uniform Cache Handles

If a shader uses more than a single uniform, a Shader::UniformCacheHandle is the fastest way to access uniform locations. After a uniform cache accessor was registered in a shader with an array of uniforms to be accessed, the resulting uniform cache handle can be used to retrieve uniform locations in constant time. It does not matter if the uniform is an auto or a custom uniform.

Candera uses this mechanism to get rid of uniform location related (string) compares.

---

## Render Order Criterion

The RenderOrderCriterion class extends the OrderCriterion class to enable the user to control preparation and sorting of render order bins. By default it is now used to skip sorting if the order of the bin has not changed since the last pass.

If an OrderCriterion was assigned to the RenderOrderBin, everything is processed as usual. If a RenderOrderCriterion was assigned to the RenderOrderBin, RenderOrderBin::Sort calls RenderOrderCriterion::PrepareRenderOrderCriterionValues, RenderOrderCriterion::HasOrderChanged, and depending on the result of the previous function, RenderOrderCriterion::Sort.
Using the new interface enables the user to have more control over how sorting is performed, and if sorting needs to be performed at all.

Candera's DistanceToCameraOrderCriterion, ReverseDistanceToCameraOrderCriterion, RankOrderCriterion, as well as RenderStateOrderCriterion have all been changed to use the new RenderOrderCriterion interface.

---

## Batch Order Criterion

The BatchOrderCriterion is a RenderOrderCriterion implementation that sorts bins by either VertexBuffer+Shader+RenderMode+Distance to camera, or by VertexBuffer+Appearance+Distance to camera to increase drawcall batching (i.e. implicit

geometry instancing) rates. It also takes advantage of the new RenderOrderCriterion feature to only sort bins when the order has changed.

---

## Extended Geometry Instancing Support

Classes derived from CameraRenderStrategy can now utilize implicit geometry instancing (i.e. drawcall batching) by passing a flag to its base class. The BreakNodeCameraRenderStrategy has been updated to use it by default.

The PointSprite class is now recognized as being batchable, i.e. point sprites will automatically be picked up by drawcall batching (geometry instancing) if possible.

---

## Render Order Usage Change

The render order was previously associated with the Scene. With this version it has been moved to the Camera. Every camera requires its own render order now. This was done to be able to skip sorting of previously sorted render order bins.

The following functions were declared deprecated:

```
    void Scene::SetRenderOrder(AbstractRenderOrder* renderOrder);
    AbstractRenderOrder* Scene::GetRenderOrder();
    const AbstractRenderOrder* Scene::GetRenderOrder() const;
```

---

## PointSprite Interface and Behavior Changes

Point sprites share a single vertex buffer now, unless a custom vertex buffer is set for an instance of a point sprite by the user. Therefore PointSprite::GetVertexBuffer() returns a const pointer in order to avoid alteration of the shared vertex buffer. The color of a point sprite is now determined by the diffuse color of the material to establish consistent usage of the auto uniform that was used by the point sprite shader. GenericShaderParamSetter::SetPointSpriteActivationEnabled() must be set to true, when using it with a PointSprite.

The following functions were deprecated:

```
    void PointSprite::SetColor(const Color& color);
    const Color& PointSprite::GetColor() const;
```

---

## MorphingMesh Changes

Just like the PointSprite, the MorphingMesh was changed to establish consistent usage of its auto uniform. Therefore GenericShaderParamSetter::SetMorphingMeshActivationEnabled() must be set to true, when using it with a MorphingMesh.

---

## GenericShaderParamSetter Changes

The GenericShaderParamSetter now handles the auto uniforms of the PointSprite and the MorphingMesh.

The following functions were added, and must be used in conjunction with point sprites and morphing meshes:

```
    bool GenericShaderParamSetter::IsMorphingMeshActivationEnabled() const;
    void GenericShaderParamSetter::SetMorphingMeshActivationEnabled
(bool enableMorphingMeshActivation);
    bool GenericShaderParamSetter::IsPointSpriteActivationEnabled() const;
    void GenericShaderParamSetter::SetPointSpriteActivationEnabled
(bool enablePointSpriteActivation);
```

---

## Renderer Statistics Improvements

Renderer::Statistics now also exposes the total number of queries and occlusion query results for the last render pass.

---

## Generation of index buffers update.

Math3D::CreateVertexBufferWithIndexGenerated is now extended with an argument specifying the minimum size of one index (8, 16 or 32 bit).

---

Revision #3
Created 2023-03-02 06:13:28 UTC by Kanai Tomoaki
Updated 2023-06-15 04:16:32 UTC by Tsuyoshi.Kato