Map

Overview

General

This is the basic structure for a level script. It should control all level specific code.

Callback Methods

These are the methods that are called from the Map:

void OnStart()
Called the first time the map is loaded.

void OnStart()
Called the first time the map is loaded.
void OnEnter()
Called every time the map is loaded
void OnLeave()
Called when player leaves the map
void CreateData()
All special data needed for the map (and not saved) are created here.
void DestroyData()
All special data needed for the map (and not saved) are destroyed here.
void Update(float afTimeStep)
Called every update tick
void PostUpdate(float afTimeStep)
Called after all normal Update during a tick has been called
void VariableUpdate(float afDeltaTime)
Called right before rendering the map. Has a variable timestep. Should be used for non-gameplay code
void OnGui(float afTimeStep)
When ImGui_* functions should be called.
void OnAction (int alAction, bool abPressed)
When an action is made.
void OnAnalogInput(int alAnalogId, cVector3f &in avAmount)
When an analog action is made.
float DrawDebugOutput(cGuiSet @apSet,iFontData @apFont,float afStartY)
This only used for pure debug purposes. Use cLux_DrawDebugText(…) for easily outputting messages. “Show Map Info” in the F1 menu must be turned on for it to show up.
void OnRenderSolid(cRendererCallbackFunctions@ apFunctions)
Useful for drawing debug output or if you want some really specific effect in the level.

Map Creation Functions

Overview

When

Map Streaming

Overview

Map streaming is a way to reduce the loading time when changing map. It works by preloading assets, static and dynamic entities. Map streaming can also be used to make seamless transitions between two maps by retaining the position and state of the player and objects near the player.

Map Streaming only works if the map_cache of the map being loaded is up-to-date.

It is not possible to save the game when a map change is occuring. Deloading or changing a map counts as map change. It is still possible to save a map while another map is being streamed. The preloading will start again after loading the save

Preloading

Preloading data is the main function of map streaming. By loading all the textures, meshes and materials in the background it is possible to reduce the time it takes to change map. The preloading is limited to 1 gb of textures (less at lower quality settings). If a map has 700 mb of texture data and the map being preloaded has 700 mb unique textures then the last 400 mb of textures will only be loaded into the CPU ram and will not get sent to the GPU until there is enough space.

Preloading a big map takes around 1-2 minutes. For computers with slower hdds this can be doubled. So make sure that the map starts preloading a minute or so before the player reaches the end. Dont preload the next map at the start of a map, since preloading can cause small lag spikes.

void Map_Preload(const tString &in asMapName)

Deloading

To make room for more data in the memory and to reduce the time it takes to switch maps it is possible to deload a map. Deloading will delete all entities in the map and all assets that are not used in the level that is being preloaded.

It is possible to limit the deloading to exclude an area. Any object that intersects with that area will

not get deleted. This can be used to make smooth transitions between levels.

Deloading will be done automatically when ChangeMap is called. But it is possible to call it manually to reduce wait time if the memory is starting to get full

void Map_Deload(const tString &in asMapTransferArea)

Map Transfer Area

In order to make smooth transitions between levels you have to use an Map Transfer Area. This is an area that is placed around the end of map A and the begining of map B. The player and any objects inside this area will retain their state and position relative to the area. The area needs to be placed at the same relative position and have the same size in both levels. It is possible to have different rotation of the area, in cases where maps dont share the same orientation. The area also needs to have the same name on both maps. It is possible to use multiple Map Transfer Areas for maps multiple endings.

The state of the objects are transfered using a save state, this contains everything from position to the value of a script variable. For each object inside the area of map A the state is saved. The code then looks for a entity with the same name in map B. If the entity exists then the state is loaded, otherwise the entity is recreated on map B with a _fckg_QUOTstreamfckg_QUOT_ suffix at the end. It is important that the area used during the transition looks the same on both maps. Otherwise the player will notice the change. This only works for Props that allow map transfer, this is true as default. Use SetAllowMapTransfer(false) to disable map transfer of a prop type.

The name of the Map Transfer Area needs to be passed as an argument to ChangeMap in order for it to work.

Deloading either happens automatically when changing a map, but it can also be done manually by calling Map_Deload(). This should be done once the player is inside the Map Transfer Area and can no longer leave it.

void Map_ChangeMap(const tString &in asMapName, const tString &in asStartPos, *const tString &in asTransferArea*,
const tString &in asStartSound, const tString &in asEndSound)

void Map_Deload(const tString &in asTransferArea = “”)

Changing Map

Changing of map can occur after the level has been preloaded. If the level is not fully preloaded there will be a wait time. The player can continue playing while this is happening, so make sure to have a larger map transfer area where the player can move around.

The change itself takes up to 300 ms on larger levels and will lead to a short freeze.

A loading screen will be shown if the map has not been preloaded or if no Map Transfer area is used.

Map_IsPreloadCompleted() can be used to check if the preload is done and the change can happen.