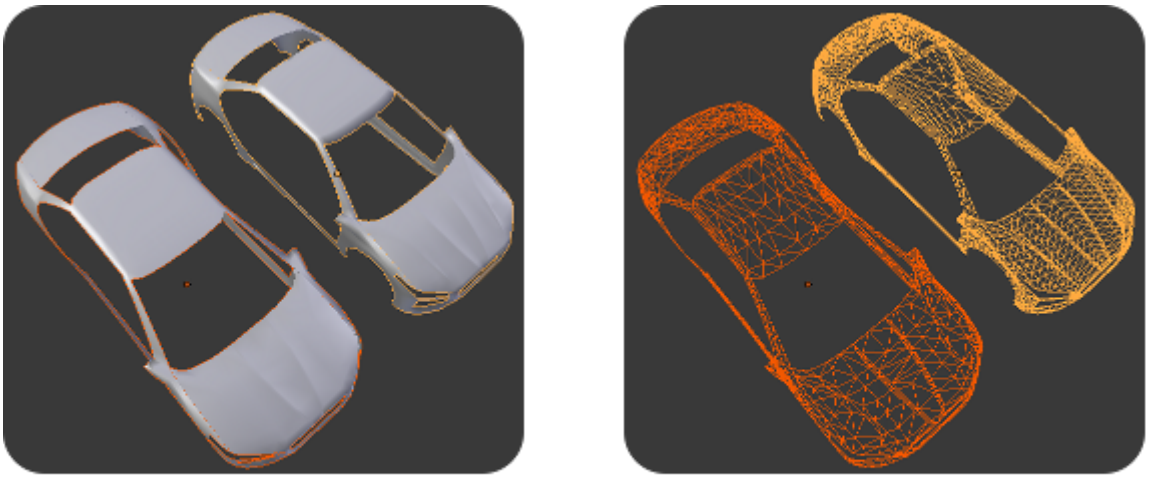
Optimization at Application Design Level

This section discusses performance optimizations in the scope of 3D scene modelling and application design based on CGI Studio Candera Engine.

- Simplify Meshes
- Optimize Number of Nodes
- Draw Visible Objects Only
- Use LOD(Level of Detail) Management
- Imposters
- Lighting
- Avoid Buffer Clearing
- Disable Depth Test and Adjust Depth Range
- Disable Blending If Not Required
- Minimize Render State Changes
- Textures

Simplify Meshes

Reduce the number of vertices in a mesh in order to improve runtime and memory performance. If viewing angle to a mesh is rather steady, it is a preferred approach to maintain higher amount of vertices at the silhouette and mainly reduce vertices in front view.



Example: Car chassis reduced from 4300 to 1300 faces retaining a similar visual quality if the car is shown from a medium range distance.

> A strongly simplified mesh does not necessarily lead to a diminished quality of visual appearance, if the level of detail matches the use case rendered (See also LOD Management).

Optimize Number of Nodes

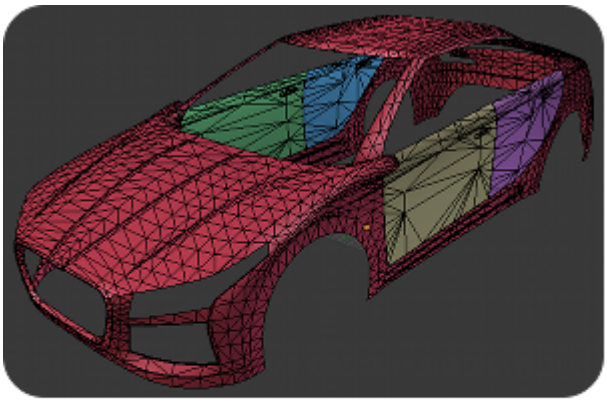
A reduced number of nodes lead to fewer node-related render state activations that have to be processed. Thus, whenever geometry can be safely combined to a single node like e.g. a mesh, then this way should be favoured. For instance a tire and a rim form a logical group sharing the same transformation. In that case the designer can unify separate geometries in a single mesh with the aid of a digital content creation (DCC) tool.
Consequently, the rim and tire do their texture lookup in a single texture, which further leverages performance gains by texture caching.

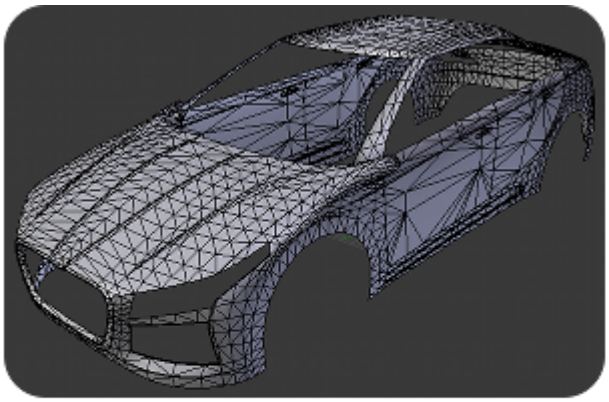


Example: Low-polygon wheel model consisting of tire, rim and brakes. The textures have been combined in a single texture atlas instead of using 3 separate textures.

See below crucial prerequisites for combining Nodes:

- Nodes share same transformation (position, rotation, scale).
- Nodes share the same Appearance (textures, shaders, material, render-mode, render-order)
- Nodes are within the same render-order bin.





Example: Instead of having 5 separate nodes for the chassis and doors, they can be combined into a single mesh if they don't need to move independently (i.e. the doors can't open) and have the same appearance (i.e. a car-paint).

Draw Visible Objects Only

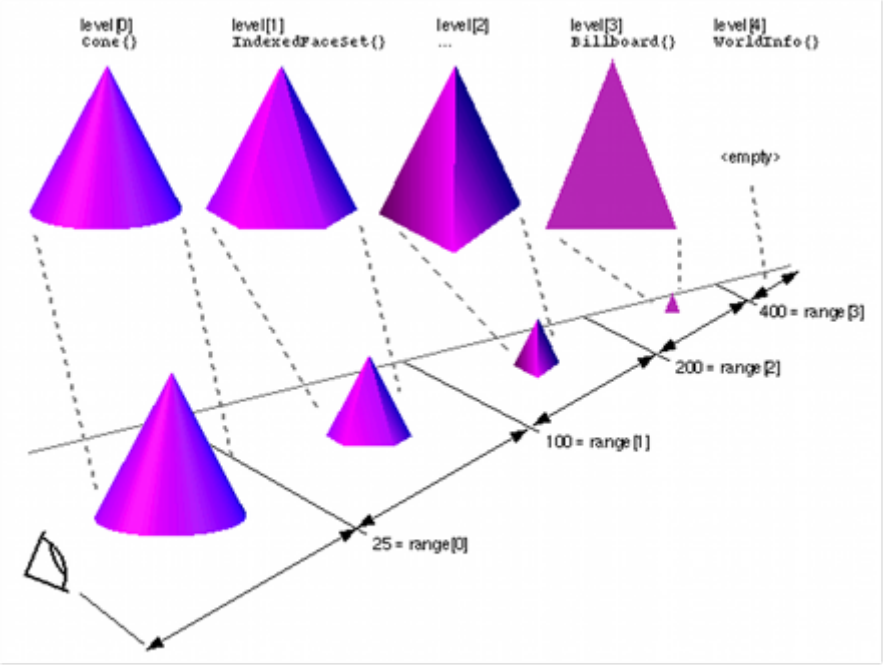
Nodes, even when attempting to render them, might be not be visible in various circumstances. Nevertheless, if no countermeasures are taken, they have to pass the render pipeline and are quite often discarded at a very late stage. To minimize computations within the render pipeline, geometry shall be discarded as soon as possible, if it is not visible at all.

Use following Candera concepts to exclude invisible objects from effective rendering earliest possible:

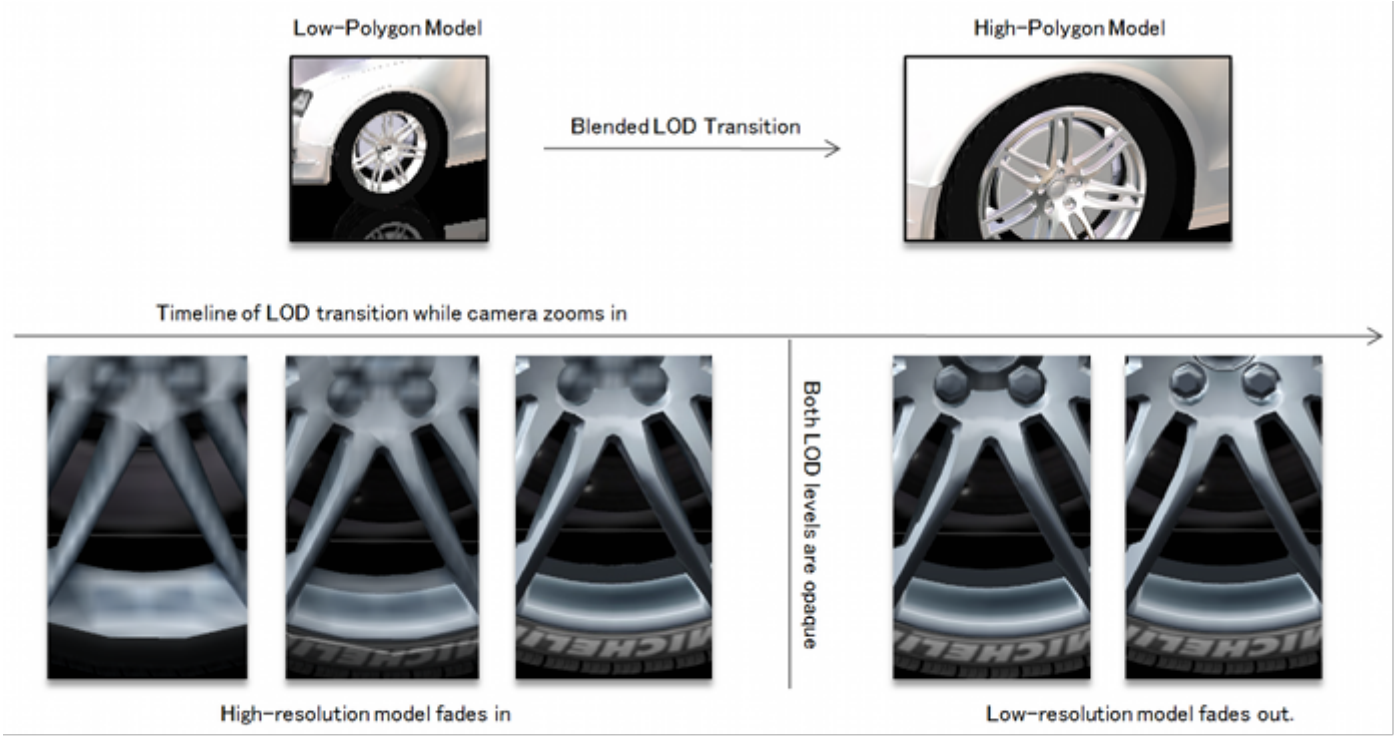
- Disable rendering of nodes manually (see `Node::SetRenderingEnabled(false)`).
  Example: Consider to disable nodes with a very low alpha value from rendering.
- Enable Viewing Frustum Culling in the Camera, to cull objects out of viewing frustum
- Use Backface Culling in the RenderMode to cull faces not visible.
- Use custom defined visibility and light culling via Candera scopes.
- Use Candera RenderOrder to sort opaque objects from near to far distance to the camera, which increases number of fragment culling according to depth-test-fails.

Use LOD(Level of Detail) Management

Performance gains can be achieved by using simplified models with less complex geometry, texture, materials and shaders for objects that need less perceptional detail level. E.g. objects that appear farer from the camera may reduce richness of detail without a noticeable decrease in visual perception.
The same applies for instance to objects that are less illuminated, moving, or in peripheral viewing frustum.



Multiple levels of simplification until the object is just a flat image



Low- and high-polygon model automatically blended and exchanged based on camera distance.

Candera supports comprehensive, multi-purpose level-of-detail functionality including discrete and blending transitions.

Please refer to the Tutorials CGI Studio Application Development Tutorial

Imposters

Billboards or point sprites are quite often used as "imposters" pretending to be a 3D geometry by showing a 2D image that is always facing the camera.

Candera supports both, Billboards and Point Sprites to achieve three-dimensional impressions by showing camera aligned two-dimensional images.

In order to relief distinction between Billboards and Point Sprites see following comparison:

- Billboards support rectangular dimension and non-uniform scale.
- Billboards support different rotation techniques (see Alignment), whereas Point Sprites are always screen aligned.
- Point Sprites have a performance advantage due to less geometry (one instead of 4 vertices).

Overall recommendation: Use Point Sprites for spherical shapes like particles, lens flares, sparkles, dust which are screen aligned. \ Further, use Billboards for world up oriented clouds, text, or yaw axis aligned trees, and signs, etc.

Lighting

Use as few *dynamic* light sources as possible. Light sources in order of their impact on rendering performance (fastest first):

- ambient light
- directional light
- point light
- spot light

Often the ambient light computation can be omitted by already including the ambient light factor in the ambient material of the affected node. Bake light into texture for static lighting. Use multi-texture effects like *lightmaps*, *spheremaps* for certain light effects to achieve high detailed pixel lighting on simple surfaces without dynamic light sources.

Avoid Buffer Clearing

Clear only buffers that need to be cleared.

- Color buffer: The color buffer does not have to be cleared if the scene is known to cover all of it.
  For instance, this is the case if the background image or a sky map covers the whole surface.
- Stencil buffer: Only needs to be cleared if it is actually used.
- Depth buffer: If depth test is enabled.

Disable Depth Test and Adjust Depth Range

Disable Depth Test if Not Required

If it can be easily predicted that an object is not occluded by other objects then disable depth test (See Render-Mode).

Adjust Depth Range

As the float resolution in depth buffer is highly limited, the depth range should be as small as possible to reduce depth-fighting artifacts. Further, geometry beyond the near and far plane can be discarded in the clipping stage.

Disable Blending If Not Required

Disable blending for opaque objects, as blending is a demanding computation (See Render-Mode).

Further, if blending for translucent objects does not require separate alpha blending (common case), set source blend factor to One, destination blend factor to Zero, and operator to Add (default settings in RenderMode). This setting allows modern GPUs to ignore the alpha blending equation.

> Within the lifetime of an object, there may be periods where blending is needed (fade-in/out) and periods where blending is not needed. In such cases take care to enable / disable blending dynamically as appropriate.

Minimize Render State Changes

Minimize render state changes within a single frame. A render state change often comes with a performance penalty, as a render state change can require complete rendering pipeline to be flushed.

Following subchapters explain how render state settings and changes can be minimized when using Candera.

Enable Candera Render State Caching

Enabling this Candera feature in the CMake build system configuration means that redundant render state settings are not propagated to graphic device driver.
Another benefit is that render state queries are not processed by graphic device driver, either.

Render Objects with same Rendering State

Render objects together that use the same rendering state, e.g.:

- Objects using the same texture (avoids flushing the texture cache)
- Objects using the same shader program (avoids changing shader programs)
- Objects using similar render modes

Use the Candera render order concept to take influence on the sequence of render operations according to custom criteria. For further details refer to Tutorials CGI Studio Application Development Tutorial

Scoping

If using scopes, scene graph nodes can form conceptual groups independent of the scene graph hierarchy, according to arbitrary

Textures

Draw Objects Sorted by Texture

This way the texture cache isn't flushed between objects. See also above, "Render Objects with Same Rendering State".

Texture Cache

Consider the size of the texture cache when defining textures. Cache misses can slow performance down appreciably. The total size needed by textures depends on their number, dimensions, and bit-depth; ideally, this size should be smaller than the cache size.

Use Texture Compression

If a texture compression feature is available, turning it on and using it is likely to improve performance.

Etc2/Eac compressed textures are typically uncompressed during texture lookups by the GPU for free. They save memory and increase the performance of the texture cache by reducing bandwidth. The following Etc2/Eac formats are supported by Candera on OpenGL ES3.0 platforms:

Compressed Bitmap::PixelFormat	Uncompressed generic format	size ratio compressed to uncompressed
Etc2CompressedRgbPixelFormat	RGB	1/6
Etc2CompressedSrgbPixelFormat	sRGB	1/6
Etc2Alpha1CompressedRgbaPixelFormat	RGBA	1/8
Etc2Alpha1CompressedSrgbaPixelFormat	sRGBA	1/8
Etc2EacCompressedRgbaPixelFormat	RGBA	1/4
Etc2EacCompressedSrgbaPixelFormat	sRGBA	1/4

EacCompressedUnsignedRPixelFormat	A	1/2
EacCompressedSignedRPixelFormat	signed A	1/2
EacCompressedUnsignedRGPixelFormat	LA	1/2
EacCompressedSignedRGPixelFormat	signed LA	1/2

Note: 'A' stands for Alpha (uncompressed equivalent format: AlphaUnsignedBytePixelFormat), and 'LA' for Luminance Alpha (uncompressed equivalent format: LuminanceAlphaUnsignedBytePixelFormat).

If texture compression is supported, avoid uncompressed textures as much as possible. E.g. 1 uncompressed RGB texture takes as much space as 6(!) Etc2CompressedRgb textures. Plus uncompressed textures decrease performance.

There a several compressed RGBA formats to choose from, which primarily affect the quality of the Alpha channel. If the alpha channel is only used to indicate fully opaque/transparent regions, then Etc2Alpha1Compressed formats are the best fit. If quality of transparency in Etc2EacCompressed textures is lacking, a combination of a compressed RGB texture together with a compressed single channel Alpha texture (EacCompressedUnsignedR/EacCompressedSignedR) should be favored over an uncompressed texture, because the 2 compressed textures need less memory and bandwidth than an uncompressed one.

Signed compressed textures have the capability to express 0 exactly in every pixel of the compressed texture. This is useful for sharp transparent edges, or for normal maps that are encoded in the EacCompressedUnsignedRG format.

Use Mip Maps

For objects that occupy static screen size, try to fit texture to rendered object size. For objects that change size on screen use Mip-Mapping to avoid texture aliasing effects and to improve runtime performance. Mip-Mapping consumes little more memory (max 1/3 more than the biggest bitmap).

Texture filter MipMapNearest consumes less time than MipMapLinear, which processes a trilinear filtering.

Combine Textures

Associated segments, for example rims and tires, should be combined into one texture. The different segments (nodes) share the same texture, but each of them refers to a different area within the texture. This way, the texture only needs to be activated once for multiple nodes.