Visibility Area

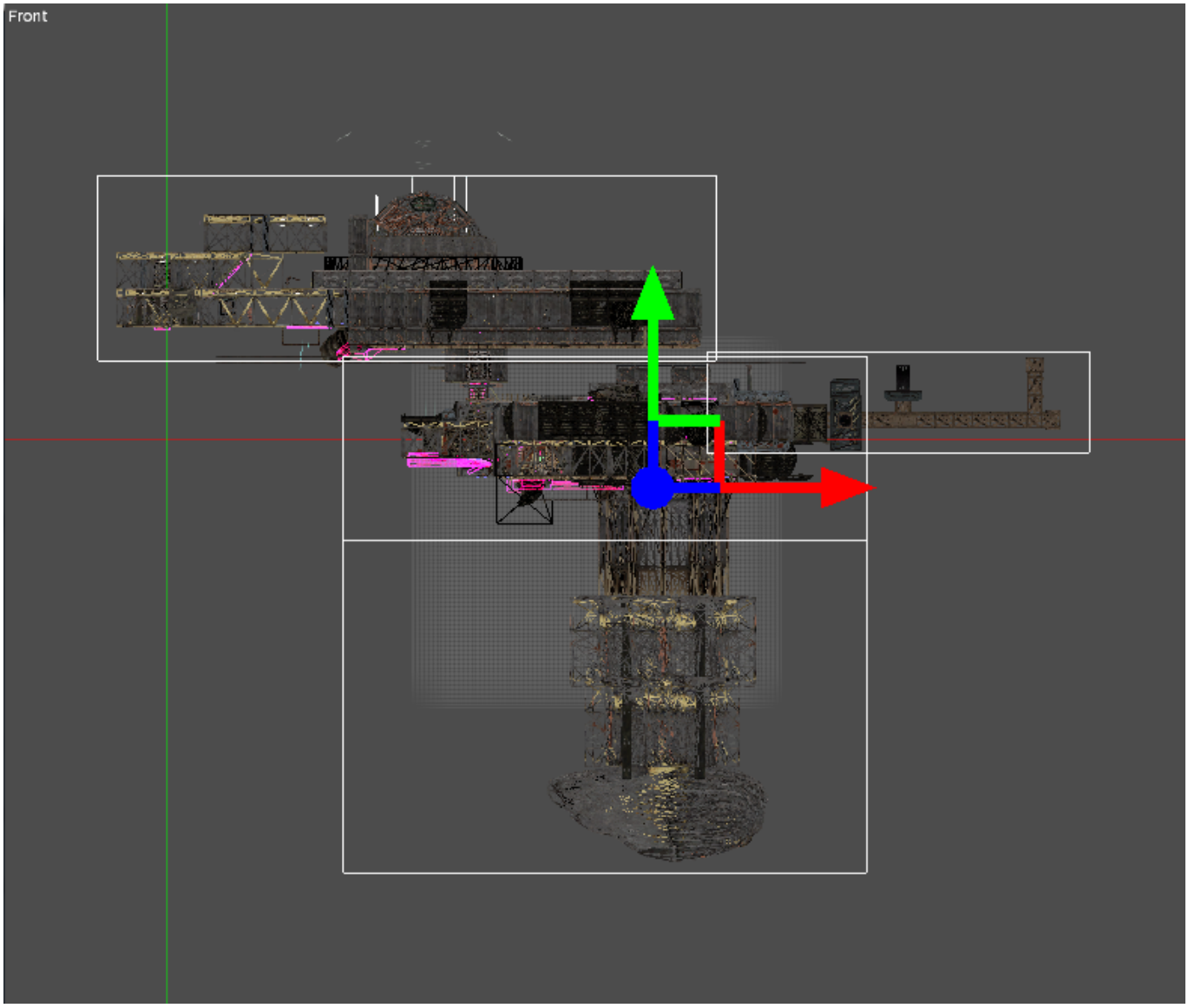
Overview

The visibility area is used to divide a map into areas that should not be visible at the same time. This helps the engine cull objects that are outside of the view and should not be visible. Adding vibility areas to the map will increase performance since it will give the engine an easier time partitioning the objects. It is also possible to manually disable a visiblity area, this will hide all the objects that are inside it. An object will be part of all visibility areas that it is fully inside of.

Objects that doesn't fit inside any of the placed areas will get added to a main render container that has an infinite size.

Placement

Visibility areas should be placed around large rooms, floors or large outdoor zones. They should be placed so that when you are inside one of the areas the other areas are not visible because it is occluded by geometry (like floors or walls). If a map has multiple floors with stairways between them then it would be a good idea to place one area per floor. A visibility area should be added to rooms that have a lot of objects. If the room is big but has few objects then placing an area is not needed. Most visibility areas should be large, covering a whole floor, but some can be small if the room has a lot of expensive objects. Only objects that are fully inside an area will be added to it.



Here is an example of how I divided 1.2 into visibility areas. Four areas are used.

Settings

Add Partial Intersecting Objects

If objects that are only partially intersecting the area should be added to it. The default behaviour is that only objects that are fully inside the area will be added. Enabling this can lead to false positives and the objects being culled when they should be visible.

Min Objects Per Node

Small nodes will merge into larger ones if the number of objects per node is less than this value. This is based on the average and not enforced per node.

Max Objects Per Node

Large nodes will be split into smaller ones if the number of objects per node is larger than this value.

Scripting

Visibility_SetAreaActive(const tString &in asName, bool abActive)

Disable or enable an area and make all objects inside invisible.

Visibility_SetMainActive(bool abActive)

Disable or enable the main area and make all objects that are not part of a visibility area invisible. This can cause issues with large lights being disabled. Add Partial Intersecting Objects can be used with this to disable world the outside while inside a room.

Visibility_SetTerrainActive(bool abActive)

Hides terrain.

Notes

Visibility areas can be rotated to get a better fit. Rotating an area makes updating of dynamic objects a little more expensive.

Play around with min/max objects per node to get the best result (F5 is supported). The goal is to reduce the number of draw calls and queries with draw calls being the main stat.

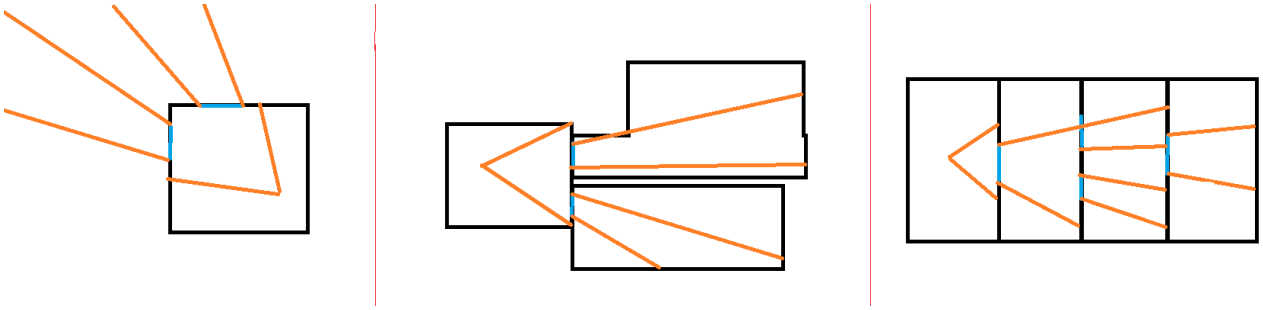
Placing visibility areas will never make objects on the map disapear when they should be in view.

Visibility Portal

Overview

A visibility portal is an area used to link together two or more visibility areas. A visibility area that is connected to a portal will only be visible if the portal is visible or if the camera is inside it. If the camera is inside of a visibility area then all the portals connected to it will be used as a portal to the outside.

A visibility portal will also clip the view frustum. This means that only objects that can be seen through the portal will be visible. Frustum clipping can also be used to disable other portals so that areas can be skipped directly.



The left image shows how a portal is used to cull the outside world. The middle image show one visibility area connected to two others with the use of portals, only the objects inside of the clipped frustum will get rendered. The last image shows that portals can be used to clip other portals. The current setup is limited so that only the first portal is used to cull other portals. If you look at the right most portal you can see that the whole portal is being used instead of clipping it with the portal before it, this might get fixed later.

Portals can also be combined with occlusion query based culling. This means that the GPU can check if a portal is blocked behind other rendered objects, and if that happens then the portal gets closed.

Portals should always be intersecting with all the visibility areas that it belongs too. If it is placed outside then the whole visibility area might get culled when it should be visible.

Settings

Connected Areas

List all the areas that are connected to this portal. Seperate using ;

Notes

A good place to place portals are windows and doors. The portal should always be a little bigger than the window or door that it covers. If it is smaller then there might be objects that pop into view.

Portals can be used on outdoor levels as well. If there is no roof on the level then the portal should be scaled to a height that covers the whole wall.

Portals also work when retreiving shadow casters.

Be vary of how many portals and areas that you use on a map. Since one query is used per visible portal you might end up having more draw calls than you would without the portal.

Portals can be disabled in the debug menu.

Unlike visiblity areas, adding portals incorrectly will make objects pop in and out when they should be visible.