Cga-LoD – User Manual

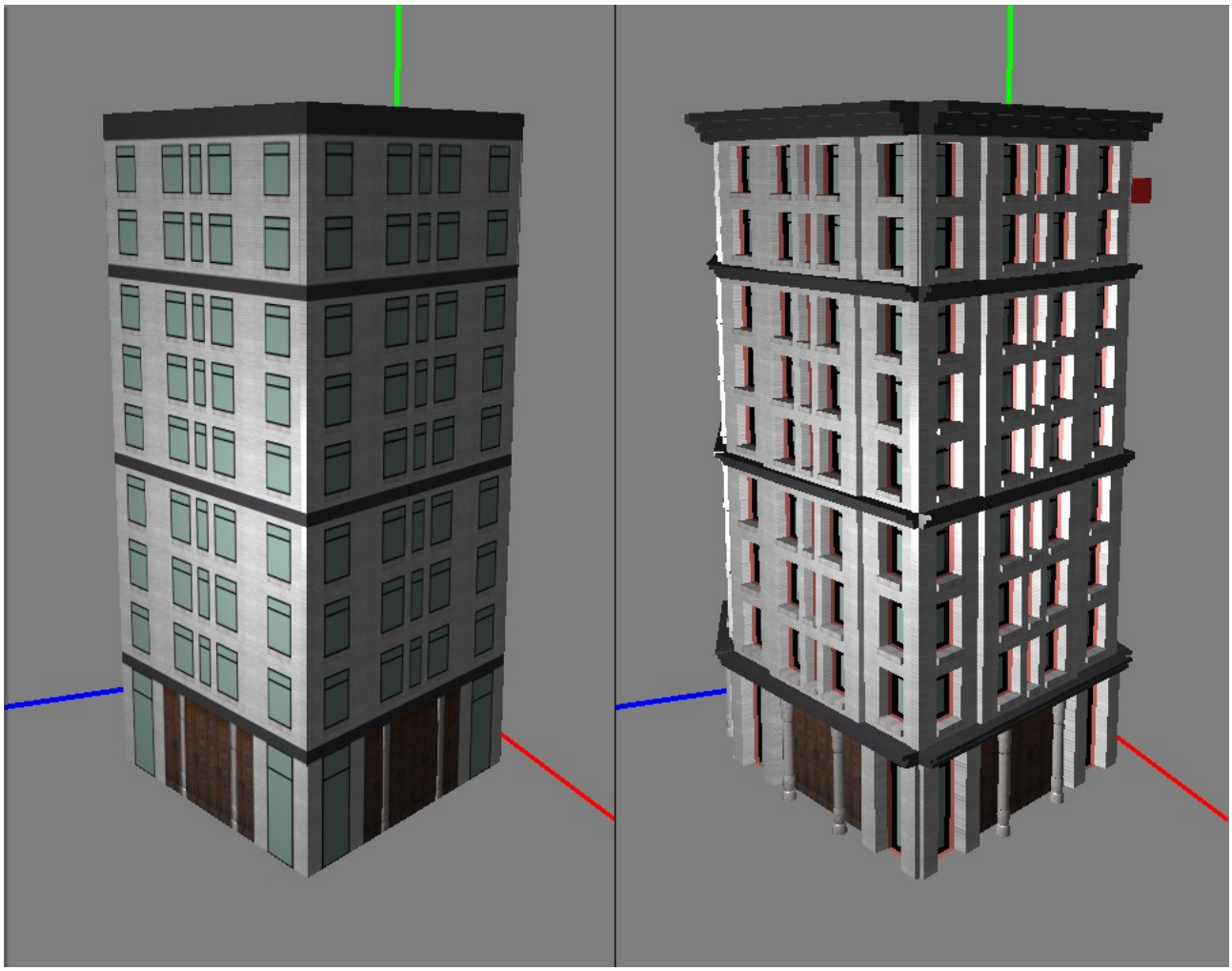

Figure 1: Cga-LoD is used to create LoD models of original models. The original models are those exported by Markus Lipp's Cga program.

Introduction

The Cga-LoD program is used to create LoD models out of models created with Markus Lipp's Cga program. The LoDs are based on bounding boxes, rather than mesh simplification.

The Cga program creates architectural models based on hierarchical rules. The Cga-LoD program now uses the hierarchy of each model to create the LoD models. Basically parts of the hierarchy will be replaced by oriented bounding boxes enclosing the geometry of this parts. This bounding boxes will them be texturized and used instead of the original geometry.

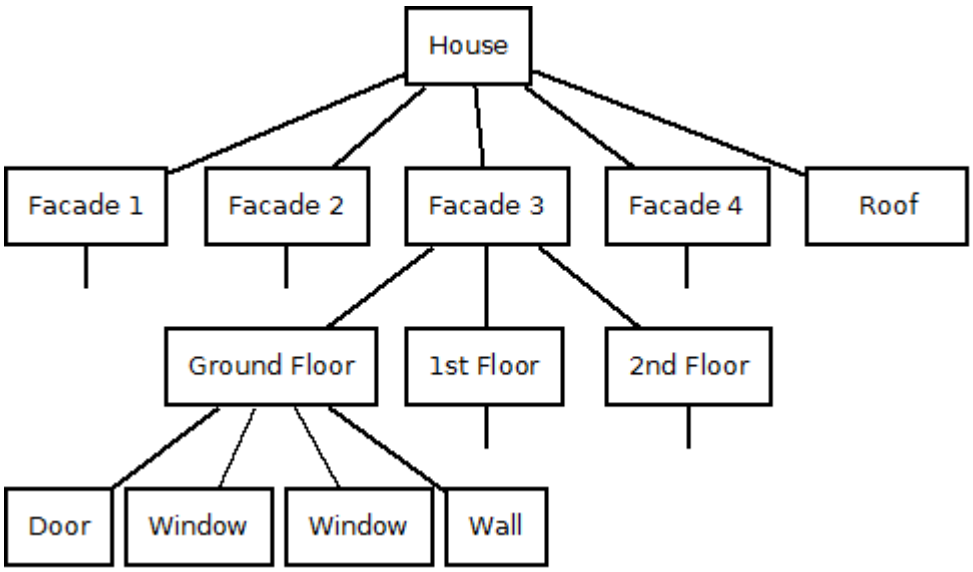

Figure 2: An example of how the hierarchy of an Cga model could look like.

Controlling the Camera

The basic method to control the camera is the keyboard (**ASDW** moves the camera) and the mouse (**left mouse button pressed** rotates the camera).

An additional way to control the camera are the **Get Distance** and **Set Distance** methods in the CGA section. With this methods the distance of the camera to the loaded mesh can be changed directly.

Another method can be found in the Options section. With **Camera Position** and **Set Position** it is possible to move the camera to one of 26 specified positions relative to the mesh.

Loading and Saving Files

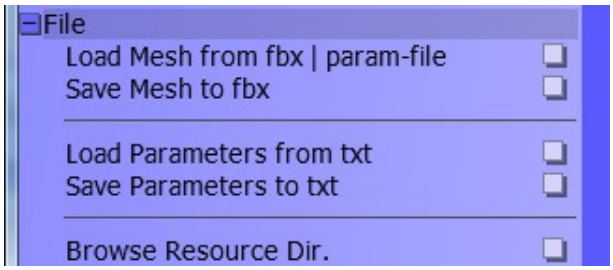

Figure 3: The controls of the "File" block allow to load models and to store LoD versions of models.

Cga-LoD can load and save two kind of files:

- FBX model files and
- Parameter files (human readable text files)

The loaded fbx files will come from Markus Lipp's CGA program [exported with "Export as FBX (+ Hierarchy)"], but Cga-LoD is also able to load the fbx files it has saved itself (although with this

files no more LoD creation will be possible).

The parameter files are human readable text files and contain:

- the filename of an fbx file
- all parameters of the CGA sections
- the current camera information.

Load Mesh from fbx | param-file

With 'Load Mesh from fbx | param-file' we can directly load an fbx mesh, but it is also possible to specify a parameter file, then the fbx mesh specified in the parameter file will be loaded and then all parameters in this file will be set.

Save Mesh to fbx

'Save Mesh to fbx' is used to save the created LoD models into fbx files. Additionally to the fbx file the following files will be created:

- Images containing the textures of the LoD model
- a parameter file
- a preview screenshot (the preview will be taken from the last rendering seen in the window)
- a statistics file containing the information of the Statistics section

Load Parameters from txt

'Load Parameters from txt' will load all parameters except the fbx file. If you want also to load the fbx file use 'Load Mesh from fbx | param-file' instead.

Save Parameters to txt

'Save Parameters to txt' creates a parameter file without saving the LoD model.

Browse Resource Dir.

'Browse Resource Dir.' allows to change the directory where the program will search for textures when a mesh is loaded. As long as the resource directory doesn't change, it just has to be selected once (the selected directory will be remembered for any further start of the program).

Creating a Level-of-Detail Model

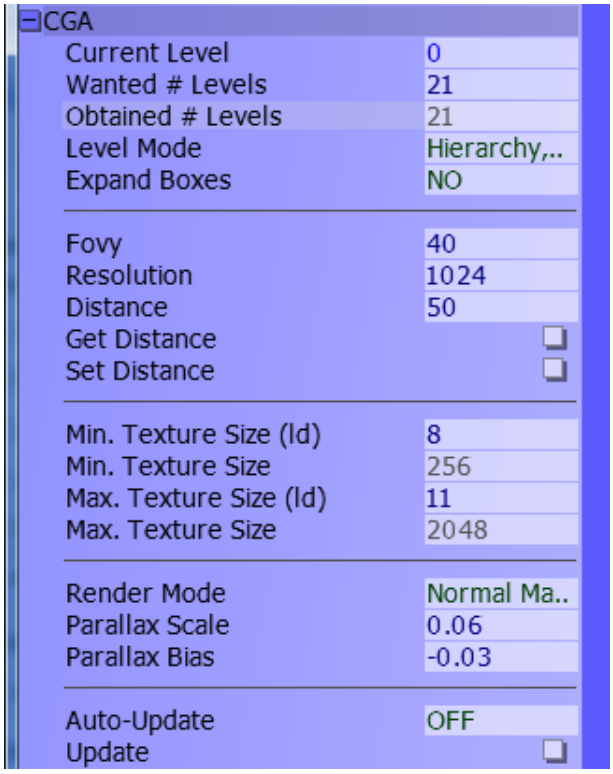

Figure 4: The controls of the "CGA" block are used to create the LoD model.

Cga-LoD performs several steps to create a LoD mesh from the original mesh. The controls in the CGA section are grouped into five blocks, the first three correspond to the steps of the LoD creation. The five groups can be called:

- Geometry
- Textures
- Texture Coordinates
- Rendering
- Update

Geometry

When we create a LoD mesh we first need to create new geometry based on the original mesh. Cga-LoD uses textured oriented bounding boxes (OBB) to approximate the geometry underneath.

The original mesh is stored with information about the hierarchy of the mesh (the hierarchy is a tree, directly related to the rules used in Markus Lipp's CGA program). Cga-LoD uses the hierarchy information to determine which parts of the hierarchy to approximate by an OBB.

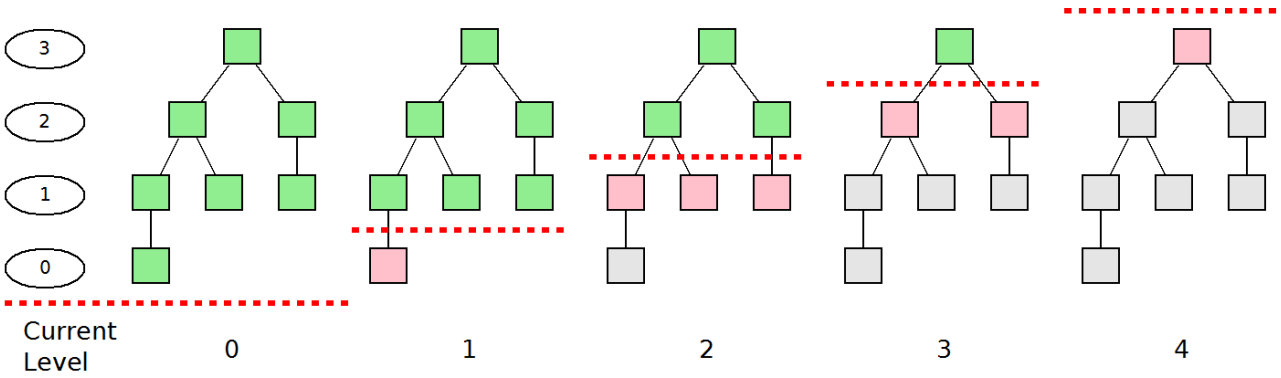

Figure 5: The bounding boxes are selected by comparing the level value of each node in the hierarchy to the current level. Green nodes use the original geometry, red nodes use bounding boxes and grey nodes are not used at all (but for each grey node there is a red node)

For each node in the hierarchy a level value is calculated. With the **Current Level** we tell the program which parts (sub-trees in the complete hierarchy) to approximate by an OBB. If the level value of a node is greater or equal then the Current Level, the original geometry of this node is used and all children of this nodes will be processed. But if the level value is smaller than the Current Level then the node (plus all its children) will be approximated by an OBB.

The minimum value for the level value is 0, therefore Current Level 0 always results in the original mesh without any Level-of-Detail. If the Current Level is the maximum level, then the resulting geometry will be just the OBB of the whole original mesh.

The maximum value for the level value is specified by the '... # Levels'. **Wanted # Levels** tells the program how many levels we would want and **Obtained # Levels** is the number of levels the program actually allows. This distinction between this two is necessary since some methods to calculate the level value need to specify the maximum value by them self (at he moment only the 'Hierarchy, ...' methods do this).

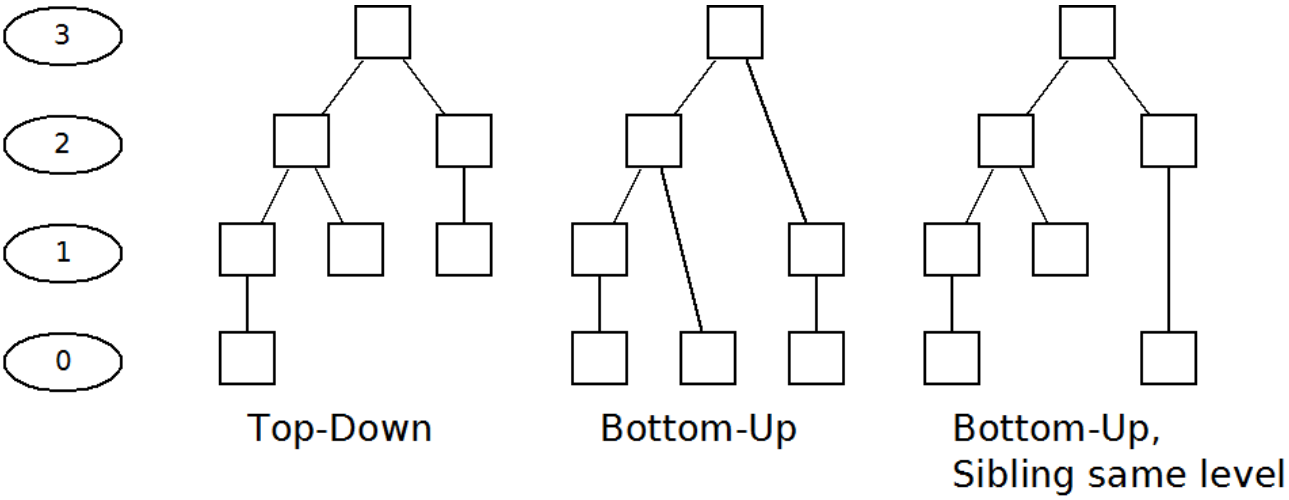

Figure 6: The 'Hierarchy, ...' Level modes. The ellipses show the level value of each node.

There are several methods to calculate the level value. With the **Level Mode** we can choose between this methods. The methods are based directly on the hierarchy or on the OBBs.

- Hierarchy, Top-Down
    - the root-node of the hierarchy gets the maximum value

    - each child node gets the value of its parent minus 1
- Hierarchy, Bottom-Up
    - all leafs of the hierarchy get the value 0
    - all inner nodes get the maximum of their children values plus 1
- Hierarchy, Bottom-Up Sibling-Same-Level
    - as the previous method
    - additionally all sibling-nodes will be raised to the same level
- Bounding Box, ...
    - A value v is calculated for each node (specified by this methods name)
    - The maximum and minimum v is calculated
    - Each v is interpolated between minimum and maximum
    - This interpolation (value between 0...1) is multiplied with the maximum level value

Figure 7: The 'OBB, ...' level mode. For each node a value v is calculated (how this value is calculated is based on which 'OBB, ...' is selected), then the Wanted # Levels are used to calculate the level value for each node by interpolating the v values.

The program uses two kind of OBBs. The first ones directly come from the CGA program and are usually the better choice for LoD creation. But this OBBs might be smaller than the geometry (so they aren't actually bounding boxes) and therefore there is the second kind of OBBs. This second OBBs are the same as the first ones, just expanded so the geometry fits into the boxes. With **Expand Boxes** it is possible to use this expanded boxes.

Textures

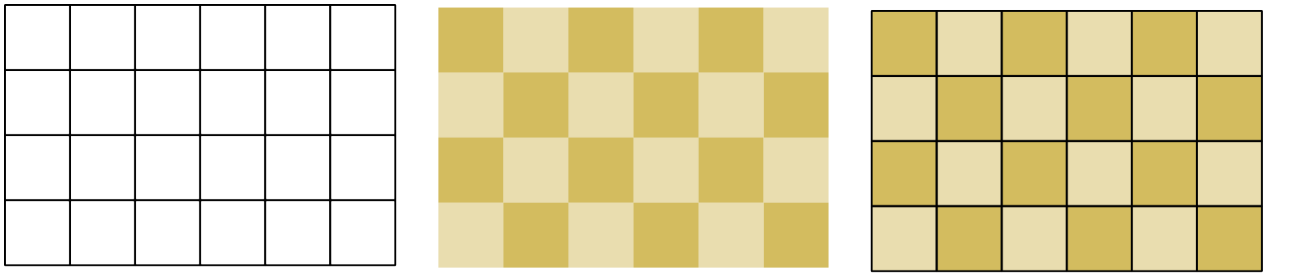

Figure 8: Mapping of texel to the screen allows to calculate the needed size of the textures.

After selecting the OBBs it is necessary to texturize them. Therefore is is necessary to calculate the size of the texture needed for the sides of each OBB. This calculation is done so that each texel of the textures will be projected 1:1 onto a pixel of a virtual screen.

For this projection a camera with a **Fovy** and a certain **Distance** is used. As long as this camera stays at the same distance (or a greater one) and keeps the same Fovy (or a greater one) the 1:1 projection holds. Since we have a virtual screen it is also necessary to specify a **Resolution** for this screen (we use square texel, so just the resolution in y direction has to be specified).

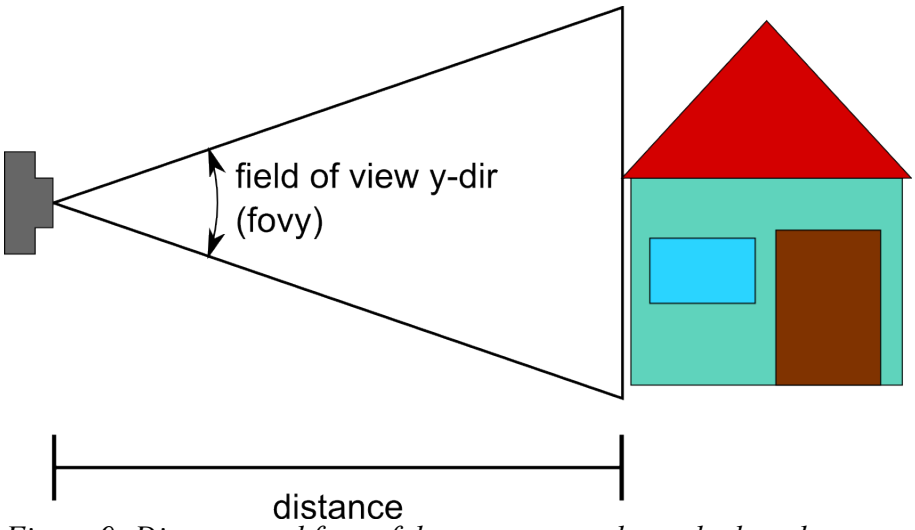

Figure 9: Distance and fovy of the camera, used to calculate the texture sizes.

Additionally there are two buttons (**Get Distance** and **Set Distance**) that will move the camera of the scene to a certain distance or calculate the scene cameras distance and store it in the control.

Texture Coordinates

Since each OBB has six sides and there might be hundreds or thousands of OBBs the program will merge the OBB textures into larger textures (LoD textures). Therefore it calculates texture coordinates for the sides of all OBBs.

It is possible to specify a minimum and a maximum size for the LoD textures (**Min. Texture Size**, **Max. Texture Size**). Cga-LoD will start with small LoD textures and try to reduce the number of textures used automatically by increasing the LoD textures size.

Rendering

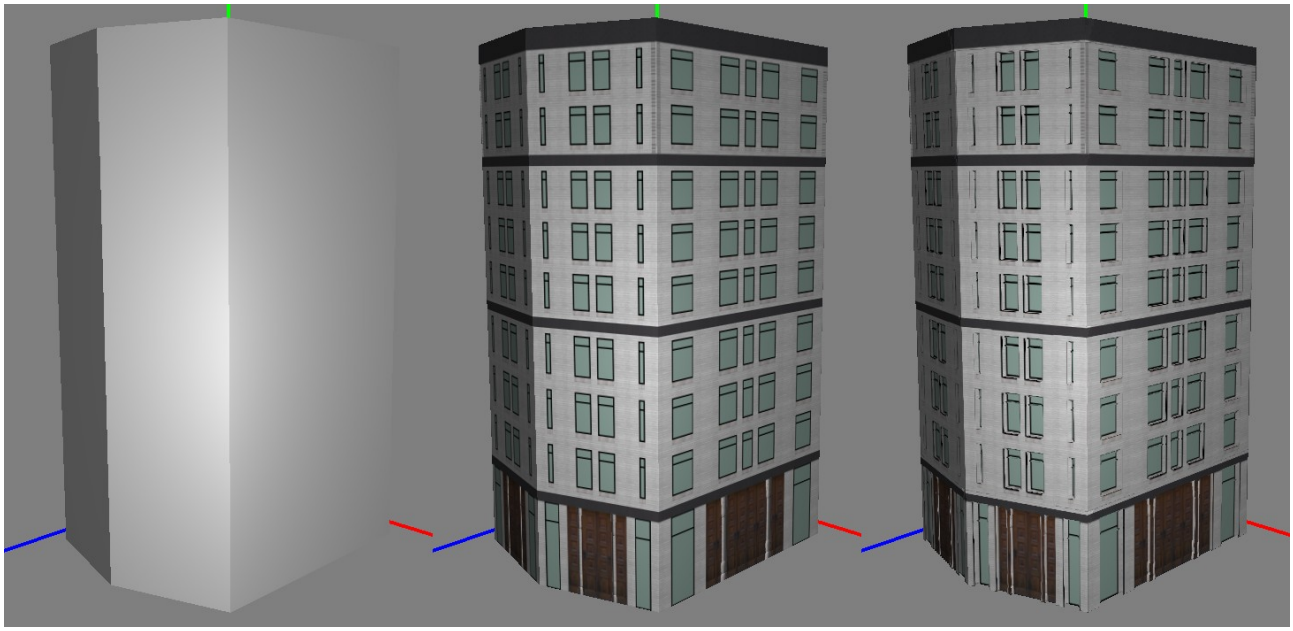

Figure 10: A LoD model, rendered with the three different render modes (untextured, normal mapping and parallax mapping)

Cga-LoD offers several **Render Modes**. The LoD mesh can be rendered without textures, which is useful to see how the actual geometry looks like. It is also possible to render the LoD mesh with Normal Mapping or with Parallax Mapping.

Using the 'Untextured' render mode, the program won't create LoD textures, and therefore it will be faster (which is very useful when LoDs on a low level). If an LoD model is exported, the program will switch back to Normal Mapping and create the textures.

Parallax Scale and **Parallax Bias** are two additional parameters when using Parallax Mapping.

Update

If a new LoD mesh shall be build with the current parameters, it is necessary to call the **Update** method. It is is also possible to enable the **Auto-Update**, then Update will be automatically called whenever a parameter changes. This is useful when testing different parameters without manually calling Update.

Textures of the Level-of-Detail Model

Figure 11: The controls of the "LoD Textures" block allow to take a look at the textures of the LoD model.

With **Show LoD Texture** it is possible to display the created textures in the window. The textures will be always displayed as a pair (color and normal texture). With the **Scaling** the textures can be

displayed larger or smaller.

Just one texture pair will be visible at a time, but with **LoD Texture** is is possible to switch between the pairs (there are **# LoD Textures** pairs). **Texture Size** tells the resolution of the LoD textures (all are square and have the same size).

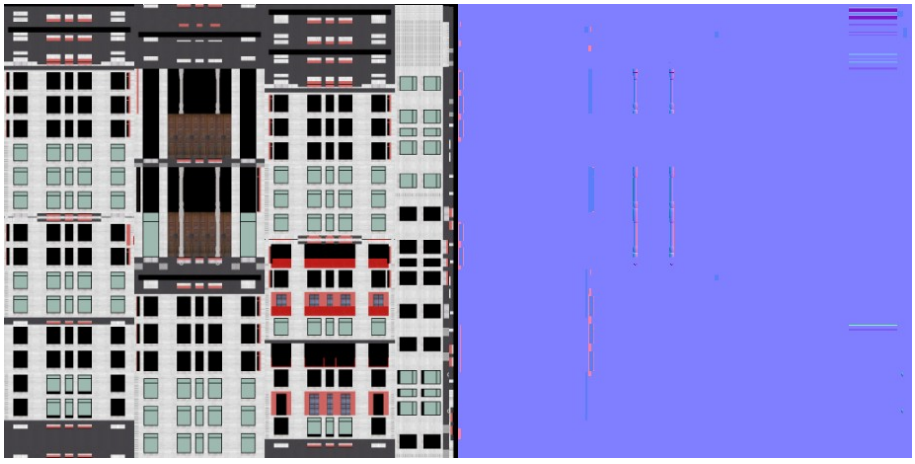

Figure 12: A LoD texture pair (color texture + normal map).

Information about the Level-of-Detail Model

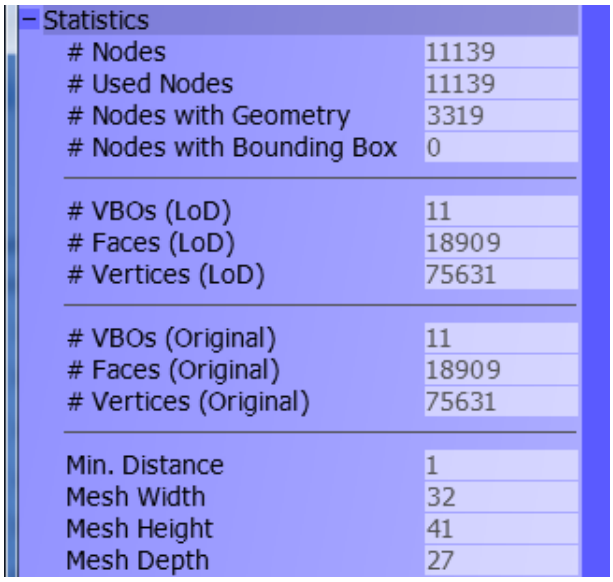

Figure 13: The controls of the "Statistics" block give additional information about the original and the LoD model.

In the Statistics Section we can find additional information that might be useful. In the first block of statistics we find information about the hierarchy and how many nodes are used. **# Nodes** is the total number of nodes in the hierarchy while **# Nodes used** is the number of nodes (with original geometry or OBB) that are used to create the current LoD mesh. **# Nodes with Geometry** and **# Nodes with Bounding Box** tell us of how much of the used nodes we use the original geometry and of how much we use the OBB (note that the sum of this two values doesn't have to be the number of used nodes, this is since not all used nodes must have a geometry or a OBB).

The second block of statistics tells us about the size of the LoD Mesh. **# VBOs** is the number of different Vertexbuffers (we count the VBOs for vertices, normals, texture coords, etc just as one

Vertexbuffer) used for Rendering. **# Faces** and **# Vertices** is the total number of faces (can be triangles or quads in the original geometry, just quads for the OBBs) and vertices in the VBOs.

The third block gives some information about spacial dimensions. **Mesh Width**, **Mesh Height** and **Mesh Depth** give the size of the axis-aligned bounding box of the original mesh. This values are in the same scale as the Distance value, so they can be used as a reference. The **Min. Distance** is the calculated minimum value for the Distance. Distances below this value would require larger textures.

Additional Options

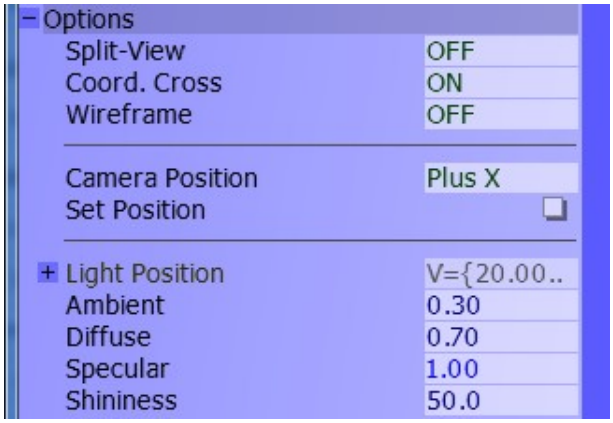

Figure 14: The controls of the "Options" block

Split-View

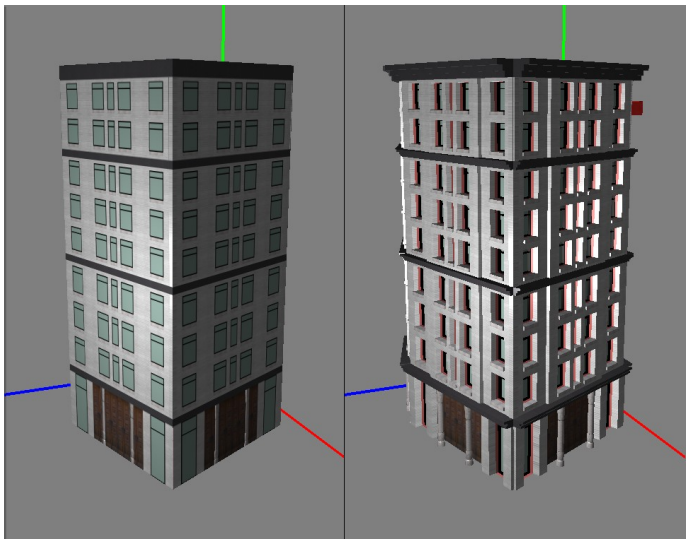

Figure 15: Split-View activated. On the left half we can see the LoD model, while on the right half we see the original model.

Cga-LoD offers a so called Split-View. With this option enabled the window will be split into two sides. With the Split-View we can easily compare the LoD model (left side) with the original model (right side).

Coord. Cross

Coord. Cross allows to enable or disable the coordinate cross and Wireframe enables or disables wireframe rendering.

Camera Position and Set Position

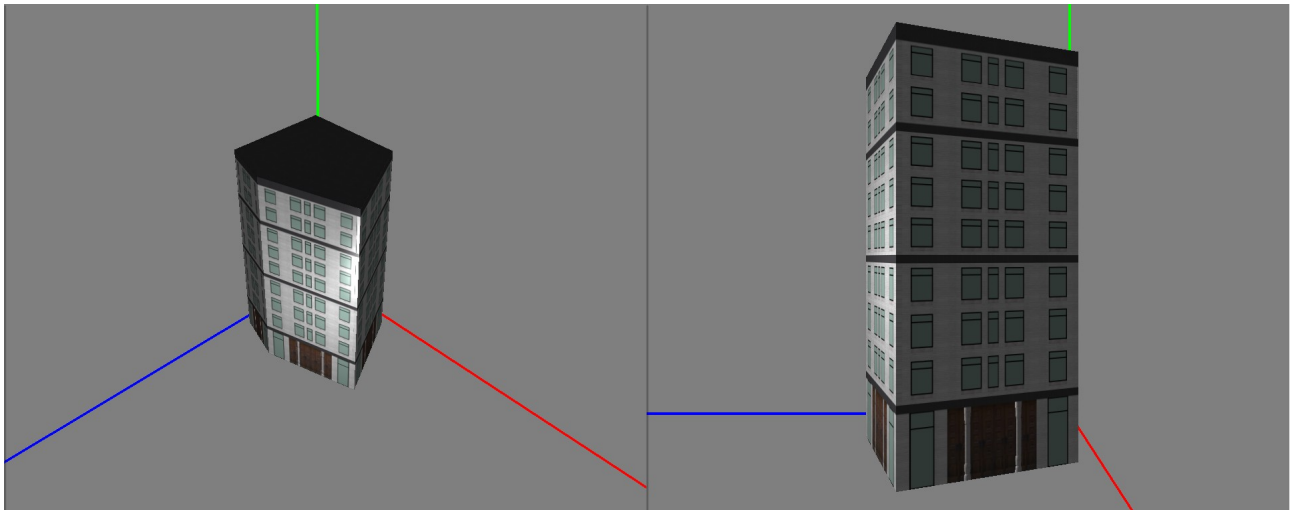

Figure 16: The scene from two different camera positions (left: +X+Y+Z, right: +X)

With Camera Position and Set Position it is possible to move the camera fast to one of 26 specified positions. All this positions lie on the plus/minus X and/or Y and/or Z side of the mesh.

Camera Positions selects one of the positions, and Set Positions moves the camera to this position.

Light-Options

Light Position, Ambient, Diffuse, Specular and Shininess can be used to change the position of the light source or the global material properties.