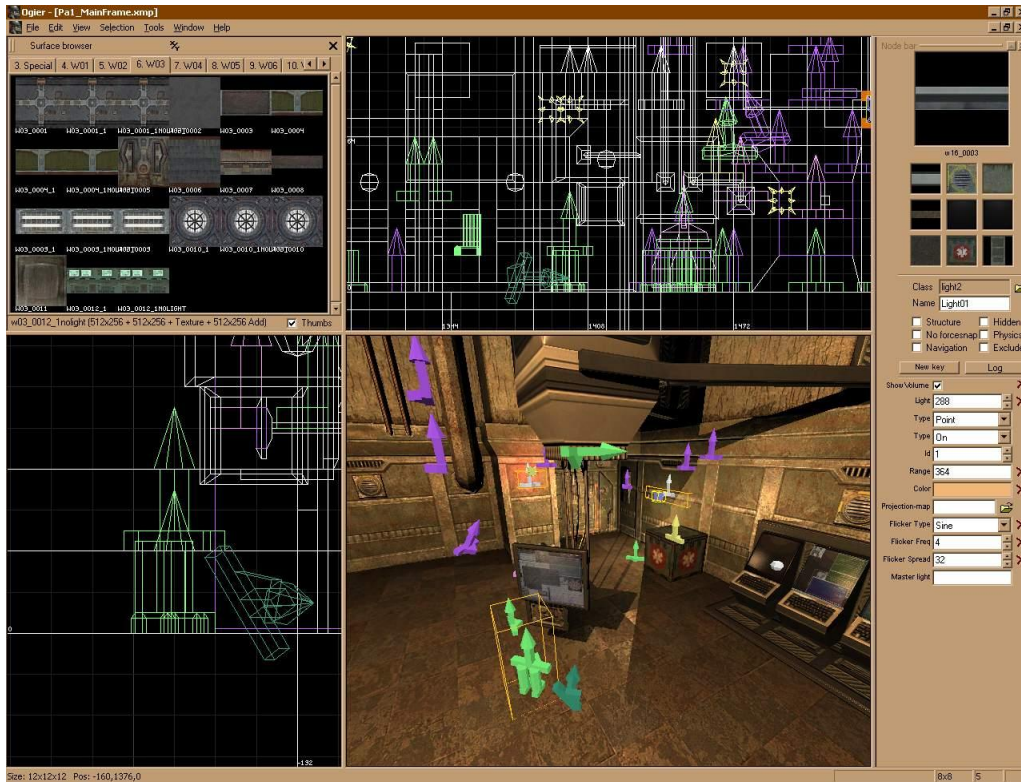


# Ogier

## Starbreeze Level-editor and Multi-tool



### Introduction

Ogier is an in-house developed game-editor, tightly built around the Starbreeze Engine. It is used for many tasks and is easily extendable to include new features as they are implemented in the engine. Ogier has been in development within Starbreeze since September 1998. It was released to the public in January 2004, primarily to allow the community to create new levels for Enclave. Ogier has support to edit content for Enclave, Knights of the Temple and The Chronicles of Riddick: Escape from Butcher Bay. It also has some limited level-editing support for Quake.

The following tasks are done in Ogier

- Level-design – Create and texture architecture using convex solids and splines.
- Lighting – Light a level using static and dynamic lighting.
- Scripting – Place NPCs, control the AI, design events and create special effects.
- Texture compilation – Bundle and compress textures into texture-sets
- Surface editing – Create and edit materials for both the world and special effects.
- Model conversion – Import models from Maya and 3DS for use in the Engine
- Animation conversion – Import animations from Maya for use in the Engine
- Dialogues editing – Edit dialogue animations, subtitles and scripting

## About the manual

This is the first drafts for a complete Ogier manual. It may not cover everything yet, but should provide a good start for creating content for games supported by Ogier.

If you have suggestions or questions that aren't answered in this manual or want to know more about Ogier please visit the official forum at:

<http://forum.starbreeze.com/>

Most of the work on this manual has been done by Johan Oskarsson, who also handles the updates and revisions. Jens Andersson has helped out with corrections. The Enclave AI scripting part was written by Anders Olsson.

*I'll start with a short explanation of the main parts of this program and how to operate in Ogier before moving on to the more detailed information. Many of you will most likely have some kind of experience and knowledge about other level design programs, but I've tried to make this manual as detailed as I possible can so that first time users will get as much information as they can get to get started. I hope that it will help you to create your own maps and game-play. Keep in mind that Ogier keep changing all the time and that this manual is a work in progress. We will try and update this document as we go but there are no guarantees that it will be 100% right.*

*Huge thanks to Jens Andersson, Jan Andersson, Anders Backman and Fredrik Ljungdahl who has endured a bucketful of questions during the creation of this documentation and probably haven't seen the end of it yet. Huge thanks to the people at the forum as well who keeps the community alive and breathing.*

*Johan*

## Table of contents

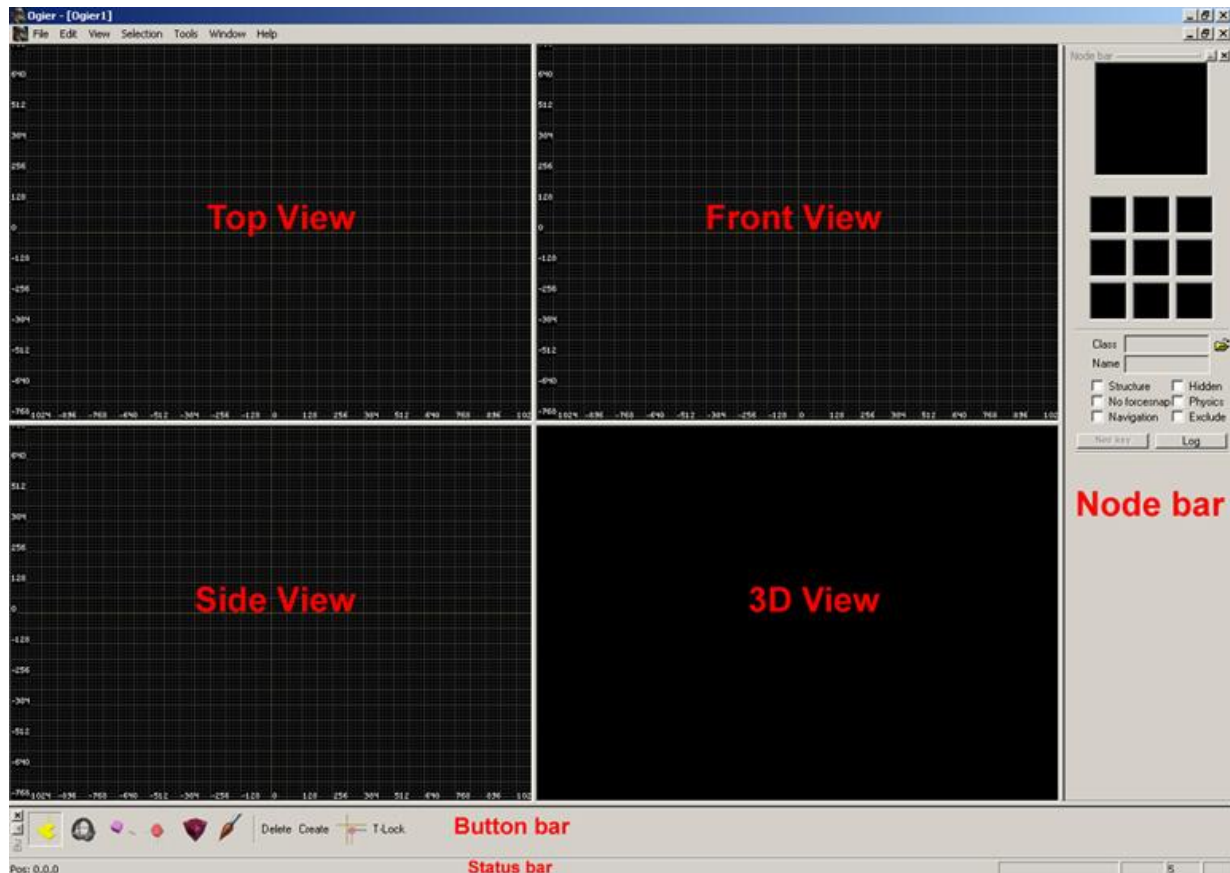
Ogier .....	1
Introduction.....	1
About the manual.....	2
Table of contents.....	3
UI layouts.....	5
Navigation / View .....	6
Selection.....	9
Workspace.....	10
Level-design.....	13
Tools .....	13
Generating brushes.....	20
Surface browser .....	22
Nodebar.....	24
Misc. info and settings .....	26
Structure.....	28
The Create object menu .....	29
Lighting.....	32
Lighting with Lightmaps (Enclave, KotT).....	32
Lighting with Unified Lighting (Riddick) .....	32
View XW .....	32
Light options .....	33
Scripting.....	35
Triggers .....	35
Engine/Hooks.....	39
Dynamics.....	43
Characters .....	45
Teams.....	49
Scenepoints .....	50
Impulses/AIImpulse.....	53
Particles.....	60
Using the AI (Old Enclave document).....	67
Using the Templar Camera system .....	89
Texture compilation .....	94
Surface editor .....	95
Model conversion.....	97
Animation conversion .....	98
Dialogue.....	99
Dialogue settings.....	99
Dialogue editor.....	102
Ogier Shortcuts/Hotkeys.....	103
Templates and Registries .....	107
Compile options .....	109
Lightmap and Light grid options .....	109
XW Process.....	110

Tutorials .....	111
Ogier Tutorial: Creating Your First Map (Enclave). ....	111
Ogier Tutorial: Ledges and Paths (Chronicles of Riddick) .....	144
Ogier Tutorial: Building an elevator (Chronicles of Riddick).....	156
Ogier Tutorial: A simple sky (Chronicles of Riddick) .....	169
Ogier Tutorial: Multi-Part Door (Chronicles of Riddick) .....	174
Ogier Tutorial: Converting Maps from Worldcraft/Hammer to Ogier .....	183

## Map-editor

Ogier has three major tasks in the creation of level. The first is Level-design where the architecture is built. The second is lighting, where lights are placed and the final one is scripting, where the actual game-play is created. These three tasks are all done in the Map-editor.

## UI layouts



(a picture says more then 1000 words?)

The main parts of the UI are the four different views, the node bar and the status bar. The Top, Front and Side view will be displayed in 2D while the 3D view will give you a preview of how the map looks like in-game. The Node-bar lets you select different characteristics of brushes, objects and models.

The Button bar has shortcuts to many of the tools you will be using. You will also be able to select the main tools with "Space" on your keyboard, so many tend to remove the Button bar once they get used to the shortcuts to free up more space for the four display views. The status bar displays the position and size for your objects among other things. We will take a closer look at all these things later on.

## Moving around in the editor

One of the basic ideas for the editor was to develop an editor where efficiency was a bit more important than user friendliness. This means that it can be a bit tricky to get used to. The most important fast-keys are places around the left side of the keyboard; more specifically the keys W, A, S and D. Most of the time, you will never have to move your hand away from there while working in Ogier.

While holding Shift the editor enters the "move-around" mode. In the 2D views W and S serves as zoom in/out when in "move-around" mode. In 3D the controls are just like any first person shooter with mouse-look.

## Selecting objects

You select objects by clicking on them with the left mouse-button, either in the 2D-views or the 3D view. A selected object will be displayed as blue. Holding down Ctrl will let you add additional objects to the selection. You also use the right mouse-button to create a selection frame, which will select all objects inside its area. Holding Ctrl while using the selection frame will add the objects to current selection and holding Alt will remove objects from the selection.

## Creating objects

You mainly create objects in Ogier by fast keys. There are two main types of objects that you will use in the creation of a map, those are called the primitives and splines, we call these building objects brushes. The objects you create will automatically be selected. There is also the option to name objects when you create them or during your work so that you can find them easy as the map gets bigger. More detailed information about how to create objects and the options around those will be described later on.

## Grouping, Layers and Hiding

You can group objects together to make the selection and moving a lot smoother. This will bind the objects together so that they work as a group; a hierarchy will be created when you group objects or other groups together. There is also the option to use layers that will help you keep track on objects and groups in Ogier. You can hide certain parts and objects in your map to clear the view from objects and parts of the map you're not working on at the moment.

## Navigation / View

As explained earlier in this document we tried to make the settings as efficient as possible, relying on shortcuts. This might mean that at a start Ogier might be a little confusing in the beginning but hopefully you will get comfortable with the settings quickly. The navigation in Ogier is focused around the idea that you will never remove your left hand from the left part of the keyboard. Ogier tries to mimic the keyboard settings a gamer would use for a game like Quake, Half-Life or Riddick. Keyboard navigation with the left hand and use the mouse to look around and select objects with. You can use **Shift** or the **Middle Mouse Button** to enable mouse look in the 3D view this will let you navigate inside your map like you would in-game. This part will only go though the navigation keys and shortcuts of Ogier, once you have read though most or all

of this documentation you will see that most of the shortcuts are in reach for you left hand.

This is how we have planned the keys around the left hand side.

Little finger: **Left Shift / Left Ctrl**

Ring finger: **A**

Long finger: **S / W**

Index finger: **D**

Thumb: **Space**

**A, S, W, D** are the main navigation keys, shift will enable mouse look in the 3D view and scroll mode in the 2D views. (You can also use the arrow keys to navigate in the 2D and 3D views. But we recommend that you try to navigate with A, S, W and D.)

To switch between multiple levels in full screen you can use **Ctrl+Tab**.

Here are some of the shortcuts and key settings for the 2D and 3D view navigation:

## 2D view

Holding down **Shift** or the **Middle Mouse Button** in the 2D views will let you scroll through your map with the mouse and you can use different ways to zoom.

**Shift+W**: Zoom in

**Shift+S**: Zoom out

**Shift**: enable mouse scrolling

**+**: Zoom in

**-**: Zoom out

**Mouse Wheel Up**: Zoom in

**Mouse Wheel Down**: Zoom out

If you lose an object in any of the views while modifying a object in one view and then move to an other etc you can center the views towards the selected object.

**C**: Center on selection.

**Alt+C**: Center all views on selection.

## 3D view / navigation

Think of this as if you were in-game and playing through the level you just made in no clip mode (lets you walk through walls). All you have to do is holding down **Shift** or the **Middle Mouse Button** to enable mouse-look in the 3D view.

**W**: Move forward

**S:** Move backward  
**A:** Strafe left.  
**D:** Strafe right  
**Shift/Middle Mouse Button:** Enable mouse look.

Holding down **Ctrl** and the **Navigation Keys** enables zoom and rotation around the selection.

**Ctrl+A:** Rotate around the selection.  
**Ctrl+D:** Rotate around the selection.  
**Ctrl+W:** Move towards the selection.  
**Ctrl+S:** Move away from the selection.

There are some alternative ways to move around in the 3D view.  
Use the ones you are most comfortable with.

**Mouse Wheel Up:** Step forward  
**Mouse Wheel Down:** Step backward  
**Shift+Right Mouse Button:** Move up  
**Shift+Mouse Wheel Up:** Move up  
**Shift+Mouse Wheel Down:** Move down

To center the views on a selection you can:  
**C:** Center on selection.  
**Alt+C:** Center all views on selection.

### Clipping distance

You can set the clipping distance in the 3D view. This is useful for extremely polygon intensive scenes that lower your frame rate.

**Ctrl+Page Up:** Increase clipping distance.  
**Ctrl+Page Down:** Decrease clipping distance.

### Navigation speed

To make it easier to move around in the 3D once a map gets bigger or you need to take a close look on some smaller details that you want to create you might need to increase or decrease the speed that you move around in the 3D view.

**Page Up:** Increase traveling speed in 3D view.  
**Press Page Down:** Decrease traveling speed in 3D view.

**WARNING!** If you hold down **PageUp** you may set the speed dial to such an extreme number that you instantly fly so far from the map that you can't find your way back. If this happens select any brush in one of the 2D views and press and hold **Ctrl+W**. This will bring the camera back to the selected object, but it may take a while.



## Selection

### Selecting objects

You select objects with the **Left Mouse Button** in the 2D-views or the 3D view, the whole or most of object has to be visible in the 2D views for you to be able to select it. A selected object will be displayed as blue. By holding **Ctrl** you can add additional objects to the selection. You also use the **Right Mouse Button** to create a selection rectangle, which will select all objects inside its area. Holding **Ctrl** while using the selection rectangle will add the objects to current selection and holding **Alt** will remove objects from the selection.

**Left click** on an object to select it.

Holding down **Alt** while **Left Clicking** enables “click-through”; this allows you to select objects behind the top-most object.

Hold down **Right Mouse Button** to draw a selection frame.

**Ctrl+Left Mouse Button:** Add/remove an object from a selection

**Ctrl+Right Mouse Button:** Add objects to your selection with the selection frame

**Alt+Right Mouse Button:** Remove objects with a selection frame

**F3:** Select all.

**Right Click** anywhere to deselect.

### Selecting vertex/edge handles

You select handles with the left mouse-button in the 2D-views or the 3D view. A selected handle will be displayed as yellow. By holding **Ctrl** you can add additional objects to the selection. You also use the **Right Mouse Button** to create a selection rectangle, which will select all objects inside its area. Holding **Ctrl** while using the selection rectangle will add the handles to current selection and holding **Alt** will remove handles from the selection.

**Left Click** on a handle to select it.

Hold down **Right Mouse Button** to draw a selection frame that can be used to select or deselect handles.

**Ctrl+Left Mouse Button:** Add/remove a handle from a selection

**Right Click** anywhere to deselect.

### Moving objects

You can use any of the views to move objects. But we recommend that you use the keyboard. The 3D view will only let you move the objects and handles with the left mouse button. To force a move in a single axis hold down Ctrl.

Select the objects you wish to move, use **A,D,W,S** to move them in the view your mouse pointer is in. You can also use the mouse to move the selected objects. To move an object with your mouse just click and hold down the **Left Mouse Button** anywhere on the object except on its handles (if you do you will only drag the handle, if you have handles

selected they will stay selected during the move). The same thing goes if you have more than one object selected; you will move all of the selected objects.

**M:** Opens up the move to dialog then the edge, vertex, texture tool is selected.

## Workspace

### TEMP

Workspace lets you search for objects, groups and layers and help you to keep track of all your work. You can move objects into groups and objects and groups into layers as you please. You can list your work here and get an overview of the work. You can also hide and unhide objects, groups and layers. This is a very useful tool to have once you get used to it.

Tree and List View will let you view all your objects in the map.

Link and Error View keeps track of most of your game play like triggers.

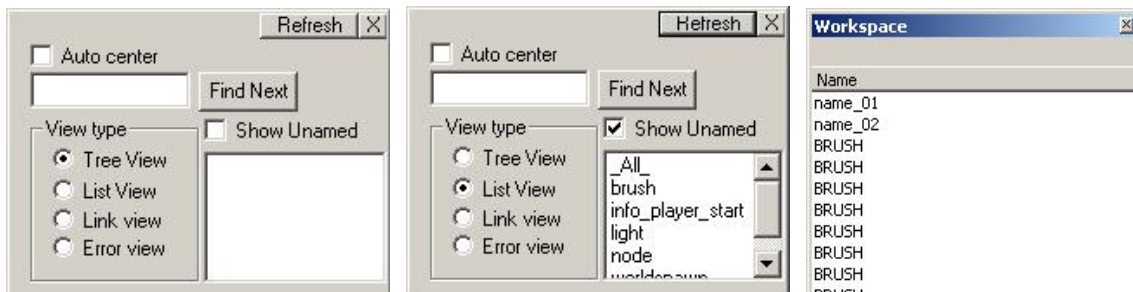
To bring up Workspace you can either press **Ctrl+F11** or go through the View menu.

Naming objects and Search for object, group and unnamed objects

You have the ability to name objects, groups and layers in Ogier.

To name a selected object or group you have to bring up the Node bar (**E** or **Ctrl+F10**). Here you can find the objects class and under that the field to enter its Name. Once you have given an object a name this field will become red. I think you will find this very useful when you need to find objects or triggers as the map gets larger. How to name the objects, groups and layers are entirely up to you, but this is a very useful thing then more than one person working on the map, just an example but if one does the building and the other does the game play for the map.

To search for an object you can either search for its name or search for it by its type and go through all objects of its class. Bring up Workspace (**Ctrl+F11** or through the View menu). To search for a named object just type its name in the field in front of the find next button and press enter or click the **Find Next Button**.



To find an unnamed object, chose List View and select the Show Unnamed option. You can select what classes to be shown when Ogier lists the objects you have created. Just

click on the object name in the list to select the object, you can use the arrow keys to move up and down the list.

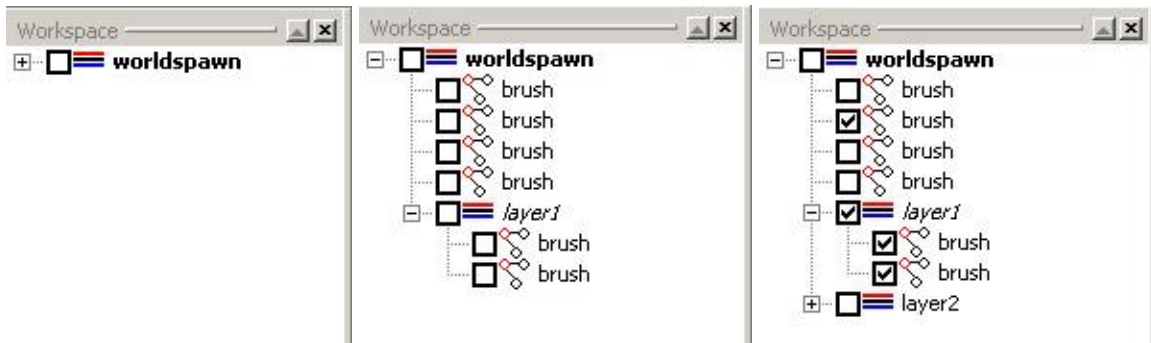
**E** or **Ctrl+F10**: Bring up the Node bar.

**Ctrl+F11**: Bring up Workspace.

## Layers

You can use layers in Ogier. It's another way to group objects that can be very useful and handy. To add a layer you have to enter the create object menu (> or **Insert**). Layers are located under classes; double click on layers to add a layer. The layer will appear as a tiny red sphere, and blue while selected. You can name the layer in the Nodebar, and view the layers in Workspace.

Press **Ctrl+F11** or select Workspace through the view menu.



If you bring up Workspace and select view type Tree View, you will see something like this. You open and close a branch by clicking on the small box on the left side of the window.

☐: Closed branch. Click on it to open.

☒: Open branch. Click on it to Close.

To hide objects, groups and layers through Workspace you click on the box on the left side of the name. To hide a whole group or layer just click the box in front of the group or layer and all objects inside will be hidden.

☐: Means that an object is visible.

☒: Means that an object is hidden.

To add objects to a group or layer just drag and drop the objects onto the different groups or layers. Be sure to drop the files in the right place though because you can change the class of an object if you're not careful, Ogier will automatically change a brush into a node if you drop a brush on a brush to create a new group. So please use **G** or go through the menu to group objects until this has been fixed.

If you right click on the object, group or layer name you will bring up a small menu.



**Set Parent:** This will set where new objects will be created. If you set a group or layer as Parent everything you create will be created in that group or layer. Worldspawn is set as Parent by default. NOTE: Ogier doesn't keep the parent settings when closed so you have to set it each time you open Ogier if you want an other parent then the default one.

**Flatten:** This will eliminate and flatten all sub groups below the one being flattened. Layers will stay but all objects and groups inside a layer will be flattened.

**Refresh:** Refreshes the Workspace window.

**WARNING:** If you drag and drop a brush on a brush you will change the brush into a node. Ogier lets you change class of objects; this is very useful in some situations but can be confusing in some if you're not aware of it. If you drag and drop a brush on a node the brush won't change its form, just the class, so it will look like a brush but is in fact something else. So please keep this in mind when working.

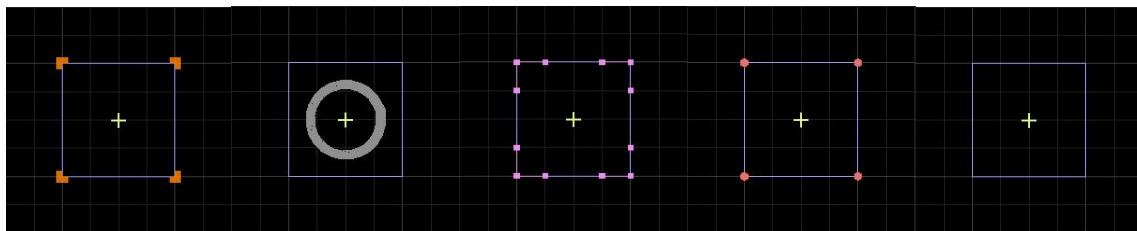
## Level-design

As many similar editors, Ogier is based on brush primitives. A brush is a convex object. When combining these brushes you can build complex worlds. Ogier was build specifically for the Starbreeze Engine, using *Stoneless* and *Worldcraft* for some inspiration and ideas.

## Tools

There are five main tools used in creating a map in Ogier all these can be cycled through by pressing **Space**. Use **Space** to cycle forward and **Shift+Space** to cycle backwards through the tools. You can also access these tools through the shortcuts **Alt+1** to **5**, the tools menu or the button bar. The Edge tool will only work and appear with spline brushes.

These are the different kinds of tools:



Scale tool

Rotate tool

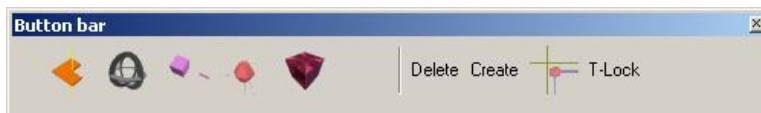
Edge tool

Vertex tool

Texture tool

### The Button bar

The button bar is an alternative way to select tools and access some features in Ogier. To enable or disable the button bar enter the View menu and select the button bar, you can also use the shortcut **Ctrl+F9** to enable/disable the bar.



(The Button bar in the newer versions of Ogier)



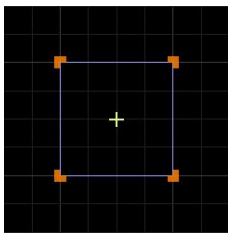
(The button bar in older versions of Ogier)

1. Scale tool
2. Rotate tool
3. Edge tool. (Only works with splines)
4. Vertex tool
5. Texture tool
6. (Paint landscape.) Removed

7. Delete
8. Create
9. Grid snap
10. Texture lock

The button bar isn't needed to create maps in Ogier it's just an alternative way to select tools. To detach the button bar from its position at the bottom of the screen double click over or under the symbols or left mouse button drag it from its position. To position it down below again just double left click on it or place it at the bottom manually.

## The Scale tool



To enable the scale tool you can toggle **Space** until it appears, through the tools menu or the button bar. Once you have the scale tool selected you will see the orange handles appear in each corner of the selected object.

To scale an object you can either use **A, D, W** and **S**, the **Mouse** or the **Arrow Keys**. Pressing **Ctrl+A**, **D**, **W** and **S** or **Ctrl+Arrow Keys** when an object is selected will modify its shape, no matter what tool you have selected in the selected view.

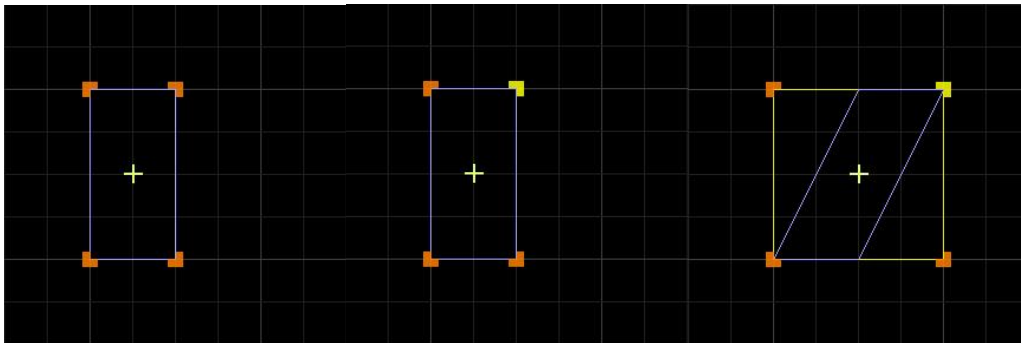
## Scale handle



You can left click and hold down on a scale handle to modify its shape.

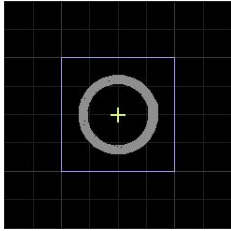
**M**: Open the scale dialog there you can scale an object in percent.

## Skew



If you right click and hold down a scale handle with the scale tool selected you will skew the brush in the selected plane.

## The Rotation tool



You can enable the rotation tool by cycling through the tools with “space” until it appears, through the tools menu or the button bar. Once you have the rotation tool selected you will see a ring appear in the center of the selection, this ring can be used to rotate objects in any of the views.

Just press and hold down the left mouse button on the ring and move the mouse left or right to rotate the selection. This will rotate the selection 15 degrees per step. If you hold down shift at the same time you will rotate the selection 1 degree per step instead of 15.

Pressing “V” will fast rotate the selection 45 degrees, this works no matter what tool you have selected but only in the 2D views.

Pressing Shift+”V” will Fast rotate the selection -45 degrees, this works no matter what tool you have selected but only in the 2D views.

“M” will open the rotation dialog there you can rotate an object in percent.

“X” will toggle if the rotation-wheel should lie in the center of the selection or not.

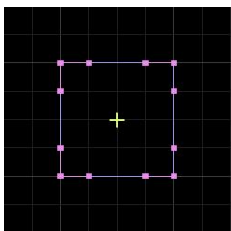
Pressing Alt+”X” will Flip the selection in X-Axis but only in the 2D views.

Pressing Alt+”Y” will flip the selection in Y-Axis but only in the 2D views.

Pressing Alt+”Z” will flip the selection in Z-Axis but only in the 2D views.

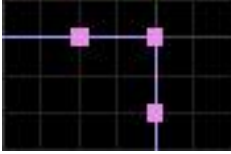
Right clicking the rotation ring will fast rotate the selection 45 degrees.

## The edge tool



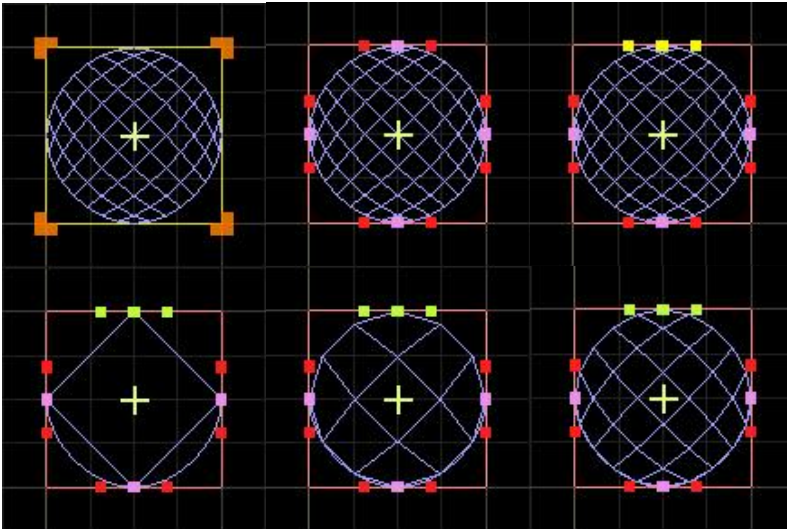
You can enable the edge tool by toggling “space” until it appears, through the tools menu or the button bar. Once you have the edge tool selected you will see the pink handles appear, these handles can be used to modify the objects in any of the four views.

### Edge handles



The edge handles can be manipulated in all the four views to change shape of the selected brush. To select the handles just left click on them or use the right mouse selection rectangle. Use Ctrl + left button or the selection rectangle to add/remove handles from the selection. A selected handle will change into yellow.

To change the detail of a spline you select the spline and go into edge mode. I am using a cylinder as an example here but this works with all types of splines.

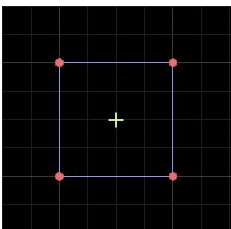


Selects the all or the handles you wish to modify and press “+” and “-” to change the detail.

“+” will increase the number of edge-points when using static tessellation

“-” will decrease the number of edge-points when using static tessellation

### The vertex tool





You can enable the vertex tool by toggling “space” until it appears, through the tools menu or use the button bar. Once you have the vertex tool selected you will see the red round handles appear, these handles can be used to modify the objects in any of the four views.

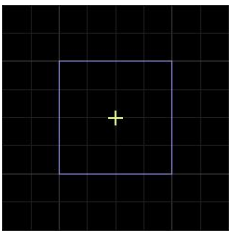
### **Vertex handle**



The vertex handles can be manipulated in all the four views to change shape of the selected brush, just like the edge and scale handles. To select the handles just left click on them or use the right mouse selection rectangle. Use Ctrl + left button or the selection rectangle to add/remove handles from the selection. A selected vertex handle will turn yellow. Sometimes it can be easier to select a certain handle in the 3D view.

A vertex handle will turn bright red if it's in a grid lower than 1, so don't be alarmed if your handles turn bright red on a brush and you're below grid 1. But if you're working at grid 1 or over this can be an indication that your brush is out of place.

### **The texturing tool**



There are no handles on the texturing tool, because you don't need any.

To enable the texture tool you can toggle “space” until it appears, through the tools menu or use the button bar. Once you have the texturing tool selected you will see no handles in the 2D views and the brush will change color in the 3D view. We will take a short look at how to select and paste a texture on a selected side of a brush in this part and then later on in the Texturing advanced there we will go in deeper on certain features and function.

To texture you need to select one or more brushes. Select the texturing tool and left click on all faces you wish to texture in the 3D view. Use Ctrl + Left mouse button on any face you wish to add/remove from texture selection.

To select all faces of the selected object/objects press "X".

You can also use the left mouse button to select multiple surfaces.

Select the object/objects. Press and hold down the left mouse button on one of the surfaces you would like to texture (you can't have any selected surfaces when doing this) then drag the mouse pointer over the other surfaces to select them.

To add a texture to an object you need to enter the surface browser.

To bring up the surface browser press "~" or "\$" depending on what keyboard setting your using. This will open the surface browser in any 2D/3D view that your mouse pointer is in (you can have different texture packs in different views). Select a desired texture from one of the menus, press "~" again to go back to your previous view. (More features and what you can find in the surface browser will be described later on) Press "K" to paste that texture onto your selected faces.

You should now have a texture on the selected sides of your brush/brushes.

There are three ways to select textures in Ogier. Either you can use the Surface browser, select a previously used one from the node bar, or copy a texture from an existing object. The surface browser is the main way and it's also here you select the textures to begin with.

"B" or "Enter" brings up the Texture tool to modify the texture settings. Change its offset and rotation. To flip a texture simply put "-" in front of the scale (for example -0.25).

You can also add a texture to a whole brush if you want to while using an other tool like the scale tool, just select a texture in the surface browser or from the node bar and then press "K" to paste the texture on all sides of the brush.

## **Texturing Advanced**

Once a face or multiple faces are selected and a texture is in place you can offset it by moving your mouse pointer over a 2D view and press "A", "D", "W" or "S", you can also hold down the left mouse button on the texture to scroll the texture up/down left/right. The arrow keys (over a 2D view) or changing the offset numerically in the Texture tool can also be used to offset a texture.

Holding down the right mouse button over a selected surface and moving the mouse left/right will rotate the texture, this will rotate the selection 5 degrees per step. If you hold down shift at the same time you will rotate the selection 1 degree per step instead of 5. More precise rotation can be done in the Texture tool.

Ctrl+Right mouse-button on a selected surface: Fast rotate 45 degrees

Keep in mind that every time you bring up the Surface browser the thumbs setting will change, so if you want to see the textures in thumb mode or not just check or uncheck the thumb box in the bottom right corner or close and reopen the surface browser to turn this on/off.

Pressing "I" will copy the selected texture.

Pressing "K" will paste the copied texture on a selected face.

Pressing "O" will copy the selected texture with its settings intact.

Pressing "L" will paste the copied texture with its settings intact on to a selected face.

Pressing "F6" will select surface faces.

You can also use the top of the node bar to select textures.



The Node bars functions will be described later on in this documentation.

### Texturing mode

Pressing "M" will cycle the texturing mode. These are:

For brushes: Box Mapping and Planar Mapping.

For splines: Box Mapping, Fixed ST-Mapping and ST-Mapping.

You'll see which mode is active in bottom left corner of screen.

Different mapping types have different colors.

**Texture lock:** Ctrl-T. To make sure a texture is kept locked during a rotation you have to change mode to plane mapping before the move/rotation.

**Planar Mapping** will retain the projection direction of the texture when altering the face to shapes that would otherwise cause the face to have the texture projected from another direction. Try playing around with the vertexes to learn more.

**Fixed ST-Mapping** will emulate how Box Mapping works, but follow the shape of the spline.

**ST-Mapping** will allow you to decide how many pixels from the texture that will be drawn on the spline surface. For example, if you map a 512\*512 texture onto a face you can enter 512 in the scale boxes in the texture settings and the face will be completely covered by the texture regardless of the shape of the face.

**NOTE:** Flipping or rotating a texture on a spline may involve unorthodox procedures. For example, instead of writing -0.25 in X scale you may have to write -0.25 in Y scale

and rotate the texture 180 degrees. Play around.

You can select all faces bearing a particular texture in any selection by selecting a texture ("I", or from the browser), then select any amount of brushes (like the whole map or a group or a single brush) and press "Select Surface Faces" in the "selection" window.

## Generating brushes

You mainly create objects in Ogier by using fast keys. Try them out, go through all these brushes and toy around with them before you start working on your map so get a feel of the different kind of shapes they have. You can also create objects by opening the Create Object dialog and select which object to create. The objects you create will automatically be selected.

### Primitive brushes

The first set of brushes is called Primitives. These have simple forms but the most used ones. Press "1", "2", "3", "4", etc. to generate different kinds of primitive brushes.

These are the shapes and their assigned keys:

- "1": Create cube
- "2": Create wedge
- "3": Create spike
- "4": Create spike wedge
- "5": Create cylinder, param0 = segments
- "6": Create cone, param0 = segments
- "7": Create sphere, param0 = x-segments, param1 = z-segments (use with care)
- "8": Create spike cube, cube made out of 6 spikes

The sphere is a tricky object and I can't recommend using it unless you save first or really know what you're doing. It's an object in Ogier that plays many tricks on you and is known to crash the program.

The spike cube is a cube made out of 6 spike primitives. Its purpose was to make a cube that could be bent and sewn in many ways, it works well but is kind of expensive seeing as it has a lot more faces than the normal cube, but then again it's not a normal cube. Try keeping to the regular primitives as long as you can.

### Custom brushes (advanced)

To create cylinder, cone and sphere brushes that has more faces than the default ones you have to enter the Create object menu. You can open the Create object menu by pressing

“Insert”, “<” or through the edit menu. Change the numbers in the Creation Params for different brushes. Then press “Close” and press “5”, “6”, or “7” to create the new shapes. Be careful with these settings though, save and try them before using them; some high numbers may crash the program. The param settings will stay until you change them again.

Param0: x- segments.

Param1: y-segments

Param2: not used

“5”: Create cylinder, param0 = segments

”6”: Create cone, param0 = segments

”7”: Create sphere, param0 = x-segments, param1 = y-segments

## **Spline brushes**

These brushes are a little different then the primitive brushes. They are more complex in their shapes and you can bend and skew them in many ways. (These may also be called curved surfaces by some, but we use the word spline)

These brushes have many faces that needs to be rendered so the objects created with them tend to become more complex in general then an object built out of primitive brushes.

These brushes have changed some over time and in the way they are compiled.

In The Chronicles of Riddick: Escape from Butcher Bay and later projects there are no difference between how the splines and primitive are rendered, but in Enclave and KoTT (Knights of the Temple) there is. In Enclave and KoTT the splines will be a little more demanding for the CPU and they will directly inflict on the game speed, so keep this in mind if you’re making a map for Enclave and KoTT.

Don’t be alarmed though, feel free to use them as much as you like but keep the game speed in mind then your building.

To generate the different kinds of Spline brushes Press Shift+“1”, “2”, “3”, “4”, etc.

These are the different shapes of the spline brushes:

Shift+‘1’: Create spline-cube

Shift+‘2’: Create spline-pie

Shift+‘3’: Create spline-cylinder

Shift+‘4’: Create spline-arc 1

Shift+‘5’: Create spline-arc 2

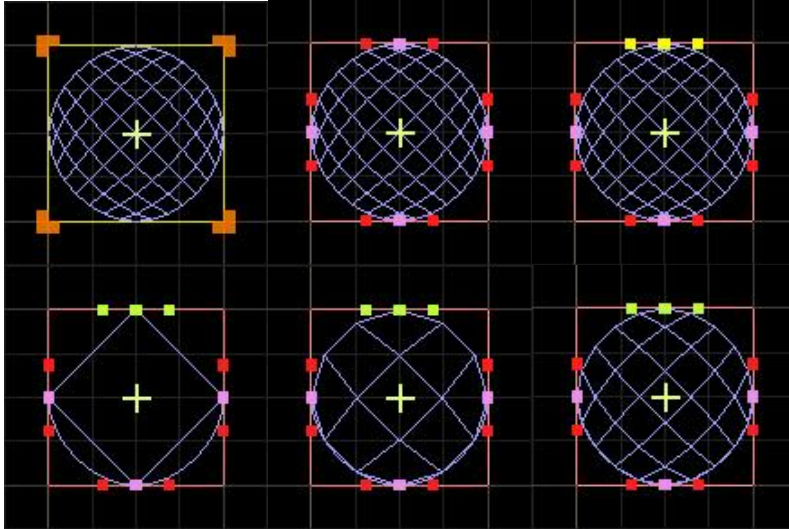
Shift+‘6’: Create spline face (use with care)

The spline face can only be selected from one side unless you drag the selection frame around it and select it that way, keep that in mind then you’re using these. They can’t be

scaled the same way as the other brushes seeing as they only have one side, that single face can be scaled though.

You can also change the detail of the spline brushes. I suggest that you toy around with these brushes. Select the edge tool with a spline brush selected and bend their shape and change the detail of them to get a feel of these brushes.

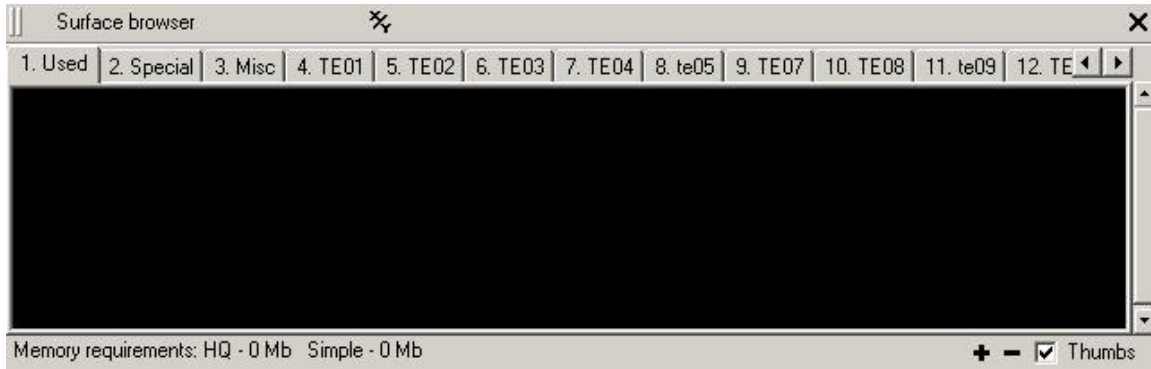
To change the detail of a spline you select the spline and go into edge mode. I am using a cylinder as an example here but this works with all types of splines.



Selects all or the handles you wish to modify and press “+” and “-“ to change the detail.  
“+” will increase the number of edge-points when using static tessellation  
“-“ will decrease the number of edge-points when using static tessellation

## Surface browser



To bring up the surface browser hold your mouse pointer over any of the 2D/3D views and press “~” or “§” depending on what keyboard setting you are using. You can find it at the top left corner of your keyboard, under Esc. Here you select textures and different kinds of characteristics a brush will have, if an object is supposed to be used as a ladder and so on, and you can have different texture packs in different views. The appearance of the surface browser and the name of the menus can change depending on what project you load. This one is from Knights of the Temple.





The number keys are used to select which surface group to display. The keys range from 1, 2..9, 0, and holding Ctrl while pressing any of those keys will select surface group 11, 12..19, 20.

By pressing the Right mouse-button, on a surface you toggle if the surface should be hidden or not. If a surface is hidden it means that all faces with that surface are skipped when rendering the 3D-view. This can be useful to find objects with certain textures and it can also be used to hide surfaces to speed up the preview in the 3D view. The edit surface function isn't implemented yet, but hopefully we will get it going some time soon.

Every time you bring up the Surface browser the thumbs setting will change, so if you want to see the textures in thumb mode or not just check or uncheck the thumb box in the bottom right corner or close and reopen the surface browser to change this on/off.

The icon at the top that look like this:  lets you minimize the surface browser so that only a bar will show in your 2D view. In minimized mode another icon will appear:  pressing this one will get you back to the surface browser.

You can use   to flip between the surface browser and 2D view but most people chose to use the “~” or “§”.

## Special surfaces

In the surface browser you will find a group called Special. In this group you will find “textures” that give brushes different attributes and textures that are used to create special functions.

*Note: The list of special surface may differ between different games.*

The ladder that was used in Enclave works well as an example how some of these work. When you create a ladder you first have to create a normal ladder with brushes and textures. Then you create a new brush that you texture with Phys Ladder from the special

menu, this brush can now be climbed on in game, so it's not really the ladder itself you're climbing on, it's the Phys Ladder brush. You can group and build with these objects like everything else in Ogier but most of them won't show in game, just their attributes. There are many different types of attributes given to a brush this way in Ogier and I'll try to explain some of them.

**Fit / Fit2:** These textures are used to do just that, to fit textures. Very handy to have when your uncertain of a textures fit. Just place the fit texture and match the two textures together, then you replace the fit textures with the one you want to use.

**Phys Ladder:** This will create a brush that you can climb. The brush itself won't show in game. (Enclave, KotT)

**AI off-limits:** This will create an area there the navigation grid won't be rendered for the AI. This can be useful to stop characters from path into certain areas. You can still force a character into this area with some game play options. The AI off limits won't affect the player character. You can also use this to block the navigation grid to be created in certain parts of the map to lower compilation time, but it has to be under certain circumstances, and better be left alone in most cases.

**No Overlap:** This is the texture we use for structure brushes, it won't show in game.

**No Draw:** The No Draw texture doesn't have any attributes and won't be rendered during the compilation. All sides of a brush that won't be shown in game should have this texture on it to make a map as effective as possible.

*Note: Having an entity with only No Draw will cause the compiler to abort since these faces are removed compile-time.*

**Player Phys:** This will stop both the player and AI without stopping for example projectiles. This is mainly used to make a area there the player risk getting stuck and to block the way for the char. Lets say that you have created a detailed area there you suspect that the player will get stuck in, then you can create a block that covers those parts to make the game play smoother.

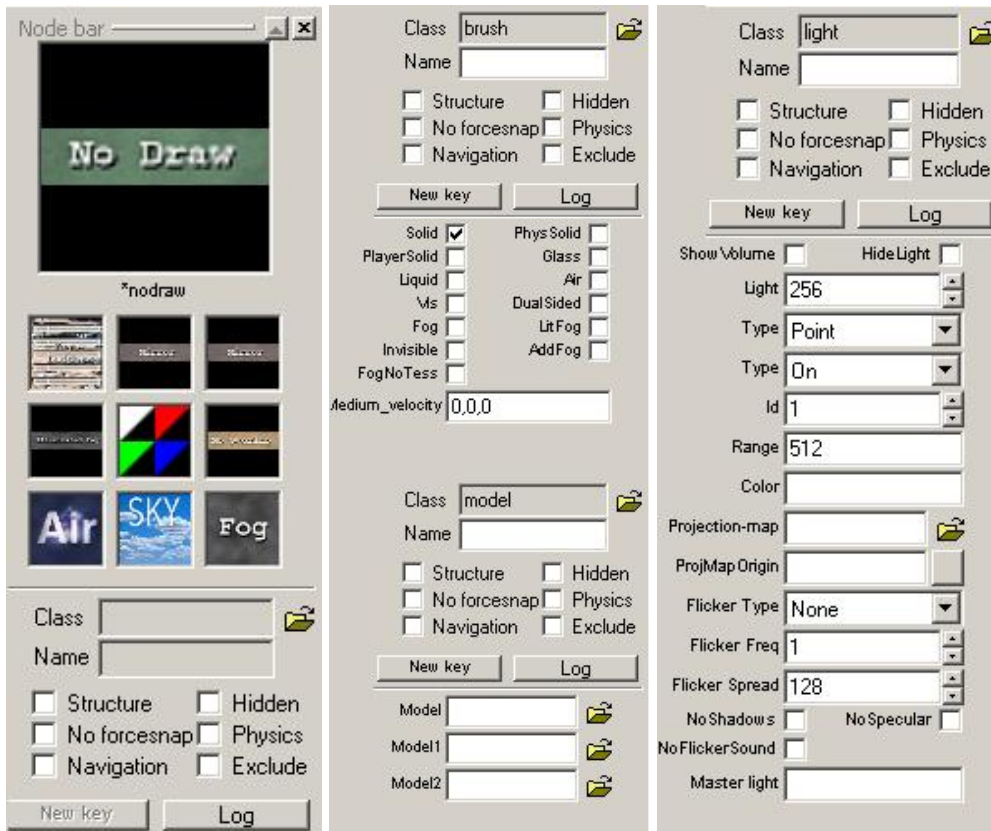
**Light Block:** This will create a brush that won't show in game that will hinder the light (Enclave, KotT)

## Nodebar

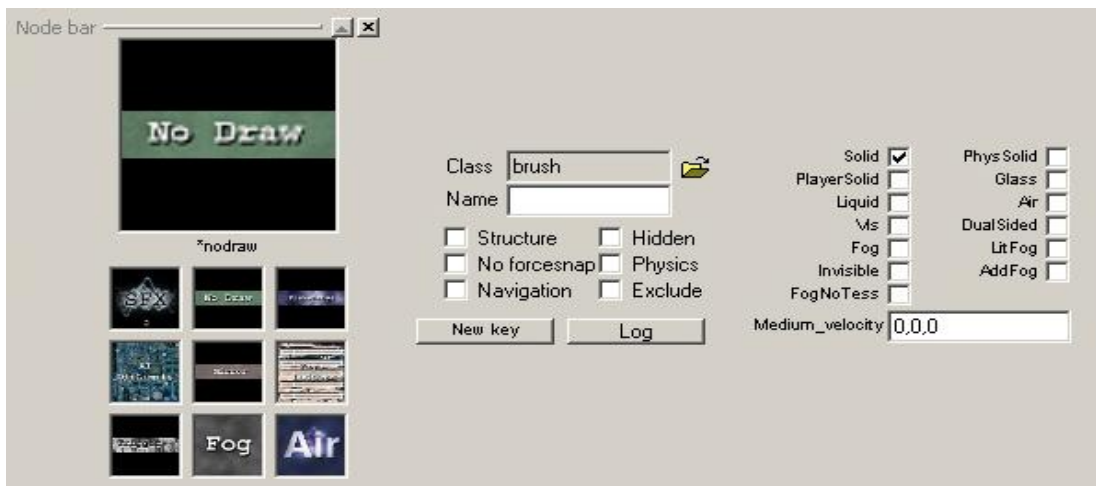
The Nodebar has a lot of properties and is an essential part of Ogier. Its here you get and set all settings for lights triggers and entities.

We wont go into all of these settings down below just now, we will take them as they come so don't worry about these settings just now. I just want to give you an idea of how the node bar can look.





This is far from all the settings you will get in contact with when using the node bar.



Top part

Middle part

Bottom part

## Top part of the nodebar

The top part of the node bar is used for texturing. The texture you have active will be shown at the top, the nine last textures used will be shown in the nine small frames below the selected texture.



Right-clicking on any of the texture icons will bring up this menu.

**Hide surface:** This is the same that you can find in the surface browser if you right click on one of the textures found there. This will hide all surfaces what has the selected texture.

*Note: If all faces of a brush are mapped with hidden surfaces, you must use **Show Hidden** to view and select it.*

**Lock surface:** This will lock the current texture show in the node bar so that you have it there until you unlock the texture. You can drag and drop to change the position of the textures in the nodebar.



Unlocked mode



Locked mode, notice the small frame around the texture icons, this indicates that the surfaces are locked.

**Select all faces:** This is not implemented yet.

**Edit surface:** This is not implemented yet.

## Misc. info and settings

### Grid size

Grid size is used to change the level of detail of your objects that you create.

Ogier supports infinitely small grid, but for beginners we recommend you never go below 1 in grid size.

Press "Q" to decrease grid size.

Press Ctrl+"Q" to increase grid size.

### Force snap / Grid snap

For inexplicable reasons vertexes may on occasion end up ever so slightly off-grid. This may cause trouble when compiling. To correct this you may select the entire level, or certain areas and objects and force snap them to a preferred grid size. Vertexes that are off grid or on less than 1 unit grid will show up as red (only visible in vertex and edge mode). Spline handles will be red ever so often, but I don't think you'll have to worry about those. Leave them red if you wish.

"F4": Force snap.

## Copying / Pasting

To copy and paste brushes models and other objects you can either use the short cuts below or through the edit menu. You can use copy/paste between different maps in Ogier.

Ctrl+C: copy.

Ctrl+V: paste.

Ctrl+P: paste nomove, paste in the exact same position as the copied object. This is extremely useful, but make sure you don't end up with several object in the same place.

Ctrl+Alt+P, Paste position

Ctrl+Alt+V Paste camera position

## Grouping

There are a few different ways to group objects. You can create Systemnodes and Nodes in Ogier and sort the objects in the workspace. The most common way to group objects is to use "G" this creates a node with the selected objects inside.

Press "G" to group several objects.

Press Shift+"G" to ungroup.

Press "F" to step down the group hierarchy.

Press Ctrl+"F" to step up the group hierarchy.

Ctrl+Alt+'F': Step into group with selection intact

NOTE: Try not to create systemnodes inside systemnodes. The functions inside them won't work properly if you do.

## Hide/Unhide objects

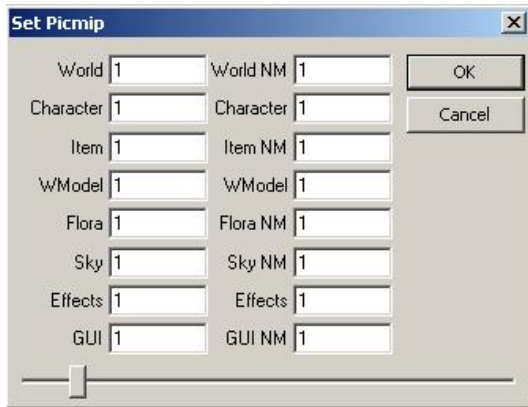
Press "H" to hide/unhide selected objects from view.

Use Ctrl+"H" to toggle view hidden brushes on and off

To hide/unhide all objects of the same class use Ctrl+Alt+"H".

## Set Picmip

You can scale the texture quality on the textures in Ogier. Enter the set picmip options under View. To change the number for all objects just use the level at the bottom.



## Structure

This is an art form. Structure is what the engine uses to find out what polygons to exclude when rendering the world. Basically, if the polygon is behind a Structure block it won't be rendered. The thing is that a Structure brush is extremely taxing for the CPU so unless it covers a lot of polygons it's cheaper to just render the polygons.

Press Ctrl+R to show only structure brushes.  
Press Ctrl+R again to go back to normal view.

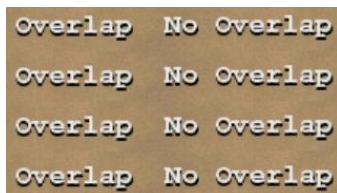
**NOTE:** A map will have to be encased in structure in order to compile, build a Structure box (six brushes) outside your level when you begin working on it.

**NOTE:** Structure should be built as primitively as possible with as few brushes as possible. Basically, if you build a room you want to encase with structure, see to it that the structure has as small holes as possible for the openings, doors, windows etc. Never make structure at an angle (unless you really know what you're doing). Let the brushes follow the grid lines.

A Structure brush should have the "No Overlap" texture on all faces. This will make it invisible in the engine as long as it is inside of a regular visible brush (the face of the structure block can be in the exact same space as a regular visible face if you wish).

You create a structure brush like this:

Create a brush, open up the surface browser and map the brush with No Overlap texture found under Special.



Select the brush and check the structure box in the Node bar (“E”, Ctrl+10).



Now you have a structure brush.

**NOTE:** You can use any texture you like on a structure, just mark the brush as structure in the Node bar but this can make it extremely hard to locate the structure so we recommend that you use the “No Overlap” texture for all your structures to begin with.

## User portals

**TODO**

Think of this as a dynamic structure block.

## The Create object menu

Create object menu contains a lot, models, info\_player\_start, lights and templates. It also allows you to make new objects.

## Classes

Here is a short description of some of the classes.

**Actioncutscene:** Used to set up custom ACS.

**Actioncutscenepickup:** An ACS version of the pickup.

**Actioncutsceneriot:** Become a RiotGuard.

**Cutscene:** Object that contains most settings for a cutscene.

**Dynamic:** These are used to calculate dynamic, each part needs to be a separate dynamic.

**Dynamiclight2:** A light that can move around ingame, this can be parented to elevators or other movable objects.

**Emitter:** Creates pre-defined entities with a set of parameters.

**Engine-**

Door: Swinging door.

Path: Very useful entity. Can be scripted from and used as AI patrol paths or even hand animated world entities like doors, hatches and machinery.

Rotate: Auto rotating entity.

**Script:** This is a version of the engine\_path used for scripting.

**Wheel:**

**Explosion:** This can be used if you want to create an explosion.

**Func\_userportal:** Structure that can be toggled on and off. Used in doors and portals.

**Healthstation:** Use this if you are constructing a nanomed.

**Hook:** Attach this entity to another hook or engine-object.

**Hook-**

**Norotate:** This entity will only take the rotation from the attached object

**Target:** This entity will attach to one object and try to rotate to be attached to another.

**ToLine:** This entity will attach to one object and try to rotate to stay within a line defined by two attach points (vertices).

**ToCircle:** This entity will attach to one object and try to rotate to stay within a circle defined by two vertices attach points (vertices).

**Info\_player\_start:** This is where the player starts on the map. (If there are more places to start use the default setting to define). There are different kinds of starts for different modes, like team starts, this depends on what entities are loaded for the current project.

**Ledge:** Used to define climbable places. Recommended heights are: 48, 64 and 94 units.

**Leveractioncutscene:** Use this on a lever.

**Light:** Please use light2.

**Light2:** Light version 2.

**Model:** Use this if you want to place a model somewhere in the world.

**Reinforcementcontrol:** Send message to this when a creature is dead to get it to spawn a new one. (Used in conjunction with reinforcementpoint)

**Reinforcementpoint:** Used to create new enemies when the first ones are dead, use only with creatures that GIBS and get removed from the game.

**Scenepoint:** Default scenepoint, has no pre-defined settings.

**Scenepoint-**

**Cover:** Pre-set scenepoint for cover. The circle is the area to be within and the cone is area where the threat is.

**Roam:** Pre-set scenepoint for roaming. Good scripting for IDLE AIs.

**Search:** Pre-set scenepoint used for guards searching. Circle is the standing point and the cone is the search angle.

**Tactical:** Pre-set scenepoint for attacking. When threat is within the SPs cone the scenepoint is valid. AIs will try not to use SPs behind the player.

**Sky\_indoor:** Set clear color and fog settings.

**Sound:** A point where a sound originates from.

**Systemnode:** A group of entities. The systemnode can relay messages to its children ingame. NOTE: Try not to create systemnodes inside systemnodes. The functions inside them won't work properly if you do.

**Team:** Set up communication and other settings for teams.

**Wallmark:** Spays a wallmark. This is a bit CPU intensive. Recommended you build a spline face with the wallmark on to optimize.

## Custom brushes

To create cylinder, cone and sphere brushes that has more faces than the default ones you have to enter the Create object menu. You can open the Create object menu by pressing "Insert", "<" or through the edit menu. Change the numbers in the Creation Params for different brushes. Then press "Close" and press "5", "6", or "7" to create the new shapes. Be careful with these settings though, save and try them before using them, some high numbers may crash the program. The param settings will stay until you change them again.

Param0: x- segments.

Param1: y-segments

Param2: not used

"5": Create cylinder, param0 = segments

"6": Create cone, param0 = segments

"7": Create sphere, param0 = x-segments, param1 = z-segments

"Insert", "<": Create object menu

## Lighting

Lighting is different depending on the technology used.

### Lighting with Lightmaps (Enclave, KotT)

**Todo**

### Lighting with Unified Lighting (Riddick)

This is an art form just like structure. Depending on what platform and power of the machine your building the map for you has to restrict the number of overlapping light volumes. The volumes range is determined by the range given to it and is the same as shown in the range option. Sometimes it can be better to create few lights with a larger range then adding small lights that have a short range. The range of the light can be seen if you select the show volume option; the volume of the light is show as a yellow box. We recommend that you don't use more then max 3 overlapping volumes because this is a major factor that directly affects the frame rate in game. It also effects the compiling time. Structure blocks will cut of light volumes, so you have to worry about a lights range that is blocked of by structure even though the light volumes seam to cross.

An easy way to see the light volumes are if you select a light in the map and select all of class under the Edit menu. Then press Ctrl+"R" to show structure. Now you will see all structure and all the light volumes at once, this can help you to get a over view of your light volumes.

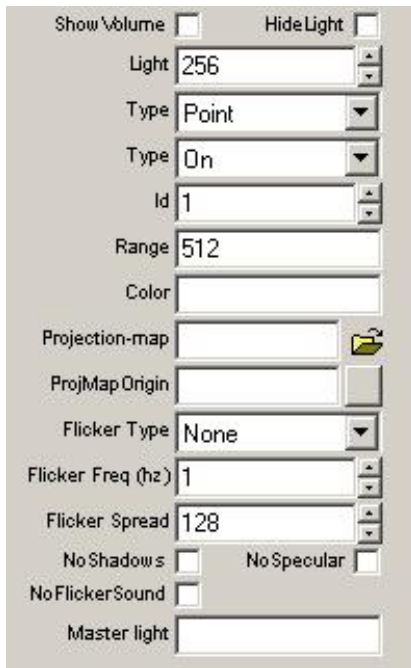
### View XW

You can get a preview of how the light will show inside the game but to get a more detailed version you have to compile the map and view as XW. To see how lights interacts more in-game like inside Ogier you need to compile a map with all its brushes then selects to view as XW under the View menu. XW needs a compiled map to render the light towards.

This may be broken for versions of the editor that are not close to the release date of the game.



## Light options



The screenshot shows a 'Light options' window with various settings. At the top are two checkboxes: 'Show Volume' (checked) and 'Hide Light' (unchecked). Below these are several input fields and dropdown menus: 'Light' (256), 'Type' (Point), 'Type' (On), 'Id' (1), 'Range' (512), 'Color' (empty), 'Projection-map' (empty), 'ProjMap Origin' (empty), 'Flicker Type' (None), 'Flicker Freq (hz)' (1), 'Flicker Spread' (128), 'No Shadows' (checked), 'No Specular' (checked), 'No Flicker Sound' (checked), and 'Master light' (empty). The 'Type' field is unique as it has two separate dropdown menus.

### Basic light options

**Show Volume:** Shows the light volume, the volume is linked to the range of the light. The volume is shown as a yellow box or cone depending on what type of light you're using. Make sure that you don't have too many light volumes overlapping each other because this is one of the major factors that directly affect the game speed and compilation time.

**Hide Light:** This hides the light, very useful when you want to see how other lights interact in the map. This only works when the map is viewed in the XW mode.

**Light:** The lights intensity, changing this will change how bright a light will glow.

#### Type:

Point light will create a light that radiate in shape of a cube. If you set the range to 200 the light volume will be 200 units up, down, left right from the origin of the light source. Spot light will create a cone shaped light map and works as a spotlight or flashlight. The volume will be set by the range. With the spot light you will get the option to change its width and height as well as its range.

**Type (2):** This sets the initial state of the light, if it starts on or off.

**Range:** This option sets the range of the light in units. This affects the light volume for the light types.

**Color:** To change color just click in the color field and a color menu will show. To make custom colors just change the settings, don't forget that you need to change the luminary at the right hand of the window as well. You can save different colors for later use, just select one of the empty slots or change a current one under custom colors and then press add to custom colors button. Try to match the color of your lights to the sky settings and such to make the scene more realistic.

## Dynamic light options (Riddick)

**Projection-map:** If you want an object that isn't a solid object in front of the light to cast a shadow you need to attach a projection map to the light. The browse option is broken at the moment, hopefully it's fixed when you read this, but until then you need to enter the path and name to the map manually.

**ProjMap Origin:** This sets the origin of the projection map, press the button next to the field to get the origin of the selected object.

**Flicker Type:** You can select None, Pulse, Sine and Random Pulse.

These are different flicker types a light can have.

**Flicker Freq (Hz):** This changes the frequency that the flickering light has.

**Flicker Spread:** If you want the flicker to appear as random you can use flicker spread. Set a number between 1 and 256.

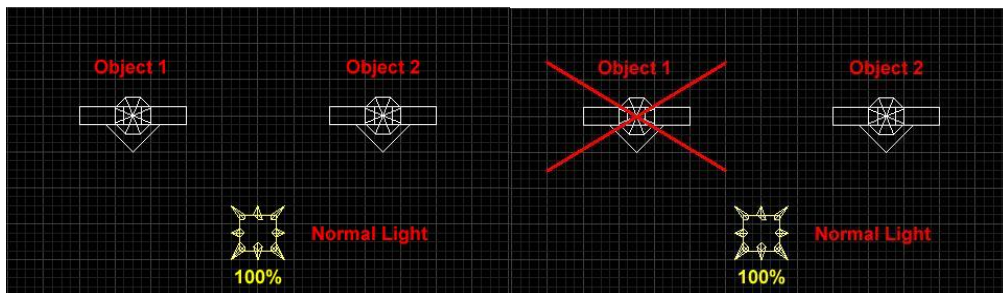
**No Shadows:** Change this if you want your light to generate shadows or not.

**NoSpecular:** This will enable/disable if a light will render specular on models.

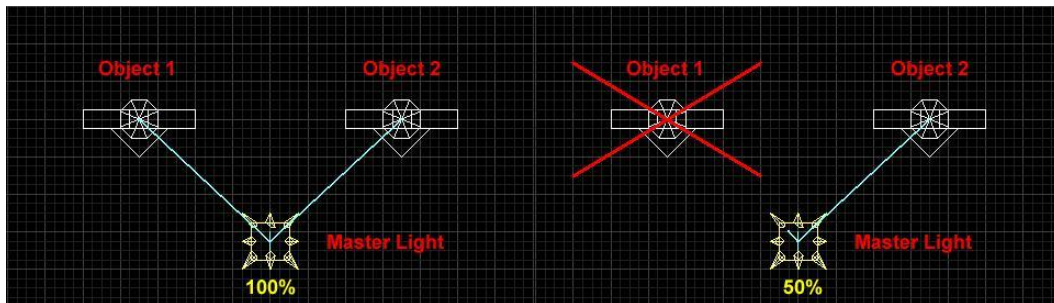
**NoFlickerSound:** This will turn the sound that a flickering light makes on/off.

### Master light

To create an illusion that two or more objects glow you can create a master light for the area. Let's say you want two objects appearing as they were glowing but you only want to use one light source. Object 1 and Object 2 will be light templates. This isn't a problem unless you want the objects to be destroyed.



If Object 1 is destroyed and we use a normal light it would still glow with its set number.



If we use a master light we can connect the two objects to the light. Once an object is destroyed it sends an impulse to the master light telling it to change its light output. To link objects to a master light you have to name the master light. Select the objects (light templates) and enter the master lights name into the master light field. You can connect more lights to the master light then shown in this example. This can be used in many other ways.

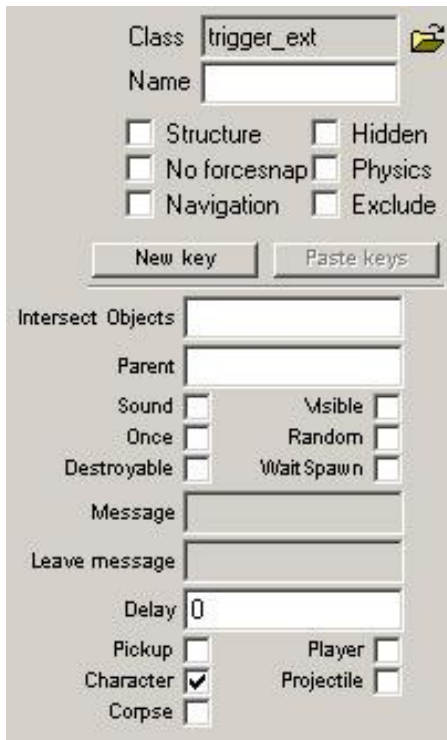
## Scripting

There are a lot of different ways to solve a gameplay problem, I will bring up some examples in this doc and just because I say something or try to explain a certain situation doesn't mean that I do it the right or best way. There are just too many ways to make a scenario work.

## Triggers

### Trigger\_ext

This is probably the most used trigger. This trigger can activate when someone or something comes in contact with it. \$activator is used to send a message through or to the one activating the trigger.



The screenshot shows the configuration window for the 'trigger\_ext' class. It includes fields for 'Class' (set to 'trigger\_ext') and 'Name'. Below these are several checkboxes: 'Structure', 'Hidden', 'No forcesnap', 'Physics', 'Navigation', and 'Exclude'. There are also buttons for 'New key' and 'Paste keys'. The 'Intersect Objects' field is empty. The 'Parent' field is also empty. The 'Sound' checkbox is checked, and the 'Visible' checkbox is unchecked. The 'Once' checkbox is checked, and the 'Random' checkbox is unchecked. The 'Destroyable' checkbox is checked, and the 'Wait Spawn' checkbox is unchecked. The 'Message' field is empty. The 'Leave message' field is empty. The 'Delay' field is set to 0. The 'Pickup' checkbox is unchecked, and the 'Player' checkbox is unchecked. The 'Character' checkbox is checked, and the 'Projectile' checkbox is unchecked. The 'Corpse' checkbox is unchecked.

**Sound** this is used when you want a sound to activate the trigger, a gunshot etc.

**Visible** this makes the visible ingame (but currently not implemented)

**Once** if this is selected the trigger can only be triggered once. The trigger will then be destroyed.

**Random** if this is selected the trigger will send a random one of multiple defined messages.

**Destroyable**, this opens up a new menu where the destroyable settings are shown. This is described in more detail later in this doc.

**Wait Spawn**, this is used when you want the trigger to be waitspawned.

**Message** the message that will be sent every tick when it's triggered.

**Leave Message** the message that will be sent when the activator leaves the trigger.

**Delay** if a delay is set a new message field will show where you can send a delayed message.

**Pickup** if a pickup is in the trigger it will activate.

**Player**, if you want the player to be the only one to activate the trigger you will have to select this one and remove the Character one.

**Character**, if you want all characters to be able to trigger this should be checked. A character can be disabled with impulses or in its character file not to trigger triggers aswell.

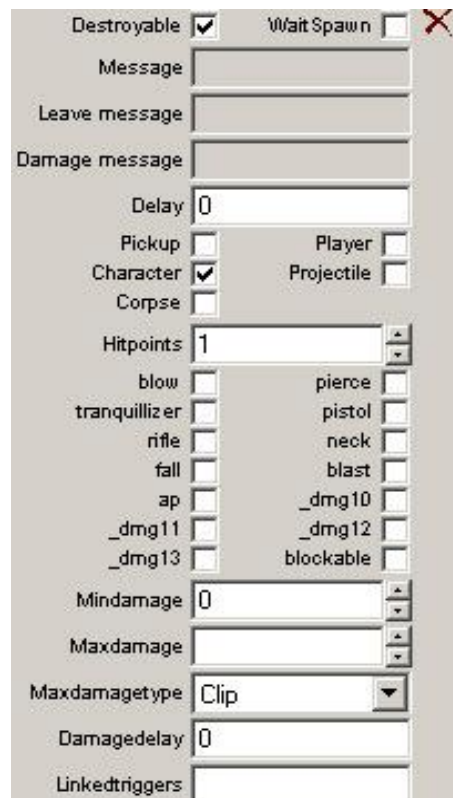
**Corpse**, use this if you want a corpse to be able to activate the trigger.

**Projectile** is used if you want projectiles to trigger, might be an Enclave feature only unless you make the trigger destroyable.

### Trigger\_ext - Destroyable

This opens up a new menu where the destroyable settings are shown.

This is used when you want to create a trigger that is activated when you fire at it etc.

The image shows a screenshot of a game's configuration menu for a 'Trigger\_ext - Destroyable'. The menu is organized into several sections. At the top, there are two checkboxes: 'Destroyable' (checked) and 'Wait Spawn' (unchecked). Below these are three text input fields for 'Message', 'Leave message', and 'Damage message'. A 'Delay' field is set to '0'. There are four checkboxes for activation types: 'Pickup' (unchecked), 'Player' (unchecked), 'Character' (checked), and 'Projectile' (unchecked). A 'Corpse' checkbox is also present and unchecked. The 'Hitpoints' field is set to '1'. Below this is a grid of checkboxes for various damage types: 'blow', 'tranquillizer', 'rifle', 'fall', 'ap', '\_dmg11', and '\_dmg13' on the left; 'pierce', 'pistol', 'neck', 'blast', '\_dmg10', '\_dmg12', and 'blockable' on the right. The 'Mindamage' field is set to '0', and the 'Maxdamage' field is empty. The 'Maxdamagetype' is set to 'Clip'. The 'Damagedelay' field is set to '0'. At the bottom, there is a 'Linkedtriggers' field.

**Message** when the triggers hitpoints runs out this message will be sent.

**Leave message** the message that will be sent when the activator leaves.

**Damage Message** the message that will be sent when the trigger is "touched".

**Hitpoints** sets the number of hitpoints the trigger will have.

**Blow, Pierce, Tranquillizer, Pistol, Rifle, Neck, Fall, Blast, Ap, \_dmg10, \_dmg11, \_dmg12, \_dmg13, Blockable** these are damage-types. When a damage-type is set the trigger will only accept damage from that type.

**Mindamage** sets the max damage that the selected attacks will do.

**Maxdamage** sets the min damage that the selected attacks will do.

**Maxdamagetyp** select a max type that will trigger it.

**Damagedelay** if this is set, the trigger will wait this duration until it can be triggered again from damage.

**Linkedtriggers** when the trigger is activated it also activates the linked triggers.

## Trigger\_changeworld

This can be used to change world/map. The spawning point can be specified as well.

Ex

<Map name>:<name of the info player start>

map02:map02start



## Trigger\_checkpoint

This trigger will activate a checkpoint. There are times when checkpoints will not be activated, for example when the player is in combat.

## Trigger\_accpad

This was used in the first versions of Enclave. It's an acceleration pad much like the ones used in Quake3 etc.

## Trigger\_aidie

Obsolete, this will be removed.

## Trigger\_restrictedzone

This is used to restrict the player and what equipment he can wear etc.

Use this if you want to define areas there the player can or can't move with different types of weapons etc. EX of this is in Riddick is Copland there the player need a higher restriction to move around, this is done by equipping the "police suit" that gives the player a higher clearance.

## Trigger\_sneakzone

Trigger sneakzone is used in riddick to define some special areas there the game was forced into sneakmode. Sneakmode will change the attributes of the AI's etc to make it easier to sneak, not recommended to use unless it absolutely necessary.

## Trigger\_softspot

This was/is used create safe falls for the player so that the player doesn't take damage on high falls.

### **Trigger\_specialgrab**

Specifies areas where specialgrab is used, wasn't fully used in Riddick but this defines where special animation will be played instead of the normal neck-break like falling off a cliff or being pushed into a grinder etc.

### **Trigger\_surface**

This is some old stuff that can be used as a projectile trigger.  
You need to set a trigger texture on this to make it work though.

## Engine/Hooks

### Paste view

To paste an EP in the current 3D position you use Ctrl+Alt+V.

This is a very useful thing when you're creating a cutscene etc. Let's say you want to create a fly-by of something. Create an EP, set a new time; select the view in the 3D window you want and Ctrl+Alt+V. Repeat till the sequence is finished.

### Destroytime

To destroy/remove an engine path in ogier at a set time you can enter New Key "Destroytime xxx" there xxx is the time when you want the EP to disappear. This is very useful when you're working with dynamics.

## Engine



### Engine\_path

Engine path is probably the scripting tool used the most.

If you create game play etc this is the main thing you will use, it has a number of functions and can be used to pretty much everything.

The screenshot shows the 'engine\_path' configuration window. It has a 'Class' dropdown set to 'engine\_path' and a 'Name' text field. Below these are checkboxes for 'Structure', 'Hidden', 'No forcesnap', 'Physics', 'Navigation', and 'Exclude'. There are 'New key' and 'Paste keys' buttons. The 'Model' field is empty, while 'Model1' and 'Model2' have folder icons. 'Light minlevel' is a black bar, and 'LightingMode' is set to 'Auto'. A group of checkboxes includes 'Game Physics', 'Auto Start', 'Auto Reset', 'Loop', 'Auto Reverse', 'Pushing', 'Push Sync', 'Unreliable', 'Never Interrupt', 'vis When Stopped', 'No Block Nav', 'Wait Spawn', 'Once', and 'Use Hint'. An 'Attach' field with a document icon is at the bottom left. On the right, 'Type' is 'Linear', 'Sequence' is '1', and 'Time' is '0'. A vertical stack of buttons includes 'Del sequence', 'Clear sequence', 'New', 'Delete', 'Next', 'Prev', 'First', 'Last', 'Maya import', 'Insert time', 'Delete time', 'Scale time', and 'Average speed'. At the bottom right are 'Timed sound' (with a folder icon), 'TimedMsg', and 'Jump' fields.

**Model:** Model selection.

**Model1:** Model selection.

**Model2:** Model selection.

**Light minlevel** used in enclave and KotT to correct dynamics etc.

**LightingMode** was used in enclave and KotT to correct dynamics etc.

**Auto**

**Lightmap**

**Gouraud**

**GamePhysics** this is used if you want the object to have a physical form in the world, it has to be checked if you want to attach certain models etc to the EP.

**AutoStart** select this if you want the EP to autostart.

**AutoReset** is used when you want the EP to be able to run more then one time.

**Loop** sets the EP to loop.

**AutoReverse** can be used for example swing-doors that will reverse if something is in the way

**Pushing** will allow the EP to push the player, if the EP fails to push the obstacle the EP will pause.

**PushSync** with this selected the EP will force the push and cause damage to the obstacles.

**Unreliable** this is sometimes necessary when having complex systems of engines and hooks

**Neverinterrupt** when this is set the engine\_path will still run a complete sequence before stopping when it receives a stop message.

**InvisWhenStopped**, when this is set the engine\_path will be invisible as long as it is not moving.

**NoBlockNav** if it's supposed to block the nav grid or not.

**WaitSpawn** if you want the EP to be waitspawned.

**Once** the EP will only run once.

**UseHint** will cause the object to flash.

**Attach** if you want the EP to be attached to a certain object.

Ex 0,0,0,object

**Type**

Different types of paths.

These are used in different situations and they have some different attributes.

**Linear** is a linear type of path.

**Spline** will create smooth paths and perhaps a little more natural looking.

**Step** lets the path jump to the next position instead of going through the path over time like it does with linear

**Backmanspline** is a different type of spline path.

**Sequence** select what sequence you're working on, multiple ones can be created in the same EP. These are impulsive with the sequence number as impulse number. There are some problems if you have sequences with timed messages so keep that in mind if you create more than one sequence. There will or rather is a fix for this so that different sequences can have timed keys to it but I am not sure it's in the version of ogier you're using.



**Time** the timeline of the sequence this is a little tricky to get right the way it is atm. It will show what time in the sequence you're at etc.

**Del sequence** deletes the sequence.

**Clear sequence** clears the sequence.

**New** create a new time stamp.

**Delete** delete a time stamp.

**Next**, jumps to the next time stamp.

**Prev**, jumps to the previously time stamp.

**First**, jumps to the first time stamp.

**Last**, jumps to the last time stamp.

**Maya import** this will import a path from maya. This requires maya plugin and that the path is saved at a specific location.

**Insert time** insert time to modify the timeline between two stamps.

**Delete time** remove time between two stamps.

**Scale time** scale the time in percent.

**Average speed** is used to averedge the speed/time of the whole path.

**Timedsound** time a sound to the path.

**TimedMsg** used to send messages at certain times in the path, a new field will be created when the first one is used.

**Jump** is used if your doing a cutscene etc and you don't want the fade inbetween the two different EP's.



## Engine\_script

This is pretty much a toned down Engine Path.

The Engine Script dosnt have the ability to draw paths etc but this is very usefull when you want to keep track of the timing of messages etc.

## Engine\_door

There are many ways to create doors and this is one of them.

Sequence	1	Sound Open		
Angle	90	Sound Move		
Axis	0,0,1	Sound Openstop		
Swing time	2	Sound Shut		
Open time	5	Sound Loop		
		Userportal		

**Sequence** selects sequence.

**Angle** defines the angle.

**Axis** defines the direction the door should be opened in.

**Swing time**, swing time in seconds.

**Open time** how long the door should be open.

**Sound Open** defines the sound at the specific time.

**Sound Move** defines the sound at the specific time.

**Sound Openstop** defines the sound at the specific time.

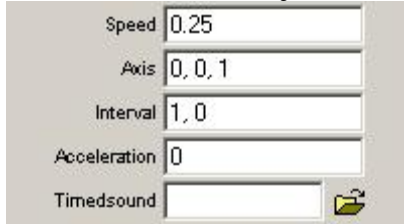
**Sound Shut** defines the sound at the specific time.

**Sound Loop** sounds that is playing as long as the door is in motion (broken?)

**Userportal** specify what user portal to use. A more detailed description of how userportals work can be found [here](#).

## Engine\_rotate

This will create an object that will rotate. Its speed etc is defined by these settings.



Speed	0.25
Axis	0, 0, 1
Interval	1, 0
Acceleration	0
Timesound	

**Speed**, 360 degrees / second

**Axis**, what axis it should rotation around

**Interval**, defining time running / time stopped durations

**Acceleration**, defines acceleration of the rotation

**Timesound**, synchronized sound to the engine\_rotate

## Engine\_wheel

Test of implementing a wheel that should rotate automatically when moved. Never used.

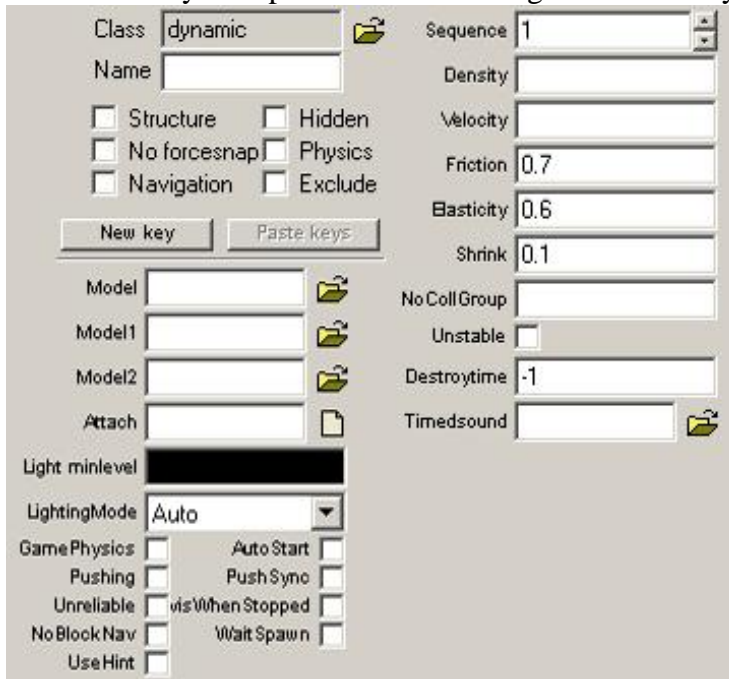
## Hooks

TODO

## Dynamics

Dynamics is used to create or rather simulate physics.

This is one of those things you need to fiddle with to get a feel for. I can't tell anyone how to make this look good, you need to try it out, change the settings and work from there. I will try to explain what the settings do and how you can work with them though.



The screenshot shows a 'Dynamics' settings panel. At the top, 'Class' is set to 'dynamic' with a browse button. Below it is a 'Name' field. A group of checkboxes includes 'Structure', 'Hidden', 'No forcesnap', 'Physics', 'Navigation', and 'Exclude'. There are 'New key' and 'Paste keys' buttons. Below these are fields for 'Model', 'Model1', 'Model2', and 'Attach', each with a browse button. Further down is a 'Light minlevel' color picker and a 'LightingMode' dropdown set to 'Auto'. A bottom section contains checkboxes for 'GamePhysics', 'Pushing', 'Unreliable', 'NoBlockNav', 'UseHint', 'AutoStart', 'PushSyno', 'visWhenStopped', and 'WaitSpawn'. On the right side, there are input fields for 'Sequence' (1), 'Density', 'Velocity', 'Friction' (0.7), 'Elasticity' (0.6), 'Shrink' (0.1), 'No Coll Group', 'Unstable' (checkbox), 'Destroytime' (-1), and 'Timedsound' with a browse button.

**A lot of the keys are the same as engine\_path, so refer to that one.**

**Density** set the density of the object.

**Velocity** with this you can set a velocity in world coordinates.

**Friction** sets the friction of the dynamic.

**Elasticity** sets the elasticity on the dynamic.

**Shrink** used to shrink the dynamic to create a gap for the parts to fall apart.

**NoCollGroup** this is a string that when matched on other objects will exclude them from colliding with eachother.

**Unstable** if you want the dynamics to be unstable.

**Destroytime** used to remove debris etc that you don't need. The objects will be removed after the time set.

**Timedsound** if you want to time a sound when the object is started.

### Example:

Create a primitive brush and change its class to dynamic by clicking on the browse button next to the class name (the brush has to be selected). You can texture it before you change it to a dynamic or after.

The brush should now have a purplish color when not selected to indicate that this isn't an ordinary brush and the nodebar should display the options shown in the picture above.

Now you need to define what the dynamic should collide with like the floor, wall etc. This is done by checking the physics flag in the upper part of the nodebar. Select the floor etc and check their physics flags. If you want the dynamic parts to collide with each other you need to select the physics flag for the parts as well.

If you select the unstable part the dynamic should start falling when you run a simulation. You need all the parts selected you want to be part of the simulation selected when you do this.

**Alt-T** is used to run and record a simulation.  
**Alt-R** is used to play the recorded simulation.

To make dynamics to fall as you want them to you can create brushes with the physics and exclude flags checked, this way they will have physics in ogier but the brushes wont be rendered into the .XW file. These can be used to create invisible foundations that give the dynamics different falling patterns etc.

You can also use these to push dynamics.  
Let's say you want to create a wall that is destroyed by an explosion.  
First you create, and turn the parts that make the wall into dynamics.  
Then you create a brush with the physics and exclude flag checked and turn it into an Engine Path. This way you can control the force and size of the pushing object.

The EP will push the dynamic parts if you have the dynamics; EP selected with the right flags checked when you run the simulation. This way you can direct the pushing part (EP) into the dynamics, the size of the object is determined by the size of the brush you created and the force by the speed and time that you move the EP into the dynamics.

### **Attaching models**

You can change the class of the dynamic into an EP without removing its rendered path. You can then attach EP's to the EP with a model etc to make the model follow the path. This works best if the dynamic had the shape of the model you want to attach to the EP.

**NOTE:** The cutting tool is very useful then creating dynamics.

Ctrl+Alt+RightMouseButton will enable the cutting tool. This can be used to cut in brushes. The brushes created will be grouped so you might have to ungroup them.

There should also be an .xmp with examples on the forum hopefully that will give you a better idea of what I am trying to explain here.

## Characters

The screenshot shows a complex configuration window for character AI. It is organized into three columns of controls. The first column, labeled 'Model', contains fields for selecting a base model and its parts, a dialogue file, and checkboxes for various spawn and movement behaviors. The second column, labeled 'AI', contains fields for AI-related parameters like lightmeter, drop item, and various 'On' events. The third column, labeled 'Behaviour', contains checkboxes for activation, start states, alarm, hit, and death events, as well as animation and testing options.

**Model:** Selects the base model for the character

**Model1:** Selects the second model, typically the head

**Model2:**

**DialogueFile:** Select the dialogue file that this char will use.

**Wait Spawn:** If this is checked the character won't spawn until it's told to.

**SpawnDead:** Character will spawn as dead.

**SpawnCrouched:** Character spawns crouched. NOT RECOMENDED unless you want the bot to do the duck-walk forever.

**NoDialogueTurn:** The character won't turn to face the speaker.

**NoDeathSound:** The character should not do any sound upon death.

**NoAutoaim:** Console auto aim does not apply to this bot.

**Global alarm:** Send alarm not only to team members in communication but to all team members (Riddick).

**Item 0:** Selected item, typically the weapon in hand.

**Easy:** Spawn the char during easy game difficulty.

**Normal:** Spawn the char during normal game difficulty.

**Hard:** Spawn the char during hard game difficulty.

**NoCharColl:** The char won't collide with other characters.

**NoWorldColl:** Wont collide with world (such as floors) don't forget to turn off gravity as well.

**NoProjectileColl:** The char won't get hit by projectiles.

**NoGravity:** Character is not affected by the downward force of gravity

**Immune:** Won't take any damage.

**MediumMovement:** If the character should be affected of movement of mediums like flowing water.

**Immobile:** Cannot move at all, no movement from actions or scripting will affect the bot, nor will recoils or blasts.

**Laserbeam:** Show laserbeam on helmet (Riddick).

**Stuntable:** IS affected by the stun gun (Riddick). The char can be stunned.

### **Flashlight**

**On:** Spawn the char with its flashlight turned on.

**Off:** Spawn the char with its flashlight turned off.

**Dynamic:** Spawn the char with a dynamic flashlight that can be turned On/Off depending on surrounding light.

### **Ai\_lightmeter**

**Off:** Don't measure light

**Basic:** Measure light but ignore shadowing objects

**Shadows:** Measure light with shadows.

**Dropitem:** Specify what pickup the char will drop on death.

**OnPickupWeapon:** Script something when the char is picking up a weapon.

**Dialogueimpulse0:** What impulse to send to the dialogue system when the player approaches and presses to talk.

**OnImpulse:** Run a script when an impulse number is sent to the character

**OnSpawn:** Send a message/impulse when the char spawns.

**OnTakeDamage:** Send a message/impulse when the char is taking damage.

**OnDie:** Send a message/impulse when the character dies.

**Team:** Specify what team the char belongs to. The characters can belong to multiple teams. A new team field will be created automatically when the first one is used.

### **Default Action TEMP**

Default actions are idle actions the character performs when nothing more urgent is at hand.

**Patrol:** Tell the char to patrol as default action.

**Explore:** Set the char to explore as its default action.

**Hold:** Tell the char to hold as its default action.

**Engage:** Tell the char to engage as its default action.

**Behavior:** Run a behavior.

**Position Object:** Used for Hold action to have the character hold at the object instead of at the spawn position

### **Minimum Stance**

Stances show how hostile the character is in his pose. Min forces a min stance on characters doing the patrol path default action.

**Default:** Let the action system decide what stance to hold.

**Idle:** General lazy stance.

**Searching:** This stance is used when searching for a target. Head swivels hither and back, looks good with flashlight equipped guns.

**Wary:** Gun up in semi ready stance to fire, a stance to take when the enemy is near but out of sight.

**Hostile:** Full combat stance.

### **Idle Speed**

An idle speed tells how fast the character should move as default. This can be overridden by actions.

**Default:** Default speed, set by the template.

**Walk slow:** The char is walking slowly.

**Walk:** Walking with “normal” speed.

**Walk fast:** Walking fast.

**Run:** The char is running.

### Alertness

How alert the character starts out as.

**Oblivious:** Unconscious and comatose, doesn't react to anything except Alertness changes through scripts.

**Sleeping:** Cannot see, reacts very slowly.

**Drowsy:** Can see, reacts slowly

**Idle:** Reacts normally

**Watchful:** Fully alerted

**Start activated:** if you want the character to start activated.

**OnStart: Pause,** pauses the AI at start.

**OnStart: Immune,** AI is immune at start.

**OnStart: Aggressive,** Aggressive at start.

**OnAlarm: Release,** will be released when AI's alarm is activated.

**OnAlarm:** specify what will happen when the AI is alarmed.

**OnHitUninjured:** message that is sent when the AI gets hit without taking damage.

**OnInjured (outdated):** outdated

**OnDie (outdated):** outdated

**OnSpawn (outdated):** outdated

**Anim State Test:** preview an animgraph animation.

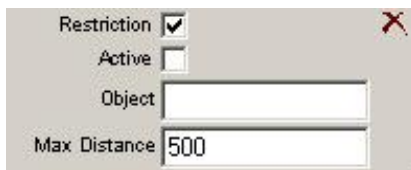
**BehaviourTest Type:** select behaviour type.

**General B.:** select animation type to preview.

**Force Anim:** preview raw animation.

**Restriction:** Restricts the character to not go beyond Max Distance of Object (or spawn pos if no object is assigned)

**Active:** if the restriction is active or not.

A screenshot of a UI panel with a grey background. It contains four controls: a 'Restriction' checkbox which is checked, an 'Active' checkbox which is unchecked, an 'Object' text input field which is empty, and a 'Max Distance' text input field which contains the number '500'. A red 'X' icon is visible in the top right corner of the panel.

Object: Select the object.

Max Distance: Define the distance from the object.

### Show Advanced:

Show the advanced character options

The screenshot shows a configuration window with a title bar containing a close button (X). The window has a 'Show Advanced' checkbox which is checked. Below it are several input fields and checkboxes:

- OnEnemySpotted**: An empty text input field.
- OnSpotPlayer**: An empty text input field.
- OnBadlyInjured**: An empty text input field.
- OnMeetPlayer**: An empty text input field.
- Alertness range**: A text input field containing the value '100'.
- Collision rank**: A text input field containing the value '25' with up and down arrow buttons on the right.
- Sight range**: A text input field containing the value '1200'.
- Hearing range**: A text input field containing the value '1200'.
- Field of view**: A text input field containing the value '360'.
- Awareness**: A text input field containing the value '0'.
- Patience**: A text input field containing the value '5.0' with up and down arrow buttons on the right.
- Reaction time**: An empty text input field.
- clumsy**: A checked checkbox.
- flyatpathspeed**: An unchecked checkbox.
- unspotfast**: An unchecked checkbox.
- saveplayerinfo**: An unchecked checkbox.
- resetplayerinfo**: An unchecked checkbox.
- scriptmute**: An unchecked checkbox.
- switchweapon**: An unchecked checkbox.

**OnEnemySpotted:** Send a message/impulse when the char spots an enemy.

**OnSpotPlayer:** Send a message/impulse when the char spots the player.

**OnBadlyInjured:** Send a message/impulse when the char becomes badly injured..

**OnMeetPlayer:** Send a message/impulse when the char comes close to the player.

**Alertness range:** The range within which the character is automatically turns to full Alertness.

**Collision rank:** This decides who will move out of the way when two or more characters meet. The one with the lower rank will move out of the way for the ones with higher. If the rank is set to zero the char will ignore the others collision rank, but the others will still try to avoid the char. Negative numbers will automatically set the char to ignore ranks.

**Sight range:** Specifies how long this char can see. This will be reduced by peripheral vision, low alertness, own movement etc.

**Hearing range:** Specifies how long the char can hear. This will be reduced by own movement and gunfire.

**Field of view:** Specify the characters field of view in degrees. About twice this is the characters peripheral vision.

**Awareness:** Magical range within which the bot spot all other characters

**Patience:** How long the character will keep doing things such as search is controlled by this value.

**Reaction time:** How fast the character reacts (obsolete)

**Clumsy:** The bot is clumsy so we give it more leeway when path finding.

**Flyatpathspeed:** Obsolete

**Unspotfast:** Obsolete

**Saveplayerinfo:** saves the relation to the player etc.

**Resetplayerinfo:** Obsolete

**Scriptmute:** tells the AI to be silent when scripted.

**Switchweapon:** Enclave stuff.



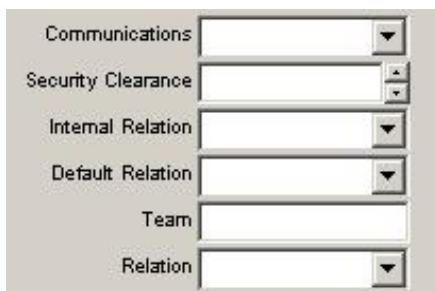
## Teams

This sets some of the rules for the different teams and relations between them.

When you create a char for you map you can select what team that char belong to.

Different teams may have different relations to the player and other teams. Some triggers might only trigger for a certain team and their members and some teams might have their own scene-points and so on.

To create a team go into the Create option menu by pressing “insert” or “<” and then select the class Team. A little blue ball will be created and this will symbolize the team. Don’t forget that you can use workspace to create layers and keep track of your teams and team members.



The image shows a vertical list of configuration options for a team. Each option has a text label on the left and a corresponding input field on the right. The input fields are mostly dropdown menus, indicated by small downward-pointing arrows. The options are: Communications, Security Clearance, Internal Relation, Default Relation, Team, and Relation.

Communications	<input type="text"/>
Security Clearance	<input type="text"/>
Internal Relation	<input type="text"/>
Default Relation	<input type="text"/>
Team	<input type="text"/>
Relation	<input type="text"/>

**Communications:** None, Voice, Direct.

**Security Clearance:** This sets the security clearance on the teams this can define what equipment they can carry without others teams react and so on.

**Internal Relation:** This sets the internal relation that the team has, if they are friends that won’t hurt each other or supposed to be able to fight with each other even start as enemies.

**Default Relation:** This sets the default relation to undefined.

**Team:** This creates a new team relation. Give the team a name and set the relation to that team below. A new Team and Relation will be created so that you can create multiple ones.

**Relation:** This sets the relation to the Team above.

These are the different relations that the teams can have. You can change these for teams or certain chars with impulses throughout the level with impulses.

**Friendly:** These won’t hurt you no matter what.

**Neutral:** Mainly ignores you.

**Unfriendly:** Watchful.

**Hostile:** Threaten.

**Enemy:** Shot on sight.

**NOTE:** To set the relation to the player you need to create a new team called Team\_Player in the empty field and then set the relation.

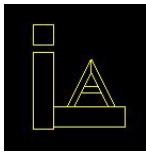
## Scenepoints

Scenepoints are used to guide the AI. The AI chars will roam, attack and a lot of other things without these guidelines. The scenepoints will help you to control the AI and it will help you to set up the game-play as you want it.

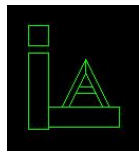
### Short description of how the scenepoints work

The AI uses, among other things, the arc from the scenepoint to determine what scene points to select. The arc doesn't define the view of the AI when reaching the scenepoint it defines what scenepoints to take depending on where its target is and what its doing. The AI will search for scenepoints that it thinks is the best and move towards it.

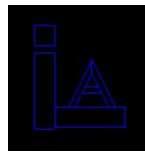
Once the AI has decided to move towards a certain scenepoint it will "take" the scenepoint and the scenepoint will declare that it has been taken and activate the "OnTake". You can set the maximum numbers of users for a scenepoint so that more then one AI can take the same scenepoint, just make sure that there is room for them. Once the AI reaches the scenepoint it will activate the scenepoint and the "OnActivate" will be activated. When the AI leaves the scenepoint it will trigger the "OnRelease". The scenepoint will then declare that it's open again, if the numbers of users was reached.



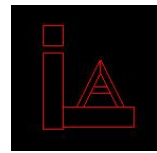
**Cover** (Yellow)



**Roam** (Green)



**Search** (Blue)



**Tactical** (Red)

These are the icons used in Ogier for the different scenepoints.  
(Yeah the icons are a little messy but you will get used to them)

### Scenepoint descriptions

**Cover:** Cover scenepoint. While taking fire from a target inside its arc the AI will search and go towards the scenepoints "radius" for cover.

**Roam:** While not engaged in fighting or searching the AI can use these scenepoints to roam to and from. The AI will use these unless there are some other engine paths that they are set to roam to or running certain behaviors.

**Search:** Search scenepoints are used when the AI is in search mode. Place the points there you want the AI to search when it thinks the target is inside the defined arc. The AI always has its sight and hearing but this give the AI the command to stop and search at a certain spot. It's good to angle the search cone up or down so that the char actually use its look upward or downward animations if you want it to look up or down. You can also tell the AI to turn on its flashlight at these points and so on.

**Tactical:** These are the points that the AI will search for while engaged in ranged combat. The selection is done depending on what scenepoint arcs the target is inside. If a char gets to close to the AI it will engage in melee combat.

## Scenepoint settings

Setting	Value
Wait Spawn	<input type="checkbox"/>
Heading Only	<input type="checkbox"/>
Crouch	<input type="checkbox"/>
User	
Team	
Radius	32
Arc halfangle	30
Arc Range min	96
Arc Range max	600
Num Users	1
Duration	3
Show Secondary SPs	<input type="checkbox"/>
OnActivate	
Activation Delay	
OnTake	
OnRelease	
Behaviour Type	General
General B.	

Setting	Value
Wait Spawn	<input type="checkbox"/>
Heading Only	<input type="checkbox"/>
Crouch	<input type="checkbox"/>
User	
Team	
Radius	32
Arc halfangle	45
Num Users	1
Duration	3
Show Secondary SPs	<input type="checkbox"/>
OnActivate	
Activation Delay	
OnTake	
OnRelease	
Behaviour Type	General
General B.	

**WaitSpawn:** The scenepoint won't spawn until it's told to.

**HeadingOnly:** This will remove the vertical limit for the take of the scenepoint. It's still limited in the horizontal arc.

**Crouch:** This will order the AI to be crouched while activating this scenepoint.

**User:** Specifies users for this scenepoint. You can add multiple users to a scenepoint as a new user field will be added once the first one is used.

**Team:** Specifies a team for the scenepoint. You can add multiple teams to a scenepoint as a new team field will be added once the first one is used.

**Radius:** This sets the radius of the area that the AI can stand inside on activation of the scenepoint. Make sure that this isn't inside a wall or the area too small for the AI to stand inside. The normal AI base size is 24x24units. The AI doesn't roam towards the middle of the scenepoint and can activate the scenepoint at the edge of the circle.

**Arc halfangle:** Use this to change the cone that the AI use when selecting scenepoints. The arc halfangle affects both the vertical and horizontal even though it doesn't display the vertical one. If you want the AI to be able to choose a scenepoint even though a target is way above or below the arc check the heading only

**Arc Range min:** Defines the range of the arc.

**Arc Range max:** Defines the range of the arc.

**Num Users:** This sets the maximum number of users that can “take” the scenepoint.

**Duration:** The duration defines how long the AI will stay at the scenepoint before it starts to look after another one.

**OnActivate:** When the AI reaches the scenepoint it activates the scenepoint, you can send messages/impulses when this happens. New “OnActivate” fields will appear once the first one is used, so you can send multiple impulses from the scenepoint when this happen.

**Activation Delay:** -

**OnTake:** “OnTake” happens when an AI decides to “take” the scenepoint. You can send messages/impulses from the scenepoint when this happens. New “OnTake” fields will appear once the first one is used, so you can send multiple impulses from the scenepoint when this happen.

**OnRelease:** Use this to send messages/impulses when the AI leaves the scenepoint. New “OnRelease” fields will appear once the first one is used, so you can send multiple impulses from the scenepoint when this happens.

**Behavior Type / General B:** You can use this if you want the AI to behave in a certain way when it reaches the scenepoint. You can also send impulses from the scenepoint to the AI in the different stages during the scenepoint.

## Impulses/AIimpulse

There are a lot of impulses and messages that can be sent to the chars you will find some of these under AIimpulse. Some of them are tricky and some are kind of self explained, toy around with them.



## Impulses

### Generic

**AIimpulse**, this will bring up a submenu with the AIimpulses.

In the sub menu you can find most of the impulses that you controll the Ais with.

**CutsceneFov**, the cameras field of view can be set with this impulse.

**Destroy**, removes objects from the world.

**DisableQuicksave**, you can disable/enable the quicksave function with this impulse.

**GetDistance**, checks the distance between the object and the target.

**GetRandom**, gets a random number, the range is set manually.

**Impulse**, used to send impulses.

**Param\_Add**, param add.

**Param\_ClearFlag** clears a specified bit on Param

**Param\_CopyFrom**, copy the param from an object to another.

**Param\_Get**, get/check the param of an object.

**Param\_GetFlag**, returns true if specified bit is set on Param

**Param\_Randomize**, this sets a random param on an object.

**Param\_Set**, this sets a specific param on an object.

**Param\_SetFlag**, sets a specified bit on Param

**PlaySound**, use this to play a sound with an impuse.

**SetDescName**, this is used when you want to change the description on a ACS etc.

**SetFocusFrameOffset**, change the focus frame offset on an object.

**SetDialogueProxy**, place a specified dialogue proxy for a character etc, the point specifies with a DialogueProxy object.

**SetModel**, with this you can change the model on a object or character, changing the head model of a char etc.

**SetModelFlags**, this controllles particle effects by turning on and of their emission stop.

**SetSound**, this sets a sound on a entity that will loop.

**SetName**, you can set the ACS username.

**Spawn**, this will spawn a waitspawned object.

**Teleport**, with this impulse you can move/teleport a object to a different location.

## **Game**

**ActivateCheckpoint**, activate the checkpoint and save the game.

**ChangeWorld**, this will change map/level

\$<map name>:<info player start name>

**ConExecute**, executes a console command.

**CompleteMission**, Ends a given mission.

**DebugMsg** writes specified string to the console

**EndCutscene**, this will end the cutscene mode.

**FadeScreen**, use this to fade the screen to a specified color. You can also fade from a set color by using a negative number.

**FadeSound**, fade a specified sound.

**FailMission**, mission failed.

**LevelExists** returns true if a specific .XW file exists.

**Play2Dsound**, this will play a 2D sound.

**RailSpawn**, this will create a rail wagon at a position.

**RailNumByType**, checks the number of wagons there is in the map of a certain type.

**RailNumByName**, checks how many wagons there are in the map with a specific name.

**RailPower**, this turns the power on/off for the rail.

**RailEnemy**, use this to set the railwagons relation to enemy to a target.

**RailUnfriendly**, use this to set the railwagons relation to unfriendly for a target.

**RailAttack**, this will be a scripted attack towards an entity or position.

**RemoveGameMsg** will remove a game message in the queue.

**SPSpawn** spawns a scenepoint.

**SPUnSpawn**, unspawn a scenepoint.

**SPRaisePrio** is none scripted raised prio for a scenepoint.

**SPRestorePrio** will restore the prio of a scenepoint.

**SPActivate** activates a scenepoint by a impulse.

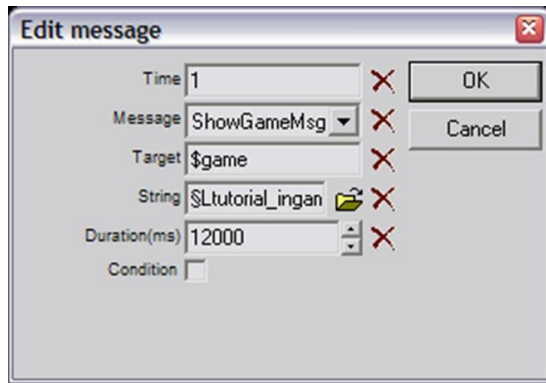
**SetMission** starts a mission.

**SetMusic**, use this to set or change the music.

**SetNVRange**, this sets how far the player can see with the nightvision.

**ShowGameMsg** will display a message on the screen, if the duration is set to 0 the message will be shown till the player manually removes it. The messages are set in the stringtable.

Message:



**ShowInfoScreen** shows a bitmap, this bitmap has to be in one of the texturecontainers and precashed in the level keys.

**Quit**, quits to the menu or quits the game.

**SetGamestyle**, this will set the different game types in Riddick.

Sneak/Action will affect the AI in different ways suited for the game type.

## Character

**Restriction** sets a restriction on a character.

**AddItem** gives a character an object.

**AddHealth** gives the character health.

**ActivateItem**, force-activates the specified item

**DestroyItem**, destroys the specified item

**Disable**, disables certain functions for the character.

**DragDoll** connects a corpse to an object.

**EquipItem** forces the target to equip a certain item.

**GetHealth** will check the health on a target.

**GetNumItems** checks the number of items of a certain kind that a character has.

**Immobile** makes the target immobile.

**Immune** makes the target immune, but the character will still display his hurt animations.

**IncreaseMaxHealth**, increase the max health.

**NeverTrigger** sets the target so that it can't trigger anything.

**PerformAnimAction** will play an animation on the target.

**PlayAnim** plays a animation on a target.

**PlayAnimAnimModel**, play animanim model (old stuff)

**Push** will push the target in a specified objects direction.

**RaiseNoiseLevel** will be removed

**RaiseVisibility** will be removed

**RenderAttached** attaches a object to an other.

**SetApproachItem** sets a specified dialogue approach string.

Ex

-100 will set the AI to say string 100 when approached.

100 will set the AI so that when a character aproach it the character will say its sting 100.

**SetAimingMode** changes the aiming mode on the character.

**SetDropItem** set what item a character will drop when it dies.

**ShowExtraItem** this will add another object in the hand etc on a character, will be attached to another attachpoint.

**SetFlashLight** turns the flashlight on/off.

**SetLaserBeam** turns the laserbeam on/off.

**SetNightVision** sets night vision on/off.

**SetPhysFlags** sets phys flags on a target.



**SetSpecialGrab** will set the sneak neckbreaking in riddick to another special event, if you want the char to be pushed out over a ledge or into a grinder etc.

**Speak**, makes the character to talk from its dialogue file.

**Stun** will stun the target.

**Team\_Alarm**, alarms a team.

## Player

**AI Control**, Toggle AI (script) control on player. Only used in Enclave

**BeginCutscene** starts a cutscene.

**BeginDialogue** begins a dialogue.

**Mount**, attaches a camera to an AI so that the player cam will display what it sees.

**Rumble**, joypad rumble.

**SetZFog** will change the zfog parameters.

**SetCameraEffects** effects test that is not beeing used.

**ShakeCamera**, camera shake.

**UseDNAWeapons** toggles the ability to use dnaencoded weapons.

## Engine/Hook

**GetSequence**, returns the currently active sequence on an engine\_path

**WaitImpulse** sets a object to wait till it gets a certain impulse.

**WaitParam**, the object will wait for a certain param.

## ReinforcementController

**RFCDelay**, reinforcecontrollDelay is used to set the delay from that a character has died and untill a new one spawns.

**RFCReset**, reinforcementcontrollReset will reset the whole reinforce system.



## Light

**LightImpulse** can send different kinds of impulses to a lamp like on/off and tell it to break etc.

**LightMorph** animates a lightsource setting over time.

**SetProjMap** is used to change the projection map on a lamp.

## AI impulses

### Misc

**Run behaviour**, run behaviours.

**Force behaviour**, force behaviours.

**Stop behaviour**, stop a current behaviour.

The normal option will create a smooth transition.

The fast option will stop the behaviour in the middle of it etc.

**Push** not in use, will be removed.

**Aggressive**, makes an AI to shoot at enemies if it's possible.

**Aim** tells an AI to aim at an object, scripted.

**Attack** tells an AI to attack an object, scripted.

**Continuous Look** makes the AI to look at an object all the time, will turn his body to keep doing so.

**Cont. Soft Look** will make the AI to look at an object if it's in its FOV.

**Crouch**, crouch on/off.

**Die**, kills the target.

**Damage** will hurt the target.

**DragDoll** use this to tell an AI to drag a ragdoll, not recommended that you use this one.

**Next of kin**, characters with the InvestigateDeath action will only investigate corpses that have NextOfKin flag set.

**Ghost Mode** turns the collision on/off for a AI.

**Fly Follow Me** tells an AI to fly after an engine path, scripted.

**Follow Me** tells an AI to follow an engine path, scripted.

**Force FM Relative**, this will force the AI to follow a path relative to the position that it has when he gets the message, scripted.

**Force Follow Me**, force an AI to go directly towards an EP will follow the paths position, scripted.

**Look at object** makes the char look at an object, turned off with 0.

**Move to object**, tell an AI to go to a specified object, scripted.

**Pause**, this will pause the AI.

**Release**, this will release the AI from all its scripts.

**Speak**, not in use and will be removed.

**Spot Character** tells an AI where a character is.

**Notice player at**, give the AI information about where it will think the player might be.

**Enemy relation** sets an enemy relation with the target.

**Friend relation** sets a friendly relation to the target.

**More hostile** makes the AI more hostile towards the target. This can be used to simulate patience, or the lack of it.

**Default relation** will reset the relation with the target.

**Switch Weapon**, only used in Enclave.

**Switch Item**, only used in Enclave.

**Teleport Follow Me**, teleports an object to a target.

**Unspot All**, this will clear the AI's awareness of targets.

**Use Weapon**, fire or use the current weapon, was only used in Enclave.

**Use Item**, Enclave use item like a potion etc carried in the hands.

## Default Action

Some of these appear in Enclave, KotT and Riddick and some don't.

**Avoid** move away from player when seen.

**Engage** seek and engage a specific target default player.

**Escape** not in use.

**Explore** lets the AI use roam sceneponits.

**Follow**, AI follows a specific target.

**Guard** not in use.

**Hold**, default action for an AI. AI stands in spawn point or at a specified target when idle.

**Patrol** tells the AI to patrol a specific path.

## Action modifier

**Move to ScenePoint**, forces the AI to move to a scenepoint, scripted.

**ScenePointOverride** tells the AI what scenepoint it should move to if it selects a scenepoint of that type. Not scripted.

## Attributes

**Activation Range** sets the activation range of the AI, if the player moves outside this range the AI will be paused.

**Alertness** sets idle response settings.

**Awareness** is a magical radius where the AI will notice the player automaticly.

**FOV** sets the Field of View of the AI in degrees.

**Health** sets the amount of hitpoints the AI has.

**Hearing Range** sets the range in units that the AI can hear.

**Idle Speed** how fast the AI moves when idle.

**Reaction Time** sets the response time between different actions.

**Reasoning** old stuff used in Enclave, sets how fast the AI notice stuff etc.

**Sight Range** sets the sight range of the AI, in units.

**Skill** sets the skill of the AI's actions, like firing a bow etc. (Enclave)

**Security** sets the security level of an AI.

**Team**, gives the target membership of a team.

## **Restriction**

**Active** the restriction is active.

**Range** the range of the restriction in units.

**Object** restricted to a specific object, default AI spawn position.

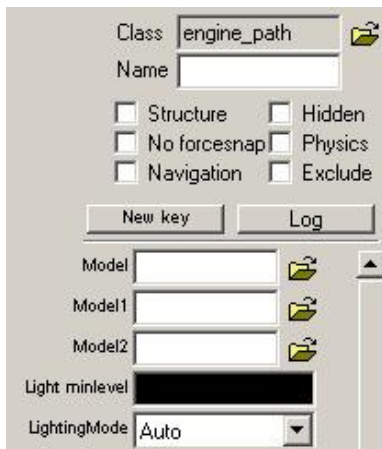
## **Special**

**Special** please don't pay any attention to this it's too special.

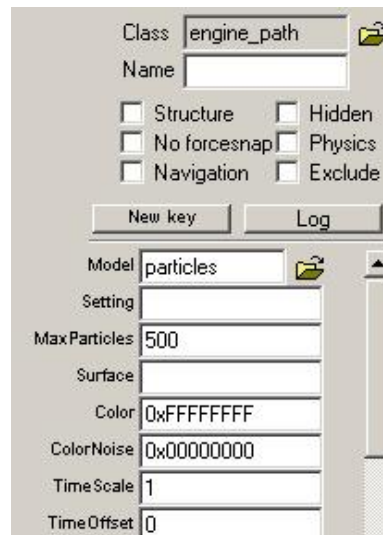
## Particles

I am not sure I can tell you how to use the particle system and how to operate it. This is one of those thing there you have to tweak and try yourself. I will try to explain what the different settings do but the rest is up to you.

To create particle effects in Ogier you have to type the word “particles” in the model field. I recommend using engine paths for the particle effects, but you can use other objects like models. Engine paths can be animated and it has room for 3 particle strings in the same object. I will use engine paths in my examples.



(Engine path)



(Engine path with particles)

This is how an engine path looks like when you activate the particle system.

(Some of the options you will encounter while working with the particle system)

## The particle string

This system will probably be replaced but this is what we have used up till Riddick. The particle strings that will be created after the “particles” can only have a maximum of 250 numbers. It has a few other flaws besides the maximum of number it can have. It has a tendency to add lots of “,” into the string.

Ex

“particles:,MP=350,SU=p\_smokeblob01b,CO=0x22FFFFFF,,DU=2,,FI=0.5,,VE=5,,AX=0 0 10,,DI=sphere,DIS=4 4 2,,,,,,,,,SZ0=5,,,,,,,,,,,,,,,,,LS=25,FL=nl+al+asq”

The extra “,” can be removed. There isn’t supposed to be more then one “,” in a row, the others can be removed.

Ex

“particles:,MP=350,SU=p\_smokeblob01b,CO=0x22FFFFFF,DU=2,FI=0.5,VE=5,AX=0 0 10,DI=sphere,DIS=4 4 2,SZ0=5,LS=25,FL=nl+al+asq”

## Settings

**Model:** particles

**Setting:** Used for presets, but there are non available at the moment.

**MaxParticles:** This decides the maximum number of particles that can be shown at the same time. If the MaxParticles number is to low the particles can be cut off depending on the duration and TimecellDuration.

**Surface:** Specify what surface that should be used for the particles. “#” can be used to sort surfaces. Ex “p\_smokeblob01#2” Note: (fade wont work with surfaces that only have the “Add” parameter, but it works with “AlphaAdd”)

**Color:** This is divided so that the first two numbers are the alpha fade and the other six are the color tint. The numbers are done in hex or decimals.

Hex: 0xFFFFFFFF – FF (alpha), FF (Red), FF (Green) and FF (Blue).

Dec: "RRR GGG BBB" with a value between 0-255.

**ColorNoise:** Randomize the above alpha and color values.

**TimeScale:** This changes the timescale for the particles.

Ex: 0.1 = 10% of the original speed.

**TimeOffset:** Change the start of the particles. A negative value equals a delay. A positive value equals a start in the middle of the simulation.

**Duration:** This decides the duration of the particles.

**DurationNoise:** Randomize the duration of the particles.

Ex. Duration=10 with a DurationNoise=5. This would mean that the particles would have duration of 7.5 to 12.5 seconds.

**FadeInTime:** FadeInTime creates a time for the particles to fade in.

**AlphaStart:** Set the alpha value of the particles when they are created.

**AlphaEnd:** Set the alpha value at the end of the duration for the particles.

**Velocity:** This sets the initial speed of the particles.

**VelocityNoise:** Randomizes the velocity.

**LocalOffset:** x, y, z

**AccelerationVector:** x, y, z as taught in physics, power is nothing without control.

Define here where your particles should fly to. You can have negative values.

**AccelerationVectorNoise:** x, y, z

**Distribution:** Different ways to distribute the particles. The different ways are: point, cube, sphere, cylinder and tube (Pyramid and Tetraid might be available aswell but no guarantees).

**DistributionSize:** x, y, z

**DistributionRotation/Time:** x, y, z rotation of the distrubution primitive Ex "0 0 1" equals one revulution in the Z-axies per second.

**HollowDistribution:** 0 to emit from the whole volume and 1 to emit from the surface.

**PositionOffset:** Defines how far from the the the original emit position a particle shall be emitted.

**PositionOffsetNoise:** Randomizes the position offset.

**Move/Rot/Size Controllers:**

When a particle is created it will get a random value between the controllers StartMin and StartMax. When the particle dies the value of the particle will be on the same level but between EndMin and EndMax.

If FluctSpeed is set to 0 the value will interpolate lineary between the start-value and the end-value. If FluctSpeed is greater then 0 the value will fluctuate between Min and Max

**Move:** Controls the particles movement around its actual position.

**Rot:** Controls how a particle rotates around its own depth axis.

**Size:** Controls the size of the particle.

**MoveStartMin:** Minimal movement at start.

**MoveStartMax:** Maximum movement at start.

**MoveEndMin:** Minimal movement at end.

**MoveEndMax:** Maximum movement at end.

**MoveFluctSpeed:** Move fluctuate speed.

**SizeStartMin:** Minimal size of the particles at the start of the simulation.

**SizeStartMax:** Max size of the particles at the end of the simulation.

**SizeEndMin:** Minimal size of the particles at the end of the simulation.

**SizeEndMax:** Max size of the particles at the end of the simulation.

**SizeFluctSpeed:** Randomize the size while the particle travels around (lets the particle pulse).

**RotStartMin:** Particle rotation.

**RotStartMax:** Particle rotation.

**RotEndMin:** Particle rotation.

**RotEndMax:** Particle rotation.

**RotFluctSpeed:** Randomize the Rotation while the particle travels.

**AngleSpreadA:** Horizontal spread from the emitting direction, 1 is a whole rotation.

**AngleSpreadB:** Vertical spread from the emitting direction, 1 is a whole rotation.

**AngleOffsetA:** Horizontal angle for the emitting direction, 1 is a whole rotation.

**AngleOffsetB:** Vertical angle for the emitting direction, 1 is a whole rotation.

**AngleOffsetChangeA:** Horizontal angleoffset/rotation for the emitting direction.

**AngleOffsetChangeB:** Vertical angleoffset/rotation for the emitting direction.

#### **Emission Time Controll:**

Time in the particle system we use is devided into timecells, during these timecells a number of particles will be emitted with a certain probability. The timecells duration and the particles spread within a timecell can be used to create emission puffs. If a timecell is 1 second long and the timecells spread is 0.5 (50%) it will only emit particles under the first 0.5 seconds of each timecell.

**TimecellDuration:** The duration of the timecell.

**TimecellSpread:** Emission spread during a timecell, values between 0-1 to create 0-100%.

**ParticlesPerTimecell:** This sets the number of particles that the emitter will try to emit during each timecell.

**EmissionProbability:** The probability that a particle emits, set a value between 0-1. This can cut down non-continuous streams.

**EmissionProbChange:** This will probably randomize the EmissionProbability. Try around and find out because my brain is on a strike atm.

**EmissionStop:** Sets the duration of the particle creation. If you set emission stop to 5 the particle effect will run for 5sec etc. The ones already created will continue to live. If you don't want an emissionstop the time is set to 0.

**StopMotionTime:** This stops the particles velocity motion after a certain time.

Acceleration values still affect the particle. This needs the SM flag in the flag field.

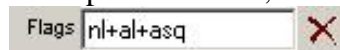
**SlowdownPower:** Exponential force that slows down the particles the longer they have lived.

**FadeStillThreshold:** Sets the velocity for when a particle is concerned to stand still and begins to fade away.

**AlignLengthScale:** How long the particle can be stretched before movement takes over.

## Flags

Some settings require that you set some flags. The flags are separated with "+". The whole name of the flag can be typed into the field but the abbreviation is normally used to save space and time, so use those ones.



**NL (nolod):** Turn off loading. The particle system won't get fewer particles when further away from the camera.

**AL (align):** Align particle rotation with its current movement vector. Particle will stretch out. Aligned in a specific angle

**FS (fadestill):** Fade when model reaches a still position. This will fade the particles that move slower than the FadeStillThreshold.

**SD (slowdown):** Slow down particle velocity to zero at full duration. This will slow down particles with a set force set by SlowdownPower.

**NH (nohistory):** This will ignore where the emitting object has moved and only use the last position for the particles.

**QD (quads):** This will render each particle using 2 triangles, as a quad, instead of one clamped triangle.

**SM (stopmotion):** is a better version of slowdown. Particles change velocity and stop after a certain time.

**PC (primcenter):** will force the particle velocity center to primitive/model center.

**WR (worldrot):** will force the particle velocity center to primitive/model center.

**HF (hflags):** Emit particles only when history flags are != 0.

**SP (spline):** Spline interpolation when calculating intermediate history matrices.

**CT (ctime):** Continuous time, Use client gametick time instead of enginepath controlled time.

**FE (erl):** Spawn particles as early as the latest tick + iptime, ignore interpolation misses.

**IHF (ihflags):** Emit particles only when history flags are == 0.

**LA (localaxx):** Apply acceleration in local space instead of in worldspace.

**NCC (noclampcolors):** clamp particle color components to 0-255.

**SOF (showoverflow):** Use default texture when exceeding maxparticles limit.



**NCCN (nochanneledcolornoise):** Don't randomize color channels individually.

**ETS (ets):** Use emitter time fraction for size interpolations.

**ASQ:** Align square, Quad aligned particles with movement aligned to CameraForward (to avoid thin particles).

**AE (ae):** Consider emitter movement for alignment.

**SMO (smo):** Will enable the old stopmotion formula, not recommended.

**Clampcolors:** Unknown.

**Fadepoles:** Unknown.

## Scale **TEMP**

1m = 32units

-----

Size

AI base size =24x24units

-----

Tools

-----

ctrl-alt-rightmousebutton

--

If you haven't got the console enabled just put 'CON\_ENABLE=1' onto the end of your environment.cfg.

--

Origo / Home

Home= 0.0.0

## Using the AI (Old Enclave document)

*Note: I am just going to copy and paste an old document here that I found from the time Enclave was done. The “Using the AI in the SBZ engine” part was not written by me so I cant take any credit for it, nor blame if something is terrible wrong. A lot of the things in the following text might not be working as it should (or not at all) as Ogier has changed a lot since it was written. I haven’t had the time to test all of it but perhaps it can help you figuring out some of the features in Ogier. I will go back to this document later on and go though it, but for now I will leave it as it is. Enjoy.*

*This part is mainly for the Enclave version of the Starbreeze Engine, but a lot of stuff has been kept for KotT and Riddick, so it could provide information on the AI there as well.*

- This document describes how to configure bots and any special scripts that govern them using the Ogier tool. A working knowledge of the Ogier tool is assumed, especially how to handle triggers and engine paths. These features will be described in more detail, regarding how they can be used for scripting control below as well. Note that the Starbreeze engine is a work in progress, and some of this information may become deprecated or incomplete.
- A note in reading this document: the actual bread text, usually coming right after the various headings are general stuff which should be read by anyone reading this document. The bullet lists contains specific info of various scripting features, and can be skimmed when reading this for the first time, and should really be viewed as reference material when actually performing scripting.
- To make bots behave the way you want to there are two general approaches. When one wants the bot to behave in a fairly independent manner one can just set it’s behaviour, and then let it handle itself. When one wants the bot to do some more specific tasks, one will have to construct scripts that explicitly command the bot to do something at a certain time or under certain circumstances. The distinction in control is quite significant; if you give the bot certain messages it will only do what you tell it to and nothing else, while certain others will only change it’s behaviour temporarily or let it decide everything by itself.
- I’ll describe the details of scripting first, as a bit of scripting can be useful when restraining independent bots, so even if you don’t have to know all about how to make detailed scripts, some basic stuff will probably come in handy, especially how triggers and engine paths work when used in conjunction with scripting.
- Finally, when constructing scripted behaviour, one must always remember that *“There’s more than one way to do it!”*

## Scripting

- The first skill to master when creating bot scripting (or any scripted events at all for that matter) is to make sure the scripts trigger at the right time. There are three types of “control structures” that you can use for this purpose: triggers, engine paths and bot events. These are described below. Also, independent bots may of

course decide to do something on their own, which may cause unwanted events if not properly restrained.

- The command structures exert control over the game by sending messages to other objects. Specifying these messages are the second skill that you'll need, and this is described below as well.

## Triggers

- A trigger can be used for starting events when someone is at a certain position. The Ogier class *trigger* is deprecated, but instead there's is the Ogier class *trigger\_ext*, which has significantly more functionality. It is triggered when an object with the specified properties (default is all characters) is inside the trigger and will normally continually trigger while this is true. It will also be triggered if receiving an impulse message. When triggered it normally sends all it's messages, with the triggering object as sender (see \$activator below).
- You should be familiar with the following keys of the *trigger\_ext*:
- The **Name** is the target name of the trigger, to be used by other objects that need to know of the trigger, for example if they want to send a message to it.
- The **Intersect Objects** key specifies the target names of any objects capable of triggering the trigger. I.e. if the trigger has intersect objects "Marcus" and the character flag set, then only characters named Marcus can trigger it.
- The **Parent** key is the target name of any parent object from which to inherit position.
- The **Visible** and **Game Physics** flags are not implemented as yet.
- If the **Once** flag is set, the trigger is automatically destroyed after sending all messages (waiting for delay if there is one).
- If the **Random** flag is set, the trigger will only send one of it's messages, randomly selected.
- If the **Wait Spawn** flag is set the trigger won't "exist" until it receives a spawn message.
- If the **Destroyable** flag is set the trigger can react to receiving damage. This is handled by specifying a bunch of other keys, which will be described below.
- The **Message** keys are the messages to be sent when triggered.
- The **Delay** value specifies the minimum time (in seconds) between each possible triggering. When first triggered it will send normal messages in the usual fashion. If this value is positive then any delayed messages will be sent after this time in seconds, and then the trigger can be triggered again.
- The **Delayed Message** keys are the messages to be sent after the delay, if that value is positive.
- The **Player**, **Character** and **Projectile** flags is used to control which objects will set off the trigger. The character flag is the default, and will set off the trigger when any character (bot or player) moves into the trigger. Thus, if the trigger should only react to players, the character flag must be disabled and the player flag set.

- The **Use** flag will trigger only if a player in the trigger actively uses it, i.e. hits the use button.

The below keys only apply to triggers with the Destroyable flag set.

- The **Hitpoints** key specified how much damage the trigger can sustain. When the hitpoints are reduced to zero, the trigger sends all messages (waiting for any delay) and is then destroyed.
- If none of the damage type flags (**Fire**, **Cold**, etc) are set, the trigger is susceptible to all kinds of damage, otherwise it's only susceptible to those types that are set. Thus you can create a door that can only be burnt down by setting the fire flag, for example.
- The **Mindamage** key specifies the minimum damage needed to damage the trigger at all. A trigger with mindamage 10 will take no damage from an attack with 9 damage and 10 damage from one with 10 damage.
- The **Autotargetname** key is the string that will show when aiming at the trigger in game if using the Enclave auto targeting system.
- The **Linkedtriggers** key is a semi-colon-separated list of the target names of other triggers to be triggered when this trigger is.
- The **Maxdamage**, **Maxdamagetype** and **Damage Message** keys are unreliably implemented. Check with Jesse if this is a problem.

## Engine Paths

- The Ogier class *engine\_path* is used for two tasks: it measures time, and can keep track of positions. It can also send arbitrary messages. Thus it can be used to trigger events at certain times and/or when it's reached certain positions. Engine paths can be stopped and started by impulses, where a impulse 0 will stop the path, an impulse 1 will start the engine path (or cause it to resume the sequence where it last was stopped) on it's first sequence, an impulse 2 will start/resume the second sequence, etc. Note that the time of an engine path is one-to-one mapped with the position; if the engine path is stopped, the time is stopped and if it's reversed or restarts the time reverses/restarts. There's also a simplified version of this class, the *engine\_script*, which only keeps track of time.
- You'll need to know how to configure the following keys:
- **Name**. Own target name.
- If the **Model** key is set to something, a model will be rendered at the paths position. If the *particles* model is selected, you can specify a number of keys to create a particle system that will move along with the engine path. This can also be useful for testing purposes, to debug a script that behaves strangely and see if the engine path is moving correctly.
- The **Game Physics** flag **MUST** be set if the engine path is to be used to keep track of positions. Otherwise the path won't have a real position, and won't actually move. To forget this is a very common mistake.
- The **Auto Start** flag will cause the engine path to start automatically when the game begins. Note, however, that the game, and thus the path, may start *before* the player is spawned.

- The **Auto Reset** flag will cause the engine path to restart if it has reached it's end and receives another impulse (with param 1+). This is useful when you want to be able to trigger a script multiple times.
- The **Loop** flag will cause the engine path to restart from the beginning when it reaches it's end. This will occur instantaneously, so it's a good idea to make sure a looping engine path end where it begins.
- The **Auto Reverse** flag will cause the engine path to reverse its current sequence if it collides with anything (useful for doors or elevators). Don't use this flag for engine paths that sends scripts, or at least don't blame me if you do ☺.
- The **Pushing** flag will cause the engine path to send force to other object when colliding them, thus pushing objects out of it's way instead of just going through them.
- The **Unreliable** and **Hand Driven** flag are unreliable.
- If the **Never Interrupt** flag is set, the engine path will always complete it's current sequence even if it receives an impulse 0. Thus, a looping path with this flag that receives an impulse 0 will continue to run until the sequence is complete and then stop.
- The **Invis When Stopped** flag will cause any model of the path to stop rendering when the path reaches the end of its sequence.
- The **BlockNav** flag will write the engine path into the path finding navigation grid, thus enabling bots to take it's movement into account. This will only occur once the sequence has reached it's end however, so there may be a noticeable delay before bots can use the information (the won't try to go through a half-open gate for example). Note that normal doors usually aren't written into the navgrid as these can be opened by the bots themselves and this happens too frequently to be effective.
- The **Wait Spawn** flag is set the engine path won't "exist" until it receives a spawn message.
- The **Attach** key is used to attach the engine path to another object in a more flexible fashion than with parenting. The value should be a string of the format <x-offset>,<y-offset,<z-offset>,<target name>.
- The **Type** decides how the engine path positions are interpolated between key frames. Linear paths give abrupt corners, so splines are usually preferable when making movement paths.
- The **Sequence** and **Time** fields, as well as the buttons directly below these are used to create or modify key frames and sequences. You should already be familiar with this. Note that the time you enter when hitting the "new" button is absolute time, not time relative to the last key frame, so you could insert new key frames between existing ones if you want. In contrast, the "insert-" and "delete time" buttons allow you to add/reduce time until the next key frame. Also note that a server frame is currently 0.05 seconds, so you could squeeze in several key frames into one server frame, in which case any messages sent during these key frames would effectively be sent at the same time.
- The **Timedsound** key is deprecated.
- The **Timed Message** keys are the messages the engine path sends when reaching the specified time/position. These messages have an extra time field, which is the

sequence time when it should be sent. Note that this is the effective run-time of the engine path, not absolute time, so if sequence 1 runs for two seconds and is then stopped by an impulse 0 it won't send a message with time 2.2 until the path has received an impulse 1 and run for another 0.2 seconds. Unfortunately, messages aren't sequence-specific, so the above message would be sent if sequence 2 was allowed to run for 2.2 seconds as well. If there are any messages after the last key frame, the engine path will continue running until the last message, although it won't move etc. This may cause unforeseen side effects, so it's usually recommended to have a key frame with time equal to or later than the latest message.

- The **Jump** key is a string with a time followed by a comma followed by a target name (eg. "0.5,Camera02"). This is used when linking several engine path-cameras of a cut scene together, where this engine path (camera) gives camera control over to the engine path specified by the target name at the specified time.

## Bot Events

- In addition to triggers which can be used to determine when something is in a specific area, and engine paths, which can keep track of time, one can let the AI itself determine when a certain situation has occurred. The AI will then check for when the event occurs, and raise it when it does, sending any message(s) specified.
- Event messages are specified with some keys in the bot itself:
- The **OnEnemySpotted** event is raised whenever the bot spots a previously non-spotted enemy. The bot will only spot an enemy if he sees him clearly or comes within awareness radius. Thus, he can be shot by an enemy without automatically spotting him, although this raises his chance of doing so.
- The **OnInjured** event is raised whenever the bot is hit by something, regardless of whether he actually takes damage or not. It is not raised when hit by friendly fire, though.
- The **OnDie** event is raised when the bot dies.
- The **OnSpawn** event is raised when the bot spawns.
- In addition there are a bunch of non-implemented events. I'll implement these if the need arises, and if there are other useful events I could probably implement those as well.
- In addition there are some special event-related keys:
- The **Starting Impulse** is deprecated. Use the OnSpawn event instead.
- The **OnStart: Pause** flag causes the bot to start paused. This is useful for making sure scripted bots don't wander off on their own for some reason.
- The **OnStart: Immune** flag causes the bot to start immune to all damage effects.
- The **OnStart: Aggressive** flag causes the bot to start aggressive (i.e. scripted, and won't move unless told so, but will attack anything it thinks it can hit).
- The **OnEnemySpotted: Release** flag is a convenient shorthand for causing the bot to be released from script control when it spots an enemy.

- The **OnSpawn: Push** key gives the bot the given speed in the direction he's positioned in when spawning. Useful to have bots flying out of ambushes etc.

## Messages

- To get objects to react when you want them to, the control structures must send messages to them. A message in Ogier is a comma-separated string of a special format, but you should not need to bother about that directly, but instead be able to assemble a message by selecting stuff from combo-boxes and writing some appropriate strings.

A message has four parts:

- i. **Message Type.** You need to know about the following message types:
  - **Add Gold** adds the given amount of gold to the targets (character) pouch.
  - **Add Health** adds the given amount of health to the target (character).
  - **AI Control** will cause the target (player) to become controlled by the AI if the param is 1 or to become controlled normally otherwise. This should be used with care. The AI control functionality is rather defective with moving characters due to the asynchronous nature of players, but this will hopefully be fixed.
  - **AI Impulses** are used to give script orders to bots. The message parameter is then the type of AI impulse and the message string is any additional parameter to the AI Impulse. AI impulses are rather complex as they are used for all detailed scripting of bots, and will be described in more detail in the AI Impulses section below.
  - **Aircraft** causes the target (character) to start banking and pitching with movement if param is 1 and stops him doing so if 0.
  - **Begin Cutscene** messages are used to initiate in game cut scenes. The target name is the name of the player who should be shown the cut scene, usually \$activator.
  - **Destroy** will cause the target object to be destroyed at the end of the next game tick. This will trigger the OnDie event for bots.
  - **Destroy Item** causes the target (character) to lose the specified item (if he had it to begin with).
  - **End Cutscene** messages are used to cancel in game cut scenes. The target name should be the name of the player who should stop viewing a cut scene, usually \$activator.
  - **End Round** messages ends the game round, with the specified result and message. The target name should always be \$game.
  - **Fade Screen** causes all cameras to fade down to black if the duration specified is positive, or fade up from black if the duration is negative. The (absolute) duration is in milliseconds. The target name should always be \$game.
  - **Immune** will cause the target (character) to become immune to damage if the param is 1 or to become normally susceptible to damage otherwise. I.e. 1 = on, 0 = off.



- **Impulses** are used to stop (param 0) or start (param 1+) engine paths and other dynamic objects. Any impulse will also set off a trigger.
- **Play Anim** causes the target (a character) to play the given animation, starting at the given time into the animation.
- **Play Sound** plays the given sound at the targets position.
- **Rumble** triggers the hand control force-feedback vibrator for target (player) with given envelope name (?).
- **Set Fade** sets the degree of fading (0 = fully transparent, 255 = fully opaque) of the target (character). Ask Jesse.
- **Set Model** changes the targets primary model to the given one. If 0 is given, the targets primary model will disappear.
- **Set Model Flags** changes the targets model flags (currently particle system on/off). Ask David for more info.
- **Set NPC Bar** gives the target (player) a health bar display of the given NPC. Useful for escorting missions.
- **Set Sound** attaches a looping sound to the target.
- **Set State** changes state of the target (message pipe).
- **Shake Camera** causes the targets (player) camera to shake. The parameter is a comma-separated string of the format <amplitude>,<duration>,<speed>.
- **Spawn** messages are used to spawn an object that is waiting for spawn (i.e. has the “wait spawn” flag set).
- **Speak** messages causes the bot to say something. The target name is the name of the character who should say something. The parameter is the index number to the specific line in the dialogue-resource that belongs to the bot. This will also cause any appropriate sub titles to appear.
- **Special Award** gives the player the specified amount of gold or something... The target name should always be \$game. (Uncertain info, ask Jesse)
- **Switch Character** will cause the target (player) to switch character object to the given one (effectively switching body).
- **Teleport Object** causes the target (character) to be teleported to the position of the given object, aimed in the same direction as this. The teleportation will fail if the target cannot spawn at the position.
- **Wait Impulse** causes the engine path or script (it’s doesn’t do anything for other objects) to stop when received, and then wait for the specified impulse before continuing. This is very useful when several things need to happen before a script should continue. For example you could have an engine path sture with two wait impulse 1 messages with target \$this at time 0, and a message that will open a door at time 2. If you then have three guards who all have OnDie: Impulse 1, target sture, the first guard to die will start sture, and the two following will get sture past the two wait impulses, thus opening the door 2 seconds after all three guards have died. Note that since a path must be started to send messages (even wait impulses to self) I had to have one fewer wait impulse than guard in this

example, but I could also have used autostart on sture and three wait impulses, of course.

- ii. The **Target Name** of the object(s) that should receive the message. In addition to the specific names of objects one can also use the following reserved names:
  - **\$activator**, which means that the target is the object which activated the trigger/engine path/bot event. This activator is propagated to any indirectly activated engine paths, so if a player activated a trigger which in turn activates an engine path, and the engine path has a message with target name \$activator, the message will be sent to the player object.
  - **\$this**, which means that the target is the object itself.
  - **\$game**, which means that the target is the game controller object. End round messages should for example be sent to this target name.
  - **\$player**, which means that the target is the closest player.
  - **\$allplayers**, which means that the target(s) are all players.
- iii. Additional parameters, which affect the message. For AI impulses one of these are the type of AI Impulse, see below.

#### Some examples:

Stop the engine path "Path01"

Message Type: Impulse

Target Name: Path01

Message Parameter: 0

Message String: -

Start the (first sequence of the) engine path "Path01"

Message Type: Impulse

Target Name: Path01

Message Parameter: 1

Message String: -

Release the bot "Sture" from script control:

Message Type: AI Impulse

Target Name: Sture

Message Parameter: Release

Message String: -

Set the bot "Sture" to patrol along the engine path "Path01"

Message Type: AI impulse

Target Name: Sture

Message Parameter: B. Patrol

Message String: Path01

Begin showing a cut scene to the player that activated this message sender

Message Type: Begin Cutscene

Target Name: \$activator

Message Parameter: -

Message String: -

Spawn the bot "Sture"

Message Type: Spawn

Target Name: Sture

Message Parameter: -  
Message String: -

## AI Impulses

- The basis of detail scripting of bots is the issuing of AI impulses. Whenever you want a bot to do something that you want full control over you must use an AI impulse. In addition some AI impulses can be used to complement independent behavior, in particular the *Add/Remove friend*, *Spot Character*, *Pause*, *Release*, *Unspot All* and behavior changing AI impulses.
- The following AI impulses force the bot to do something for a period of time. When a bot is given such an impulse it comes under *script control* and will stop doing anything other than what it has been explicitly told to do. However, if the tasks it has been ordered to perform ends, it will be automatically released from script control and revert to its normal behavior. The impulses which start tasks that are described as *continuous* will never end by themselves.
- There are also some impulses that explicitly end script control. These are described below later.
- The different tasks also depend on certain *devices*, namely the *Animation*, *Move*, *Look*, *Jump*, *Weapon*, *Item* and *Stance* devices, and each device can only be used by one task at a time, so tasks using the same device can override each other. For example, if a bot has received a “Vector Move“ impulse, it start moving according to this, but if it later receives a “Move To Object” impulse, this will override the previous task, so the bot will quit moving according to the “Vector Move” task and start moving to the object instead.
- **Aggressive.** The bot will continually aim and attack any spotted enemy if he thinks he can hit them. He will not move though. Useful in combination with movement scripting or when you want a bot with ranged weapon to stand still. Uses look and weapon device.
- **Aim.** The bot aims at the given object continuously. Uses Look device.
- **Animation.** The bot performs the given animation until the animation ends. Some animations are looped and are thus continuous. Uses Animation device.
- **Anim Torso.** The bot performs the given animation using the upper body only, until the animation ends. Some animations are looped and are thus continuous. Uses Animation device.
- **Attack.** The bot makes an attack towards the given object and then switches to continuously aiming at the object. The attack is subject to the usual targeting penalties, so it might miss. Uses Look and Weapon devices.
- **Crouch.** Toggle crouching. Crouches continuously when toggled on, and ends when toggled off. Uses Stance device when toggled on.
- **Fly Follow Me.** Continuous. Bot will stop engine path and move to it, using normal path finding. When at path he will start path and then start flying with same velocity as path had previous frame. Useful when performing controlled

jumps or slow-moving flying, but will be somewhat erroneous for fast paths. Otherwise, see Follow Me.

- **Fly FM True.** Continuous. Exactly the same as Fly Follow Me, except slightly more expensive, and correct with high flight speeds.
- **Follow Me.** Continuously follow the path which sends this message if Move device is available and looks the same way as the path does if Look device is available. If falling behind the path, bot will stop path and move to path using normal path finding, starting the path again when near. Bot will thus always follow slightly behind path, and never move faster than it's top speed (or idle max speed). Note that since bot continually sends impulses to path, it will automatically become the paths activator. This is the most effective way to control a complex bot movement, while still maintaining good control. Does not use any devices.
- **Follow Path.** As Follow Me except that bot will start following path with the given target name.
- **Force FM Relative.** Bot will mimic paths movement, at best speed he can manage, but will never stop path and won't use path finding. Useful when wanting to move a large group of bots in a coordinated fashion, but one must be careful since bots can get stuck or fall off edges etc. Otherwise, see Follow Me.
- **Force Follow Me.** Bot will always move towards path in a direct line, without stopping path or using path finding. Useful when wanting to move a bot over area where path finding doesn't work, for example through a teleporter or on a moving object. Use with care though, since bots won't care about getting stuck, falling off edges etc, and since path isn't stopped, you must time the speed of the engine path carefully. Otherwise, see Follow Me.
- **Jump.** Jump once then ends. Uses Jump device.
- **Look at object.** Look at the given object continuously. Uses Look device.
- **Move to object.** Move to the given object . Uses Move device.
- **Pause.** Stand still continuously. Uses Move Device. This impulse is quite useful when you want a bot to do nothing until we "wake it up" with a release impulse.
- **Smart look.** Look at the eye-position of any given object continuously. Uses Look device.
- **Switch Weapon.** Switch weapon once, or switch to the given weapon if any, then ends. Uses weapon device.
- **Switch Item.** Switch item once, or switch to the given item if any, then ends. Uses Item Device.
- **Teleport Follow Me.** Continually teleports bot to path position. This doesn't stop path. If bot cannot spawn at paths position, bot isn't moved until path reaches a position where bot can be spawned. Useful for simulating "flying" at high speeds (bot movement is interpolated on client, so this looks like it flies normally, although it really doesn't). When wanting to teleport a bot once, use Teleport Object message (see above) instead. Otherwise, see Follow Me.
- **Use Weapon.** Use the current weapon once, then ends. This is equivalent to the player holding the attack button for one frame. Uses weapon device.
- **Use Item.** Use the current item once, then ends. Uses Item Device.

- **Vector Look.** Change look direction by the amount of the given comma-separated vector each frame continuously. Uses Look Device.
- **Vector Move.** Move in the direction given by the comma-separated vector. The values should be on the interval  $-1 \dots 1$  and the three values are forward/backward, strafe left/right and go up down, respectively. For example, “1,0,0” is full speed forward, “-1,0,0” full speed backward, “0,0.5,0” is strafe right at half speed. Uses Move Device. Useful when you want to force a move, regardless of pathfinding, such as when you want someone to step off a high edge.

The following AI impulses generally govern how a bot should fight in melee, but not exactly. Thus they can be used to quickly create moderately choreographed melee combats. The results of such a script is unfortunately not the same every time it’s run, though. Bots that receive any of these impulses will be locked in scripting as long as there’s a valid target around:

- **Hold Attacks.** Bot will move normally, but won’t attack.
- **Release Attacks.** This negates a previous Hold Attacks impulse, so that bot starts attacking normally again.
- **Charge.** Bot will charge opponent attacking when in range.
- **Dodge.** Bot will evade opponent, but attack if he gets too close.
- **Flank.** Bot will slowly circle opponent, or charge in if behind him. Will attack when possible.
- **Parry.** Deprecated.
- **Riposte.** Bot will charge in for a quick attack and then dodge away.
- **Wait.** Bot will stand fast, occasionally taunting opponent. Will not attack, but will dodge if opponent gets too close.

The following AI impulses releases the bot from script control, effectively ending any scripted tasks prompted by AI impulses and causing the bot to revert to it’s normal behaviour (which may be changed by this impulse). The behaviours are described later:

- **B. Ambush.** Change behaviour to ambush, with any given object as target.
- **B. Avoid.** Change behaviour to avoid.
- **B. Engage.** Change behaviour to engage, with any given object as target.
- **B. Escape.** Change behaviour to escape, with any given object as position parameters.
- **B. Explore.** Change behaviour to explore
- **B. Follow.** Change behaviour to follow, with any given object as leader.
- **B. Guard.** Change behaviour to guard, with any given object as VIP.
- **B. Hold.** Change behaviour to hold, with any given object as position parameter. Optionally you can specify hesitance distance and max distance as well with a string of the following format: <object target name>;<hesitance d.>;<max d.>
- **B. Lead.** Change behaviour to lead, with any given object as path.
- **B. Patrol.** Change behaviour to patrol, with any given object as path.
- **B. Survive.** Change behaviour to survive.
- **Release.** Just release the bot from script control, so that it reverts to any current behaviour.

The following AI impulses will not directly influence the bots general behaviour, but will only give the bot information which it can use to act independently upon or cause it to do something that doesn't affect it's general behaviour. These impulses will neither cause the bot to come under script control, nor will they release a bot from script control:

- **Add Friend.** The given object (i.e. the object specified by the target name in the message string) is added to the bots friends.
- **Damage.** The bot receives the given amount of damage, possibly killing it. If a negative amount is specified the bot heals that much instead.
- **Death Animation.** Sets the death animation of the bot to given one.
- **Ghost Mode.** Bot loses/regains physics if param is 1/0. This means bot can move absolutely freely, without gravity or collisions, although you will need to use fly follow me or some other scripting to get it to move up or down. If you send ghost mode 1 to a released bot, funny things will happen, so I suggest you script any such tightly. This is mostly useful in cut scenes or when teleporting bots to make sure they'll spawn even though they aren't normally able to do so. While in ghost mode a bot can walk through a player and vice versa, but it can still be damaged normally.
- **Force Shoot.** Bot makes an attack against the specified target immediately, regardless of whether weapon has reloaded etc. Deprecated.
- **Force Use Item.** Bot uses item immediately, regardless of whether it should be able to or not. It does not aim. Deprecated.
- **Force Use Weapon** Bot uses weapon immediately, regardless of whether it should be able to or not. It does not aim. Deprecated.
- **Push.** Bot receives the given speed in the direction of the sending object.
- **Remove Friend.** Remove the given object from the list of friends.
- **Speak.** Speak the given line. Deprecated.
- **Spot Character.** Immediately spot the given character.
- **Unspot All.** Bot "forgets" all spotted characters.

The following AI impulses will change the bots personality, without causing the bot to come under script control, or releasing it. The personality types are describe below:

- **Berserker**
- **Coward**
- **Normal**
- **Passive**
- **Sneaker**

The following AI impulses will change some bot attributes, without causing the bot to come under script control, or releasing it. The attributes are described later in this document:

- **Activation Range**
- **Awareness**
- **FOV**
- **Health**
- **Hearing**

- **Hearing Range**
- **Idle Max Speed**
- **Rank**
- **Reaction Time**
- **Reasoning**
- **Sight**
- **Sight Range**
- **Skill**
- **Team**

## **Independent bots**

A bot released from scripting, or independent bot will take actions on it's own when it deems appropriate. The basic nature of its actions, given the current situation, is determined by its behaviour in combination with its personality and some other minor attributes. Its understanding of the situation is dependent on its perception attributes. In addition to controlling a bots general behaviour using the above attributes, one can send certain impulses (as detailed above) to influence behaviour. Note that when bots move around independently, they will always use normal path finding, and if they get stuck they will try to jiggle free according to what they think is best at the moment. This gives them the greatest flexibility, but of course reduces the game play designer's control. If bots behave in an unwanted fashion, scripting will have to be used to force them in line. Also, independent behaviour is constantly subject to revisions and tunings, which may change bot behaviour. Thus, if you need a bot to perform a specific task in a dependable manner, use scripting, which I strive to keep as backwards compatible as possible.

The attributes of a bot are listed in the sections below. I'll start by describing the behaviour attribute, along with behaviour parameter attributes, and will then continue with the personality and perception related attributes, followed by the remaining miscellaneous ones. There are also some general character attributes that are useful to know about, although they aren't really AI related, which will be described last. When setting attributes in Ogier, you just specify the value in the key field with the appropriate name, as usual. Bot attributes can also be conveniently set in templates, especially for balancing purposes, but then you must use the unique key identifier, which I'll provide within parenthesis. Those attributes that currently cannot be set in Ogier is marked with an asterisk (\*). The reason for this is usually that they should be changed only in templates so that all bots of the same type will act similarly. If wanted it is very simple to make them available in Ogier as well, though.

## **Behaviour (AI\_BEHAVIOUR)**

The behaviour is the major governing factor on how an independent bot will act. This controls how the bot moves, takes normal actions and how efficiently it uses its perception (i.e. how alert it is). For example, most behaviours scan for enemy presence and react upon it according to personality, with some actively moving or looking around while others are more passive. Each behaviour have the possibility to start sub-behaviours which will take over (part) of the decision making of the primary behaviour. The most common examples are the explore behaviour which is frequently started as a

sub-behaviour when something suspicious is detected or the engage behaviour which is the normal sub-behaviour to start when an enemy is spotted. When I say “switch to behaviour X” below, I really mean start up a sub-behaviour of type X. “Normal combat behaviour” means the behaviour personality chooses when a combat situation occurs, see personalities below.

The currently implemented behaviours are described below.

### **AMBUSH**

This behaviour is only rudimentary implemented. Bots will hold position and switch to ranged weapon if they have one, using them if enemy comes in line of sight. If given an target param(AI\_TARGET) the bot will always attack this target if it's in LOS, regardless of other potential targets, otherwise it will fire at will. If bot is attacked, he will switch to normal combat behaviour.

### **AVOID**

The bot will try to avoid enemies, fleeing when any are spotted. It will preferably flee towards allied characters. If given a target param (AI\_TARGET) it will try to avoid this character above others, and if given a leader param (AI\_LEADER) it will prefer to flee towards this character. If no enemies are spotted it will stay put.

This behaviour should be fixed to also take an area info object as parameter, so one can limit the area where the bot is allowed to run around inside. Otherwise important story-driving characters (like Marcus in Enclave) cannot use this behaviour for fear of them running away somewhere from where they can't get back.

This behaviour is meant to be started by other behaviours when a combat situation occurs, and will then automatically become inactive when there are no spotted enemies about.

### **ENGAGE**

The bot will follow and attack any spotted enemy. If the bot has a given target param (AI\_TARGET) it will always prefer to attack this target if possible.

When wielding a ranged weapon it will avoid close combat with melee weapon wielding enemies, but switch to melee weapon if forced into melee. Magic users will not switch to melee weapon though, although they should perhaps do so when waiting for mana to be restored. Depending on personality, they will choose between evading incoming enemy fire and standing still for more accurate aiming.

If wielding a melee weapon, bot will attempt to close with enemy and will then fight offensively or defensively according to personality. Setting the different melee factors described below can also control this. Bot does not actively cooperate with other allied combatants, although the use of escape sequences in melee combat will automatically cause multiple allies to surround single opponents.



Currently any combat decisions are largely unaffected by circumstances such as own and opponents injuries, numbers and equipment. This will be fixed in the future.

This behaviour is meant to be started by other behaviours when a combat situation occurs, and will then automatically become inactive when there are no spotted enemies about.

### **ESCAPE**

This behaviour is only rudimentary implemented. If the bot has one or more position params (AI\_POSITION), it will try to get to one of these randomly chosen. If there are no escape positions, the bot will switch to a Follow behaviour.

If enemies are spotted bot will switch to normal combat behaviour. If something threatening is detected it will Explore.

### **EXPLORE**

Bot will move fairly swiftly around in a somewhat random fashion, unless given a position (AI\_POSITION), such as when having decided to investigate a threat, in which case it will first visit that position before starting to explore randomly. When moving randomly there's a small chance that bot will search out allies with higher rank and Follow them for a while.

If enemies are spotted bot will switch to normal combat behaviour.

This behaviour should be expanded to take an area info object to delimit the area where the bot should explore, and should be made more intelligent, for expmlae preferably going to areas that haven't been visited yet, etc.

### **FOLLOW**

If the bot has a given leader param (AI\_LEADER) it will follow him. If it doesn't have a specific leader or that leader is unavailable it will instead wait around until an ally with higher rank comes close, and then designate this ally as leader. While following a leader it will try to keep fairly close, but not too close, matching speed with leader when within this "safe" distance, but running to keep up if falling behind. When stopping it will scan around in the same general direction as the leader.

If enemies are spotted bot will switch to normal combat behaviour. If something threatening is detected it will Explore. In both cases, the bot will return to the leader once the threat has been dealt with.

### **GUARD**

This behaviour just default to follow behaviour currently. I will fix it so that bots with this behaviour will follow a VIP and try to shield him from ranged attacks, as well as engage any melee weapon wielding enemy that comes too close to the VIP.

## **HOLD**

If given a position param (AI\_POSITION) bot will stand guard at this position, looking in the direction of the position object. If no position is given bot will use it's spawning position. While at the position bot will slowly scan about in an approximate 90 degree arc. If not at position (and not under influence of a sub- behaviour) bot will move there.

If enemies are spotted bot will switch to normal combat behaviour. If something threatening is detected it will Explore. However, if the max distance param (AI\_MAXDISTANCE) or hesitance distance param (AI\_DISTANCE) are given, then these will restrict any movement the sub-behaviour is allowed. If the max distance is 0, bot will always be stationary, no matter what. Otherwise bot will always start back for it's hold position when he gets to the max distance range, and there a chance he will start back every frame once he gets past his hesitance distance range. It is, however possible for an opponent attacking the bot in melee to drive the bot outside of his max distance range, but he will try to get back as soon as possible. He will never abort this special "go back" move until he comes within hesitance distance range again. Be careful when using max and hesitance distance. They should always differ at least 100 units to avoid any "twitchy" movement. Also it's not recommended to use a max or hesitance distance with a melee-weapon armed bot, since he will always try to reach the enemy this will make the bot look like a dog on a leash if the enemy stand just outside of the max distance range. Then again this is quite ok if the bot actually is a dog (or other mindlessly homicidal beast) on a leash ☺.

## **LEAD**

This behaviour is identical to the Patrol behaviour, except that one can also specify a follower param (AI\_LEADER) which is the character that the bot should lead along the path. If the follower is unspecified the bot will use the nearest player as follower. If there's both a path and a follower, the bot will lead the follower along the path, stopping if the follower gets too far away from the bot and turning towards him. This should be reinforced more with visual and vocal cues later.

If enemies are spotted bot will switch to normal combat behaviour. If something threatening is detected it will Explore. When any threat has been dealt with, the bot returns to the path and continues to lead.

If there's no path to follow the bot stands still looking at follower.

## **PATROL**

If the bot has a given path param (AI\_PATH), which is an engine path, it will walk along the path (starting it if necessary, and stopping it if it gets too far ahead). While following the path, it will always look in the same direction as the path itself. The bot will also try to match speed with the path, but if the path moves too quickly the bots stops it when it gets ahead and moves to it using normal path finding. This means that a really fast path will always cause the bot to move at maximum speed (or idle max speed, see below). When reaching the end of a path (if the path isn't looping), the bot will stay put, looking in the direction of the path.

If enemies are spotted bot will switch to normal combat behaviour. If something threatening is detected it will Explore. When any threat has been dealt with, the bot returns to the path and continues to patrol.

If there is no path to follow, the bot switches to the Hold behaviour.

## **SURVIVE**

The bot will randomly move around slowly from time to time, or stand idle while looking around lazily. It also has a fairly high chance of going to the position of an ally, and switching to Follow behaviour if he's a suitable leader.

If enemies are spotted bot will switch to normal combat behaviour. If something threatening is detected it will Explore.

This is the default behaviour for most bots.

Most behaviours take one or more parameters. Most parameters should be names of objects, preferably objects that are used for no other purpose than the behaviour. If there are more than one object with that name, one will be selected randomly, unless the behaviour can handle multiple parameters of the same kind, in which case all such objects will be used. The behaviour parameter attribute keys are listed below:

- **Position Param (AI\_POSITION).** The name of any (preferably static) object. When world is created bot will retrieve the position and/or direction of this object.
- **Path Param (AI\_PATH).** The name of an engine path.
- **Target Param (AI\_TARGET).** The name of an enemy character.
- **Leader Param (AI\_LEADER).** The name of an allied character.
- **Area Param (AI\_AREA).** The name of an area info object.
- **Distance Param (AI\_DISTANCE).** A distance (lower) limit in world units.
- **Max Distance Param (AI\_MAXDISTANCE).** An upper distance limit in world units.

Bots with the Engage behaviour (the behaviour used by all bots when fighting) can have their melee behaviour altered by setting a few attributes. These are useful for creating different melee behaviour for different types of opponents, but are not available to set in Ogier (although this can be easily added if necessary). When in melee a bot will choose one of several combat manoeuvres, with the possibility to switch manoeuvre at regular intervals. The following values are integer multipliers to the relative chance that a manoeuvre is used. The default value is 10 (i.e. set the value to 10 for normal chance, 0 for no chance of choosing the manoeuvre, 20 for double chance, etc). The keys are listed below, with description of the combat manoeuvre:

- **Charge\* (AI\_CHARGE).** Bot will press his attack, mostly moving towards the opponent, and only occasionally taking a few steps back to consolidate after attacking. The bot can perform combo attacks in this manoeuvre if the combo use flag is set (see below).

- **Dodge\* (AI\_DODGE).** Bot will try to keep distance between enemy and self by stepping backwards or to the sides, attacking if enemy presses too close.
- **Flank\* (AI\_FLANK).** Bot will circle around enemy slowly, and moving in for an attack if at targets rear. Will always attack target if near enough though.
- **Riposte\* (AI\_RIPOSTE).** This manoeuvre should really be called lunge or something. Bot will charge in for a single attack, before dodging away again.
- **Standfast\* (AI\_STANDFAST).** Bot will stand fast and attack opponent only if he comes within range.
- **Wait\* (AI\_WAIT).** Bot will wait for opponent to act, being better at detecting enemy manoeuvre while in this manoeuvre. If opponent comes close bot will dodge, but never attack. Will occasionally stop entirely to taunt opponent.

Finally there are some miscellaneous attributes that influence behaviour:

- **AI Max Speed\* (AI\_MAXSPEED).** The maximum speed the bot will ever attempt to move at, even though it can move faster.
- **Evade Speed\* (AI\_EVADESPEED).** The speed the bot can run backwards when evading in melee combat. Useful to influence the difficulty (and annoyance) level of fighting the bot in melee.
- **Idle Max Speed (AI\_IDLESPED).** The maximum speed the bot will move at under normal circumstances (i.e. when just strolling around scratching its butt). Very useful when one wants to have bots patrol (or when under Follow Me scripting) at a certain speed, since one just set the idle max speed and need not bother about making the path go at the correct speed.
- **Use Ranged At (AI\_USERANGED).** Influences the engage behaviour only. When target is above this range, bot will prefer to use ranged weapon. When below he will prefer to use melee weapons.

## **Personality (AI\_PERSONALITY)**

The personality determines how the bot handles stressful situations, such as when an enemy is spotted, when in melee, when in ranged combat and when pursuing an enemy. The sections below describe each personality type in detail.

### **BERSERKER**

Berserkers are very straight forwardly aggressive: they always Engage spotted enemies and never make evasive actions while closing with them. In melee they tend to charge, with slightly lesser inclination towards flanking. They will almost never dodge.

### **COWARD**

Cowards will always try to avoid combat (thus choosing the Avoid behaviour when spotting enemies). They will always try to evade threatening enemy fire and if for some reason forced to engage in melee will be very defensive.

### **NORMAL**

Bots with normal personality will attack enemies (i.e. choose Engage behaviour), but with a certain amount of prudence. They will take evasive actions if being attacked by

ranged weapons and will use a fairly well balanced mixture of offensive and defensive manoeuvres in melee.

### **PASSIVE**

A passive bot has taken way too much Prozac. They won't react to danger at all, and will never attempt to attack anyone (i.e. won't switch behaviour when spotting enemies even if they attack him).

### **SNEAKER**

Sneaky bots will choose to Ambush enemy if possible. Since this behaviour is rather rudimentary the sneaker personality won't work that well currently. They will evade enemy fire a lot and use circumspect melee manoeuvres such as flanking and riposte in favour of charges.

I plan to use a number of personality parameters instead of the set personalities in the future. Parameters that define the bots aggressiveness, carefulness, hurry, loyalty, curiousness, greed, sneakiness etc will interact so that one can create a very large number of complex personalities. The basic personalities described above will still be usable, and will provide defaults for the personality parameters.

## **Perception**

Bots perceive the worlds with different senses, remember perceived information and can reason (or at least fake reasoning) about it. How alert the bot is, is governed by a bunch of perception attributes, which either qualifies how good the bot is at that particular form of perception or delimits some perceptive ability in some way.

The main senses are sight, hearing and awareness, where the latter is a convenient value to use for creatures with magical perception, smell etc. The bot remembers information about all other important characters (mainly enemies) such as if they're detected (presence suspected), spotted (presence and general location definitely known), hostility etc. A bot can detect a character if he hears it, glimpses it (i.e. sees it but not clearly) or automatically if being attacked by the character. A bot can spot a character if it's been previously detected and is seen clearly, if it comes within awareness range or when attacked by it if he is smart enough (succeeds with a reasoning test). The normal reaction to detecting a character is to look in the general direction of the detected character in the hopes of spotting him, and if this fails, possibly going there and exploring a bit. When an enemy is spotted the bot will react according to personality, which usually is to attack, take cover or flee.

The perception attributes are listed below:

- **Awareness (AI\_AWARENESS).** Anything closer than this range is automatically spotted.
- **FOV (AI\_FOV)** is the field of view in degrees (0-360). The bot can't see anything outside of its FOV.
- **Hearing (AI\_HEARING)** is a qualitative value between 0 and 100 that determines the chance that the bot will hear something. This chance is modified

by own and target movement and activity, as well as if there are any obstacles in between.

- **Hearing Range (AI\_HEARINGRANGE).** The maximum range a bot can hear anything.
- **Sight (AI\_SIGHT)** is a qualitative value between 0 and 100 that determines the chance that the bot will see something. This chance is modified by own and target movement as well as general sight conditions (and line of sight, of course).
- **Sight Range (AI\_SIGHTRANGE)** is the maximum range that a character can see anything.
- **Reasoning (AI\_REASONING)** is a qualitative value between 0 and 100 that determines the chance the bot has of making reasoned observations (such as determining from where a shot comes etc) and general intelligence level of the bot. Not used as much as I'd like to, but it will have more impact later.

## Miscellaneous Attributes

In addition to the attributes described above there's a bunch of other minor ones that can be used to tweak a bots behaviour and performance:

- **Activation Range (AI\_ACTIVATIONRANGE).** A normal bot will be completely inactive (not even seeing or hearing things) until attacked or a player comes within this range.
- **Activation Time\* (AI\_ACTIVATIONTIME).** A bot can turn itself off if, when it doesn't think it's useful anymore. It will always consider turning itself off for this number of seconds before actually doing it.
- **Aiming bonus\* (AI\_AIMBONUS).** All bots with ranged weapons will aim better at stationary targets, but if you want this effect to become even more marked (such as for a sniper) then set this value. This is the skill bonus the bot gains each frame the target remains stationary.
- **Friend (AI\_FRIEND).** The target name of an allied bot. If spotting an enemy or coming under attack, this bot will always call on its friends for help. The friends will then automatically detect (not spot) the threat to the bot. Bots with the same name are automatically friends.
- **Default Heading\* (AI\_DEFAULTHEADING).** Deprecated.
- **Destroy Timer\* (AI\_DESTROYTIMER).** The bot will be destroyed (i.e. removed from the world) after this number of seconds after being spawned. The OnDie event will be raised when destroyed, but there will be no death animation or effect.
- **Hit Ratio (AI\_HITRATIO).** A bot with hit ratio less than 100 will sometimes force himself to miss a shot, even if he should hit. This is useful if you want several bots to lay down a barrage of fire, without actually killing the character. Use this with caution however, since this will look extremely weird at close ranges.
- **Idle sound interval (AI\_MIN\_IDLE\_SOUND\_INTERVAL, AI\_MAX\_IDLE\_SOUND\_INTERVAL).** The minimum and maximum time between idle sounds (i.e. noises the bot makes when no enemy have been detected or spotted).

- **Life Drain\* (AI\_LIFEDRAIN).** The amount of damage per frame (this can be less than 1) the bot takes when it's life timer has expired (see below).
- **Life Timer\* (AI\_LIFETIMER).** If set, this is the number of seconds after the bot has spawned until the life drain (see above) sets in.
- **Min health\* (AI\_IMMORTAL).** If set, this is the minimum health value the bot can be reduced to normally. A bot with a value higher than 0 cannot die.
- **Rank (AI\_RANK).** The rank of the bot (0 - 255) is primarily used by the Follow behaviour. Bots without designated leaders will only follow bots with higher rank, usually the closest such bot. Rank and follow behaviour can therefore be used to create rough formations, or marching order. Players always have the maximum rank.
- **Reaction Time (AI\_REACTION).** The time in seconds the bot takes to react to new information.
- **Skill (AI\_SKILL).** A qualitative measure (0 - 100) of the bots precision with ranged weapons and ability to predict opponent's manoeuvres in melee.
- **Start Activated (AI\_ACTIVATED).** A bot with this flag set (i.e. value 1 in template) will always be active, and never consider turning itself off.
- **Taunt Chance\* (AI\_TAUNTCHANCE).** The probability per five frames that the bot will start a taunt when using the wait manoeuvre in melee.
- **Use Flags (AI\_USEFLAGS).** Several flags which can be used to turn certain features on/off:
  - *alwaysaggressive.* The bot will always be aggressive when scripted, see scripting AI impulses above.
  - *alwaysblock.* The bot will use any shield frequently.
  - *combo.* The bot will use melee combos.
  - *magicshield.* The bot will use magic staff shields.
  - *scriptmute.* The bot will not make any noises when scripted. Useful for avoiding sleeping bots making detection noises.
  - *shockwave.* The bot will use magic staff shockwaves.
  - *summon.* The bot will use summoning staff summon ability.
  - *switchweapon.* The bot will switch weapon when it wants to.
  - *all.* The bot will have all use flags set.

In a template the values should be specified as a "+"-separated string, such as "\*AI\_USEFLAGS switchweapon+combo", for example

## General Character Attributes

In addition to all the attributes described above, there are some attributes, that don't affect the AI itself, but will affect the character. I only give the names that'll show in Ogier along with the descriptions:

- **Character flags.** Several flags which can be used to turn features on/off:
  - *keepbody.* Don't know what this doe. Ask Jesse.
  - *nonpushable.* Character cannot be pushed aside by players. Useful for bots which must complete critical scripts.
  - *spawndead.* Character will be dead when spawned. Used when you want a corpse lying around, since normal models don't support skeleton animations.

- *waitspawn*. Character won't be spawned until receiving a spawn message. This is very useful, since you can control precisely when a bot should come into play, and it saves processing power.
- **Dialogue.** The dialogue file the character uses.
- **Drop Item.** The pickup the character drops when he dies.
- **FA Start Time.** If force animated, this is the time into that animation where bot will start performing it.
- **Force Anim.** Character will automatically perform an animation. Somewhat deprecated.
- **Health.** Hit points.
- **Item 0 / Item 1.** The items a character has is mainly described in templates, but you can vary them in Ogier by setting these keys. If you give the bot an item of the same item type as one specified in the template, the template weapon is removed. For example, if you give an orc a battle axe and he has a short axe in the template, the short axe is removed. If you instead give him a crossbow and a shield, he will still have the short axe, in addition to the crossbow and shield.
- **Max Health.** Upper limit to hit points received through messages, health potions etc.
- **Max Targeting Distance.** The maximum range at which a targeting ring and info will be shown if aiming at the character and using the Enclave auto-targeting system.
- **Speed.** The physical limit to the character's speed.
- **Team.** The team the character belongs to. In enclave this is dark or light, as well as a number of "neutral" teams: Bandits will consider everyone except other bandits their enemies. Friendly citizens will consider everyone allies. Angry citizens will consider everyone except other citizens enemies. Aggressives will consider everyone enemies.



# Using the Templar Camera system

This document describes how to use Ogier to place cameras for the Templar game. It will describe the various types of cameras and their attributes, and also give tips on how they could be placed to avoid bad situations.

## Placing Cameras

Cameras are placed in the viewport in Ogier by pressing F11 or Ctrl+F11. That creates two objects, a trigger\_ext and a camerapos. Both objects have generated names such as camTrigger001 (for the trigger) and cam001 (for the camera). The trigger is already linked to the camera by name, so if you need to change the cameras name for whatever reason you need to make sure that the trigger also has the name in the message changed. The camera also has a key named Trigger, with the name of the trigger that it was created with. This key does not influence how the camera is executed and only serves as a visual aid and points out in what area/areas the camera is active. And if a new trigger is added to a camera or if the name of a trigger is changed it is strongly recommended that you make sure that the Trigger fields in of the camera corresponds to all the triggers that activates the camera, or else you will be getting faulty visual feedback while using Ogier.

## Camera theory

### Triggers and messages

First off, every level must have a camera named StartCam. It is the camera that is active as the level completes to load. It is also good practice to see to it that the trigger for the StartCam is at the players start position. A camera remains active until the player enters a trigger for a new camera while not being in the trigger for the active camera. This means that both the area between two triggers that are not overlapping and the area that are overlapping is used as buffer zones where the camera that you are leaving remains active until you enter a new camera, and has completely left the last camera. The messages that activate a camera are impulses with a value of 1. Impulses with a value of 0 are used to signal that a transition is favoured, but more on that later.

### Camera types

Cameras should always be set up in the same manner, namely be using the predefined keys (F11 and Ctrl+F11). This ensures that the cameras have valid names and are set up correctly. However, if a camera is created like one type and it is desired to change type, it is not required to delete it and create another. The difference is simply some simple attributes that can be changed simply, as will be described.

#### **Point camera** - F11

A simple camera that has one position (no spline), and is linked to one or more triggers.

#### **Rail camera** - Ctrl+F11

If a camera has more points, making it a spline, it is considered a rail camera. A rail camera is either mapped against the camera spline itself, or a separate spline linked to it, hereby called rail.

There are three possible ways of mapping:

### ***Camera mapping***

The camera has no rail, so the mapping will be against the camera's spline itself. Meaning that the camera will try to be at the place along the camera spline, that is the closest to the player. Setting a value to Rail Distance for the camera will make the camera be tolerant up to that distance with changes in the mapping. It will be like a string where you pull around the camera, and the camera will only move when the string is stretched.

### ***Camera mapping with dimension***

The camera has a rail, and that rail has Rail Dimension checked. Works like camera mapping, except the distance for the tolerance is only calculated in the dimension set by the rail. This means that if you move perpendicular away from the rail spline it will not count it as distance.

### ***Rail mapping***

If the camera has a rail, and it is not checked as Rail Dimension, the camera will be mapped to the same percentage of the camera spline as the player is closest to the rail. Please note that the camera's Rail Distance can be set to a higher value to achieve the drag effect.

### ***Rail distance mapping***

If the camera has a rail, and the Rail Distance value is not zero. The camera will be further down the camera spline the further away from the spline (or point for a circular mapping). If the Rail Distance value is greater than zero then the camera will be at the start of its spline when the player is Rail Distance units away, and the end when directly on top of the rail. If the Rail Distance is less than zero, the relationship is the opposite.

### **Transitions spline – F12 with two cameras selected**

To create a transition spline, select the two cameras that you want to create a transition between and press F12. Two dialogues will appear. The first one will ask you for the length of the transition, in seconds. The second will ask you for the number of subdivisions that you want the spline to have.

## **Transitions and interpolations**

There are three kinds of transitions:

### **Transition spline**

If both the previous and the next camera have an association to a transition spline, that spline will be used to make the transition. A transition spline works in the following way. The spline defines the way the camera must take, and the part of the transition that is not defined by a spline is performed like an interpolation. This means that you don't need to, and actually should not, make the spline go all the way to the cameras. This is especially important when the then transition is made to a camera that is not fixed in one place (rail cameras and point cameras with safe radius). If you expect the angle between the spline and the interpolation to be somewhat big, you can make sure the Trans Smooth is checked. One thing to pay attention to though, is that the camera will travel with the relative speed specified by the time of the key frames. For example, if a spline has three key frames A, B and C and A and B is much closer than B and C is. Then the camera will travel faster in the B-C section than in the A-B section.

The sensitivity during the interpolation will be the Sensitivity value of the trans spline. The time for the interpolation will either be equal to the Trans time value of the trans spline, if it is not zero. Or it will be the total length of the spline (the time of the latest key frame) plus the time for the interpolations to and from the spline, and these will vary depending on the distance but will have the same velocity as the spline has at its ends. You can have multiple transitions between cameras as well. To indicate which one of the transitions will be chosen you can set up a trigger area that sends an Impulse with a value of zero to the transition that you want the trigger to activate. This trigger does not have to trigger a camera as well.

## **Interpolation**

If both cameras have Interpolate All checked, and none of them has the other one in its interpolations list. Or if one of the cameras does not have Interpolate All checked, and the previous camera has the next one in its interpolations list. Note that the last way described is a one-way method for interpolations. If you want the interpolation to be two-way you will have to add camera A in camera B's list, and camera B in A's list. When an interpolation is performed it will first check if there is something in the way along the path. If there is, then a warning will be written in the log and, a snap is performed instead. You should try to keep the settings correct so an interpolation is not attempted where it is not intended, as it will make it easier to see in the log where the intended interpolations failed due to collision.

The sensitivity during the interpolation will be the greater of the two cameras Interpolation Sensitivity values or you could add an interpolation to the interpolations list and add a semicolon and a value for the interpolation behind (for instance, interpolation: cam042;70 will make the interpolation have a sensitivity of 70).

The time for the interpolation will be the greater of the two cameras Interpolation Time values.

## **Snap**

If none of the above applies, then a snap will be performed. The camera will be instantly transported to the new position, and aligned toward the player. This type of interpolation is preferred in narrow passages where there is no room for the camera to move around.

## **Camera attributes**

### **SafeRadius**

This is the radius that the camera is allowed to move within to avoid that the player is directly under the camera. Some collision detection is done to avoid that the camera goes through walls, but a good rule of thumb is that the safe radius should not be higher than a safe area all the way around the camera. The safe radius works in the following way. When the player enters the safe radius the camera is displaced the same distance in the opposite way.

### **SR Sens. Add**

When the safe radius is active and displacing the camera it can be good to have an increased sensitivity since the player will most likely be running around close to the

camera. This key sets amount the sensitivity is increased when the player is within the Safe Radius.

### **FOV**

The Field of View of the camera, the value is in degrees and should for obvious reasons not exceed 140 degrees or be below 20. A value higher than the default value makes it appear that the camera is more far away than it actually is, and buildings and characters may seem wider or higher depending on from where the view is. A value less than default create a zoom effect. And makes it look like the camera is closer than it actually is.

### **Sensitivity**

Sets the speed with the camera focuses on the player. The higher the value, the faster it will be. Increase this value if you know that the player will be moving fast by, or if the camera is moving (railcam or saferadius). Max value is 255.

### **Freelook LR**

Degrees that the player will be able to free look in the left-right dimension. Max is 180.

### **Freelook UD**

Degrees that the player will be able to free look in the up-down dimension. Max is 90.

### **Locked Yaw**

If checked, the camera is fixed in the left-right direction that it has in Ogier.

### **Locked Pitch**

If checked, the camera is fixed in the up-down direction that it has in Ogier.

### **Trigger**

No functionality at runtime and only serves as a visual aid to know which area triggers the selected camera. You will have to update these manually when changing the name of triggers or adding new ones.

### **Interpolate All**

Checkbox: When a transition is executed, if both the previous camera and the new camera have this value checked and not a common trans spline, an interpolation is performed.

### **Interpol Time**

Time for interpolation, during interpolation the length in seconds of the interpolation is the biggest value of the previous camera and the new camera.

### **Interpol Sens**

Sensitivity during interpolation, the sensitivity is the biggest value of the previous camera and the new camera. Max 255.

### **Interpolation\***

Target list, takes names of other cameras. If Interpolate All is checked, then this list cancels the interpolations. If Interpolate All is not checked then it is a list of cameras that will be interpolated.

### **Trans Spline\***

Enter the names of the transition splines that you want attached to this camera here.

### **Trans. OneWay**

Only used by transition splines. If this is checked, then the spline is only used in the direction starting with time zero and either an interpolation or snap is performed depending on Interpolate All and Interpolation.

### **Trans. Time**

Only used by transition splines. This is the total time for the transition, including fade in and out. This overrides the splines original length.

**Trans. Smooth**

Only used by transition splines. Check this box if the spline is not in a straight line with the target. It will smooth the way the camera enters and leaves the spline.

**Rail**

Name of the rail spline, if one is used.

**Rail Heightmap**

Only used by rail splines (Checkbox). Only matters if Rail distance is not zero. Check this if you want a distance map that also includes the height distance.

**Rail Distance**

If entered on a rail spline (or point if you want a circular mapping) then it makes it into a distance map. If this value is positive, then the camera will be at the end of the camera spline when the player is at the biggest distance and start of the camera spline when you are directly at the rail spline. If the value is negative then the opposite applies.

If entered on the camera spline, then it will create a tolerance distance for the camera. It will work like you are pulling the camera with a string that is the length of this value.

**Rail Dimension**

Only used by rail splines. If this is checked, then the rail spline is only used as a dimension for distance calculations and is never used for mapping against.

**Rail Speed**

Only used by rail cameras. This is the speed with which the camera will move towards the mapped position. Higher value is faster speed. Max is 255.

## Texture compilation

**Todo**

## Surface editor

### TEMP

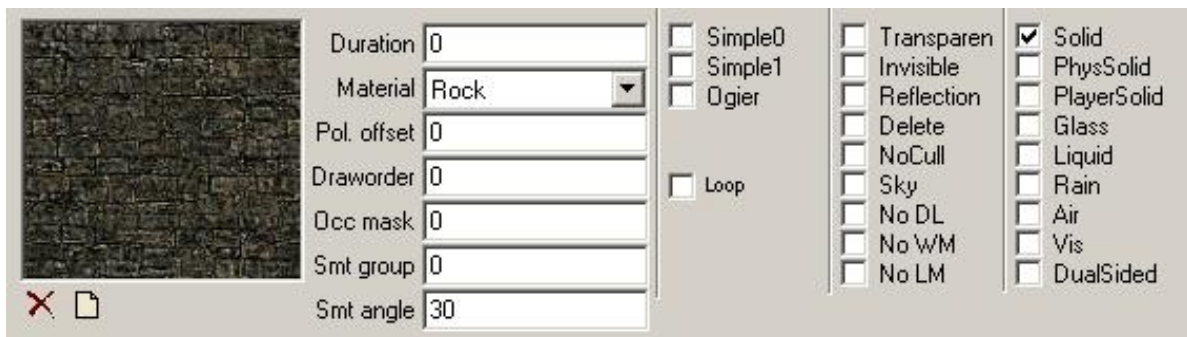
In the surface editor you can change the settings of your surfaces, keep in mind that if you change a texture it will change all of them on your map and though the whole game. If you want a surface to look a specific way but don't want to change the original you can create copies and change those. The surface files are located in your Surface folder. Default location should be, depending on project, something like this.

Ex

C:\Program Files\Starbreeze Studios\Knights Of The Temple\Sbz1\Surfaces

The surface files have the extension .xtx.

Open up Ogier and remove the write protection of the surface files you want to modify. To remove the write protection on a file all you have to do is to right click on the file and select properties, make sure that the read-only option inset selected. Once you have removed the write protection from the files just select Open in the Files menu or drag-n-drop the file onto Ogier.



#### Duration:

**Material:** This setting sets the material type for the object. This effects the sound effects that an object emit when walked upon among other things.

**Pol. Offset:** wallmarks

**Draworder:** Enclave och KoTT

**Occ mask:** Riddick+

#### Smt group:

**Smt angle:** Smoothing angle, this sets the max angle you can have before the texture cuts of and don't smooth the edges. A ground texture could probably go a lot higher then the one in the example does, then you create corners in a city environment you might not want the corners to smooth as much as on a ground texture.

#### Set Options

**Simple0:**

**Simple1:**

**Ogier:**

**Loop:**

**Set flags**

**Transparent:**

**Invisible:**

**Reflection:**

**Delete:**

**NoCull:**

**Sky:**

**No DL:**

**No WM:**

**No LM:**

**Set medium\_flags**

**Solid:**

**PhysSolid:**

**PlayerSolid:**

**Glass:**

**Liquid:**

**Rain:**

**Air:**

**Vis:**

**DualSided:**



## Model conversion

**Todo**

## Animation conversion

**Todo**

# Dialogue

The dialogue files should be found in the ...Content\Dialogues folder, related files like the characters should be located in the ...Content\Registry folder.

The locations may vary between different projects though.

## Dialogue settings

### Subtitle flags

Syntax:

```
*subtitleflag "[N]"
```

N =    0 Casual/Idle  
         1 = Interactive  
         2 = AI  
         3 = Cutscene

Example:

```
*0
{
    *SUBTITLE "Wanna trade some UDs?"
    *SUBTITLEFLAG "1"
}
```

### Priority

Syntax:

```
*priority [N]
```

N =    0 min  
         255 max

Example:

```
*0
{
    *subtitle "Get ready to die, Riddick!"
    *subtitleflag 2
    *priority 255
}
```

### Approach items

Syntax:

```
*setapproachitem "{ character},{ approach id}"
```

Example:

```
*setapproachitem "Quintana,-101"
```

Note:

Negative approachitems refer to the speaker's dialogfile.

Positive approachitems refer to the player's dialogfile.

### **Link conditions**

Test for parameter value

Syntax:

```
*link {linkto}|ParamEquals:{object}:{value}
```

```
*link {linkto}|!ParamEquals:{object}:{value}
```

Examples:

```
*link Player:136,148|ParamEquals:ep_hallo:2
```

```
*link Player:136,148|!ParamEquals:$this:3
```

```
*link Player:136,148|ParamEquals:$player:3
```

Test for player object ownership

Syntax:

```
*link {linkto}|POwns:{objectid}[:{amount}]
```

```
*link {linkto}|!POwns:{objectid}[:{amount}]
```

Examples:

```
*link Player:200|POwns:0x122:20
```

```
*link Player:201|!POwns:0x185
```

General message test

Syntax:

```
*link {linkto}|C:{condition}
```

Example:

```
*link "Monster:151|C:0x3f;Monster;1;;0;1;"
```

### **Sending messages**

Syntax:

```
*message {message}
```

Example:

```
*message "0x1014;$player;0;pickup_smokes_10"
```

### **Impulses**

Send an impulse to speaker object.

Syntax:

```
*impulse {impulse id}
```

Example:  
\*impulse 0

### **Subtitle/Voice range**

Set subtitle and voice attenuation ranges.

Syntax:  
\*subtitle\_range [range units] (default range is 576. Fadeoff is 64 units)  
  
\*attenuation [max units],[min units] (default attenuation is 1024,64)

Example:

```
*dialogue Dlg_Default_Guard
{
    *subtitle_range 1024
    *attenuation 512,32

    *100
    {
        ...
    }
}
```

### **Listener**

In a dialogue item you can set:  
\*Listener "player"

There are some extra flags that you can use if you want:

KeepBehaviourSpeaker  
KeepBehaviourListener  
NoLookSpeaker  
NoLookListener  
NoCrowd  
StopBehaviourCrowd

Example:  
\*Listener "player,KeepBehaviour+NoLookListener"

This would mean that the player is the listener; the one who is talking should not abort its behavior, and won't look at the listener (player).

## Dialogue editor

**Todo**

# Ogier Shortcuts/Hotkeys

Ctrl+Tab: switch between different documents.

## 2D View

“W”: Fast move up

“S”: Fast move down

“A”: Fast move left

“D”: Fast move right

Shift: Enable mouse scroll

Mouse-wheel up: Zoom in

Mouse-wheel down: Zoom out

Shift+“W”: Zoom in

Shift+“S”: Zoom out

“+”: Zoom in

“-“: Zoom out

## 3D View operation

“W”: Forward

“S”: Backward

“A”: Strafe left

“D”: Strafe right

Shift: Enable mouse look

Middle mouse-button: Enable mouse look

Ctrl+“A”, rotate around the selection

Ctrl+“D”, rotate around the selection

Mouse wheel up, step forward

Mouse wheel down, step backward

Shift+Mouse wheel up: move up

Shift+Mouse wheel down: move down

Shift+Right mouse button: move up

## Scale

Ctrl+“W”: Fast scale up

Ctrl+“S”: Fast scale down

Ctrl+“A”: Fast scale left

Ctrl+”D”: Fast scale right

## Texturing

Left mouse-button:

Scroll with 5 units step in selected plane

Holding Shift will allow stepping 1 unit at a time instead of 5

Right mouse-button:

Rotate with 5 degrees step in selected plane

Holding Shift will allow stepping 1 degree at a time instead of 5

## Rotate

Rotation circle:

Rotate with 5 degrees step in selected plane

Holding Shift will allow stepping 1 degree at a time instead of 5

## Brushes

'1': Create cube

'2': Create wedge

'3': Create spike

'4': Create spikewedge

'5': Create cylinder, param0 = segments

'6': Create cone, param0 = segments

'7': Create sphere, param0 = x-segments, param1 = z-segments

'8': Create spike cube, cube made out of 6 spikes

Shift+'1': Create spline-cube

Shift+'2': Create spline-pie

Shift+'3': Create spline-cylinder

Shift+'4': Create spline-arc 1

Shift+'5': Create spline-arc 2

Shift+'6': Create spline face (use with care)

## File

New map                      Ctrl+N

Open                          Ctrl+O

Close                         Ctrl+F4

Save                          Alt+S, Ctrl+F12

Map info                     F8

Reload DLLs                 Ctrl+F5

Exit                           Alt+F4

## Edit

Undo                          Ctrl+Z, Alt+Backspace



Redo	Ctrl+Y
Copy	Ctrl+C, Ctrl+Ins
Paste	Ctrl+V, Shift+Ins
Paste nomove	Ctrl+P
Paste position	Ctrl+Alt+P
Paste camera position	Ctrl+Alt+V
Grid up	Q
Grid down	Ctrl+Q
Texture lock	Ctrl+T
Create object	Insert, <
Select all	F3

## View

Workspace	Ctrl+F11
Node bar	E, Ctrl+F10
Button bar	Ctrl+F9
Redraw views	F5
Simulate world	Alt+R
Simulate physics	Alt+T
Simulate game	Ctrl+Alt+T
Lock Cam (remember pos)	Ctrl+Alt+Z
Lock Cam (no memory)	Ctrl+Alt+X
Increase timescale	Ctrl+Inc
Decrease timescale	Ctrl+Dec
Center all	Alt+C
Show structure	Ctrl+R
Show models	F9

## Selection

Step into	F
Step into sel	Ctrl+Alt+F
Step out of	Ctrl+F
Group	G
Ungroup	Ctrl+G
Flip X	Alt+X
Flip Y	Alt+Y
Flip Z	Alt+Z
Carve	Ctrl+X
Toggle Hidden	H
Toggle Type Hide	Ctrl+Alt+H
Copy surface	I
Paste surface	K
Copy surface	O
Paste surface Params	L
Force snap	F4

## **Tools**

Scale	Alt+1
Rotate	Alt+2
Edge	Alt+3
Vertex	Alt+4
Texture	Alt+5
Next tool	Space
Previous tool	Shift+Space

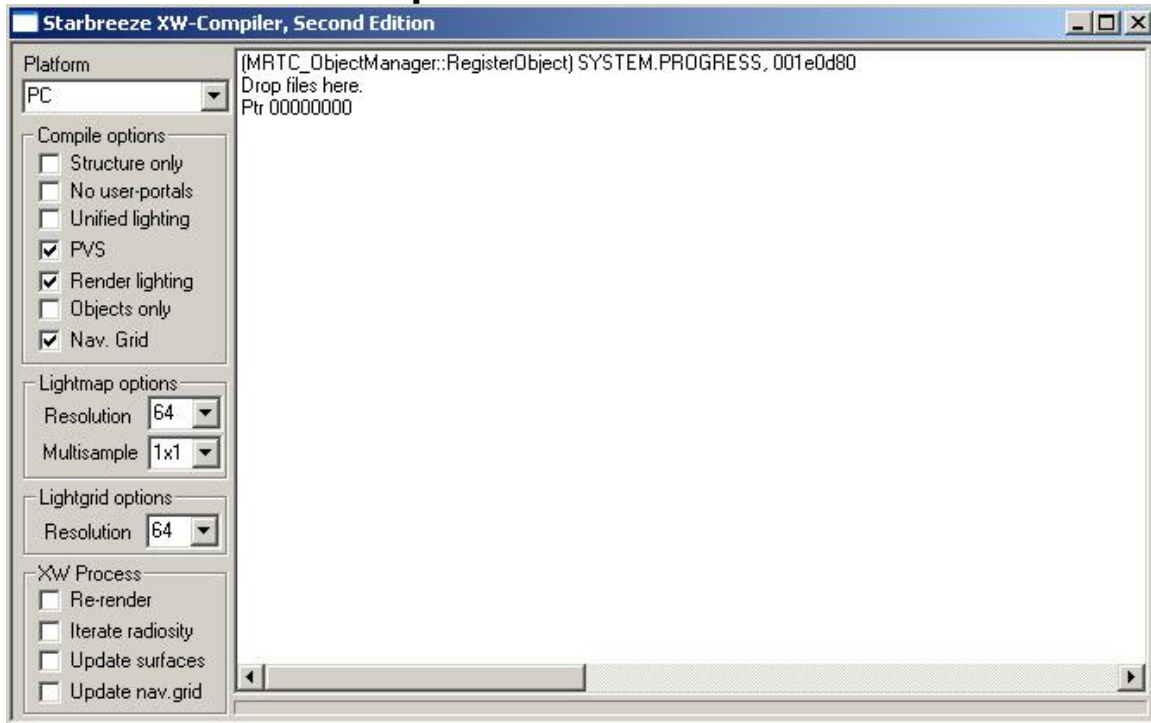
## **Window**

NVIDIA pleaser	F2
----------------	----

# Templates and Registries

**Todo**

# Starbreeze XW-Compiler



The location of your XW-Compiler may vary depending on where you install it and what project you're working on, but it should be in the same folder as the game by default.

Ex:

C:\Program Files\Starbreeze Studios\Knights of the Temple\ XWC\_Static.exe

Open up XWC\_Static.exe and then just drag and drop the .xmp map file into the window to compile them. But before you compile you might want to read though the Compiling options there you can find ways to speed up the compilation time and how to change the detail of your maps shadows etc.

The map itself should be found in the Worlds folder.

Ex:

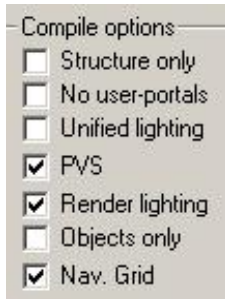
C:\Program Files\Starbreeze Studios\Knights of the Temple\Sbz1\Worlds

To test your map in game you need to bring down the console and give the command to run your creation. To bring down the console you need to press the "\$" or "~" depending on what keyboard layout you have. Once you have the console down type the command: "Map <name of your map>"

This should load the map if there were no errors during the compilation.

## Compile options

First you have to select for what platform you want to compile your map. Default platform is PC.



**Structure only:** Not in use.

**No user-portals:** This is mainly used to bug fix. If you have compilation problems and you suspect that there is something wrong with the user portals you can enable this option and compile the map without the user-portals.

**Unified lighting:** This is for the stencil shadows, should be checked when compiling maps for Riddick and later projects.

**PVS:** This will pre-render a Portal View System the engine use to reduce the network traffic for online (and offline) games. It is also used as simple portal culling for cases not handled by the normal visibility algorithms. This is recommended to be turned on at all times.

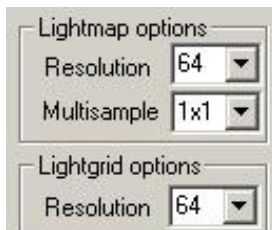
**Render lighting:** You can disable this for projects like KoTT and Enclave to get a full-bright version of your map this will speed up the compilation time a lot but you won't be able to see the light settings or shadows.

**Objects only:** This will only re-render the objects in your map the navigation grid and structure, brushes and lights will stay the same. This will reduce the compilation time drastically if you only change the models and objects in the map. The name of the .xmp file can't be change for this to work and requires a compiled .XW.

**Nav. Grid:** This is what the AI and chars use to navigate. It takes a good while for the compiler to create a navigation grid so if you're not in a stage where you don't have game play in your map yet there is no need to select this option. Structure, player phys and AI off limits are some of the brushes that block this grid.

## Lightmap and Light grid options

These three settings are only used for the Knights of the Temple and Enclave projects. The Lightmap and Lightgrid options can be used to change the resolution of the light in your maps can be used to

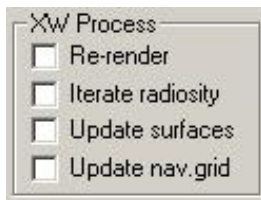


**Resolution:** This affects the shadows in your map. The lower the value is the better the shadows will look, but it will also take a lot longer to compile and use up more memory. This affects the resolution on all the light maps for the brushes in the map. Default is set to 64.

**Multisample:** This also affects the shadows in your map, but not as much as the resolution option does. This option won't affect the memory or the game speed, it will take a whole lot longer to compile though. It's recommended that you raise the multisample then you're compiling the final build for your map. Default is set to 1x1.

**Lightgrid options:** The light grid will affect the models in the world and how the light react on these, the higher resolution the better the light and models will interact with each other. This will affect the game speed as well, if you select a higher resolution the more the engine needs to render. Default is set to 64.

## XW Process



**Re-render:** This will only re-compile the light maps in the map, only used for KoTT and Enclave.

**Iterate radiosity:** This will turn on radiosity rendering that will be calculated after the lightmaps has been render. This feature is probably not fully functional.

**Update surfaces:** This will only update the surfaces of your .XW files.

**Update nav.grid:** Use this to update the navigation grid of your .XW files.

# Tutorials

## Ogier Tutorial: Creating Your First Map (Enclave).

Written by Shawn "AmranX Davison & Edited by Curtis "Methril" Hielema

The only reason that this tutorial is in any coherent form is probably because Methril did an editing job on it. Sorry if I kept repeating myself even if you understood the first time, as I was thinking more along the lines of "never mapped before" while I was working on this. If you have already worked on Quake or Half Life engine based games, then you probably don't really need this tutorial, and can start out by reading the FAQ post.

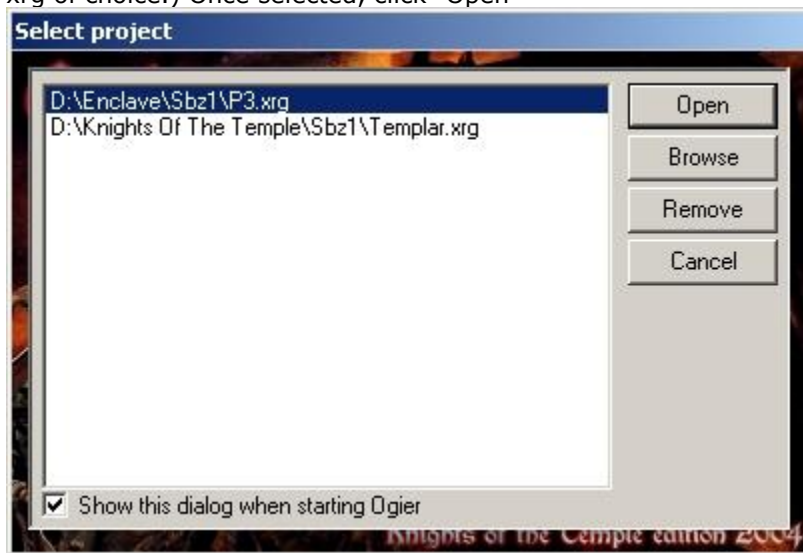
For this tutorial you will need the latest version of Ogier and the GDK from <http://ogier.starbreeze.com/> I HIGHLY suggest that you read through the FAQ topic on the Ogier forums, as it explains a great number of features. Keys and their uses, and so on. <http://forum.starbreeze.com/viewtopic.php?t=15>

The finished xmp file for the map can be downloaded at:  
<http://members.shaw.ca/AmranX/TestLevel.xmp>

Also, I will not mention saving until the end of this tutorial, but I do suggest you save after every few steps just in case.

### ----Setup----

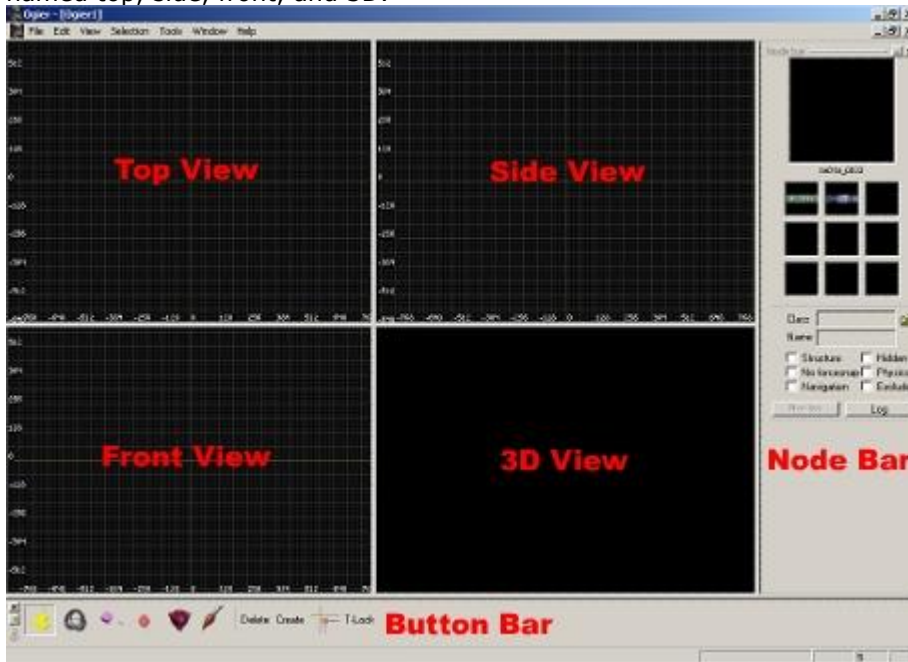
When you first open up Ogier, a window entitled "Select Project" will pop up. You must select the xrg file located in the games Sbz1 directory for the game you wish to work on, which is in this case Enclave. (If this is the first time running Ogier, click "Browse" and navigate to the xrg of choice.) Once selected, click "Open"



Ogier's main window will now open onto a flat gray screen with three menus at the top. In this tutorial we are creating a new map, so we will select "File" and then "New Map". (Ctrl-N for short) The editor will load the resources and then bring up a new window containing all of the tools we need to create our level.



There will be four view windows covering the majority of the screen. These windows are named top, side, front, and 3D.



Also on the screen are a number of menus and other items. Two of the most important being:

-Located on the right side of the screen is the Node bar. This area contains the most recent textures that we have been using, as well as the properties of the brushes and entities we have selected. If the window titled "Node bar" isn't showing up, hit "E" to toggle it on and off.

-Running along the bottom there are buttons which are used for editing the brushes, curves, textures, etc of the map.



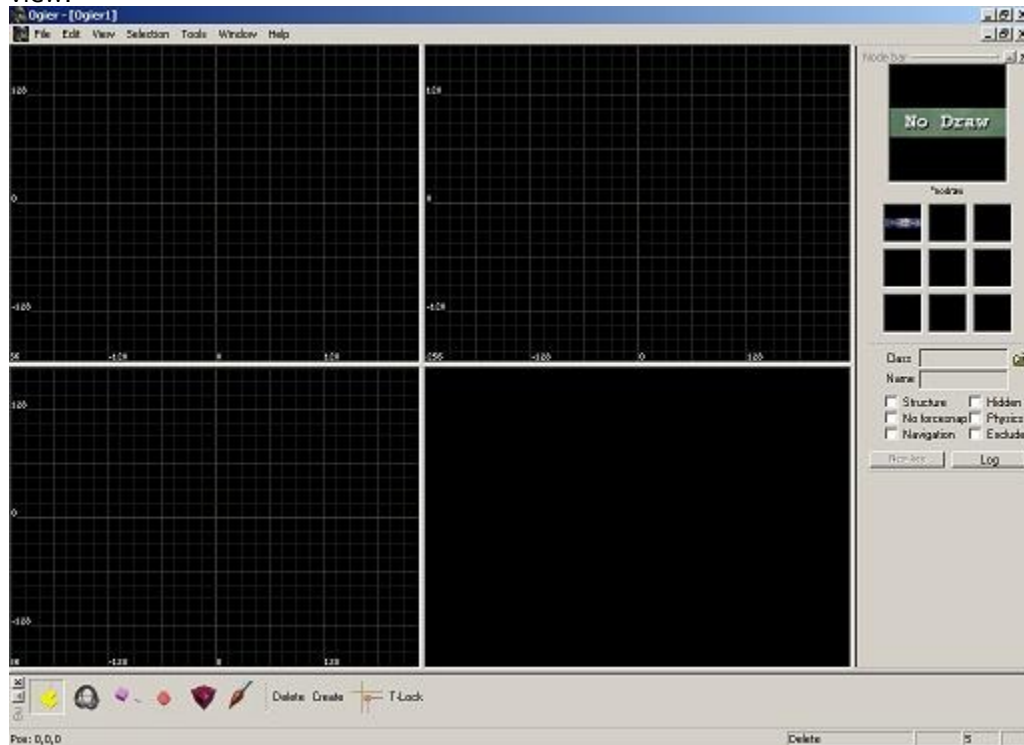
1. Resize/Scale
2. Rotate
3. Bend (For use with curves/splines)
4. Vertex Editing (This is used to edit the corners of a shape or spline)
5. Texturing Mode
- 6?
7. Delete selected
- 8?
- 9?
10. Texture Lock

Some basic navigation info: To move around the three 2D views (top/front/side), hold the shift button while the cursor is over the view and move the mouse. Also, in the top/front/side views, use the +/- keys or the mouse wheel to zoom in and out. In the 3D view hold shift and then either move the mouse to look, or use the W,S,A,D keys to move around.



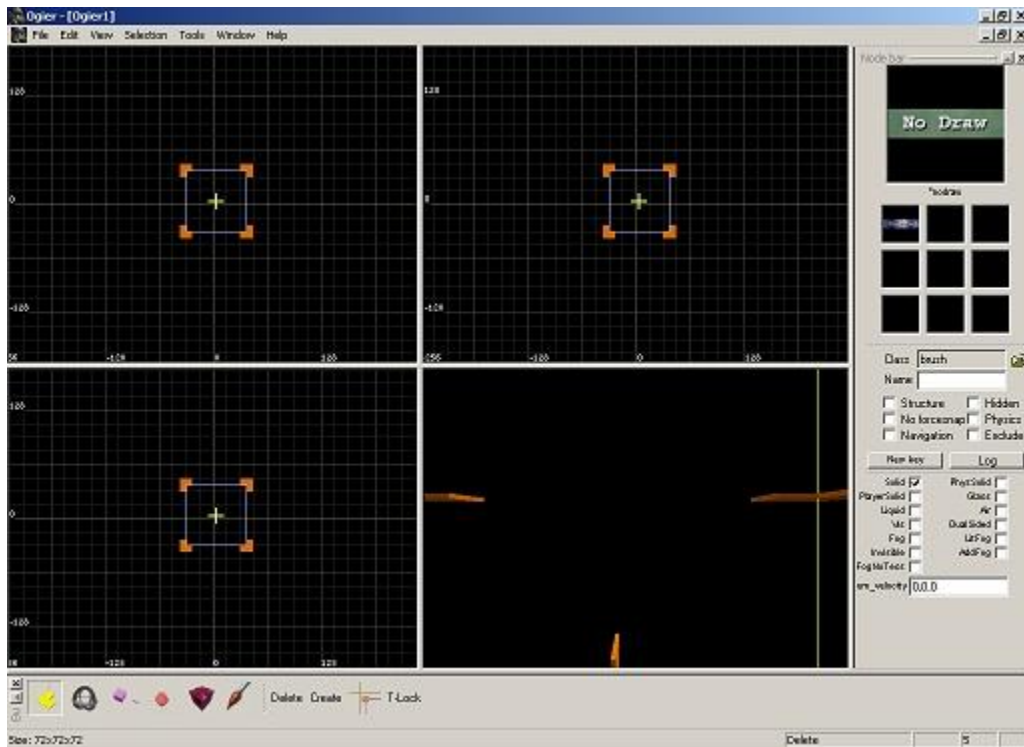
## ----Zoom----

Let's get started by zooming our 2D views in closer. Place your cursor over the center of the top view and press the + button three times. Repeat this step with both the front and side view.

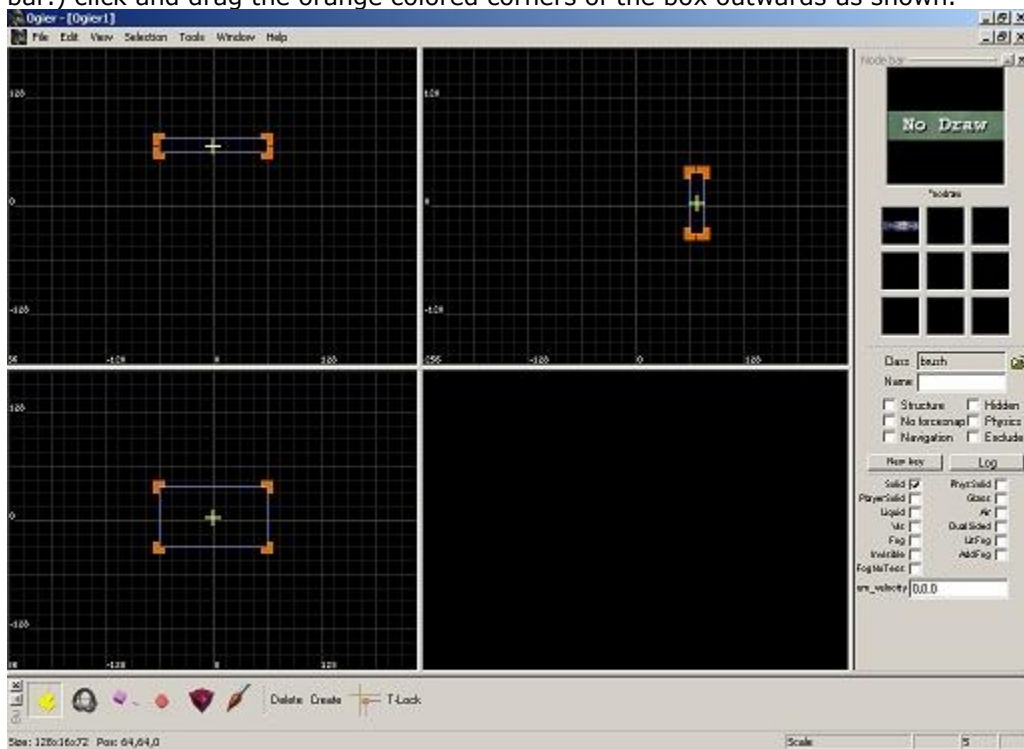


## ----Creating and Resizing the First Brush----

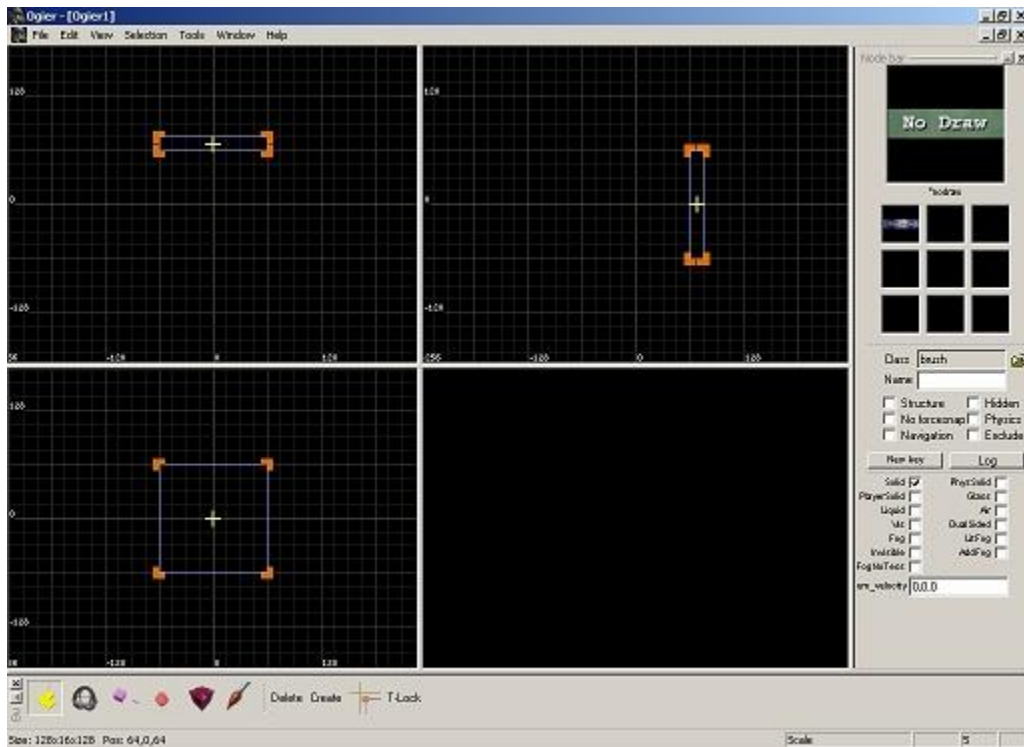
Now, let's make our first brush. Hit the "1" key to create a box at the cursor's location. (The number keys create brushes, while holding shift and pressing a number key creates a curved surface or spline.)



Making sure the scale tool is selected (The yellow L shaped icon on the left side of the Button bar.) click and drag the orange colored corners of the box outwards as shown.

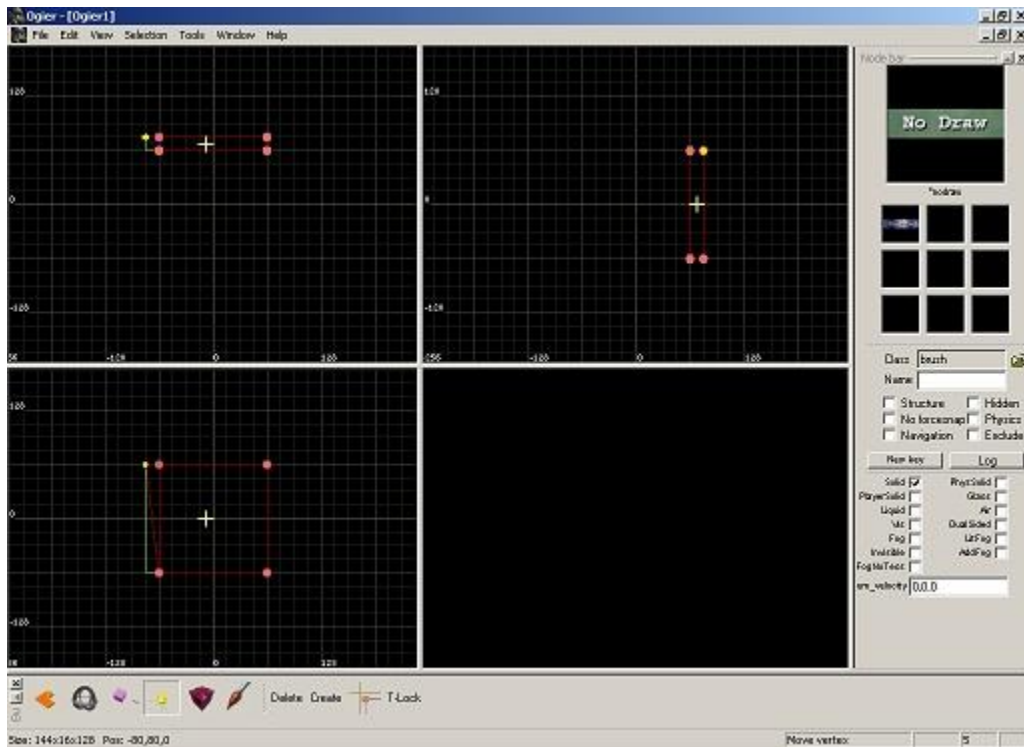


Now, click and drag the corners of the box in the front (or side) view to resize it as shown below.



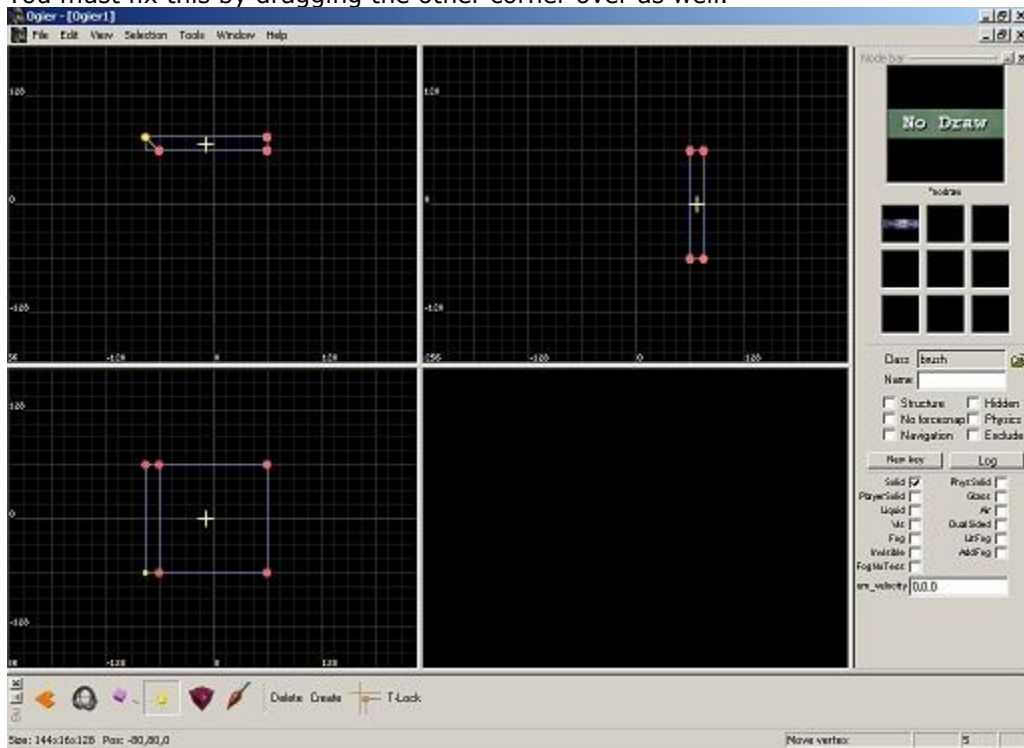
## ----Vertex Editing----

All right, now let's take a look at some vertex editing. Click the vertex editing button in the Button bar (the sphere with three grey lines extending out from it) then click on one of the pink corners in the top view and extend it outwards one grid spacing.

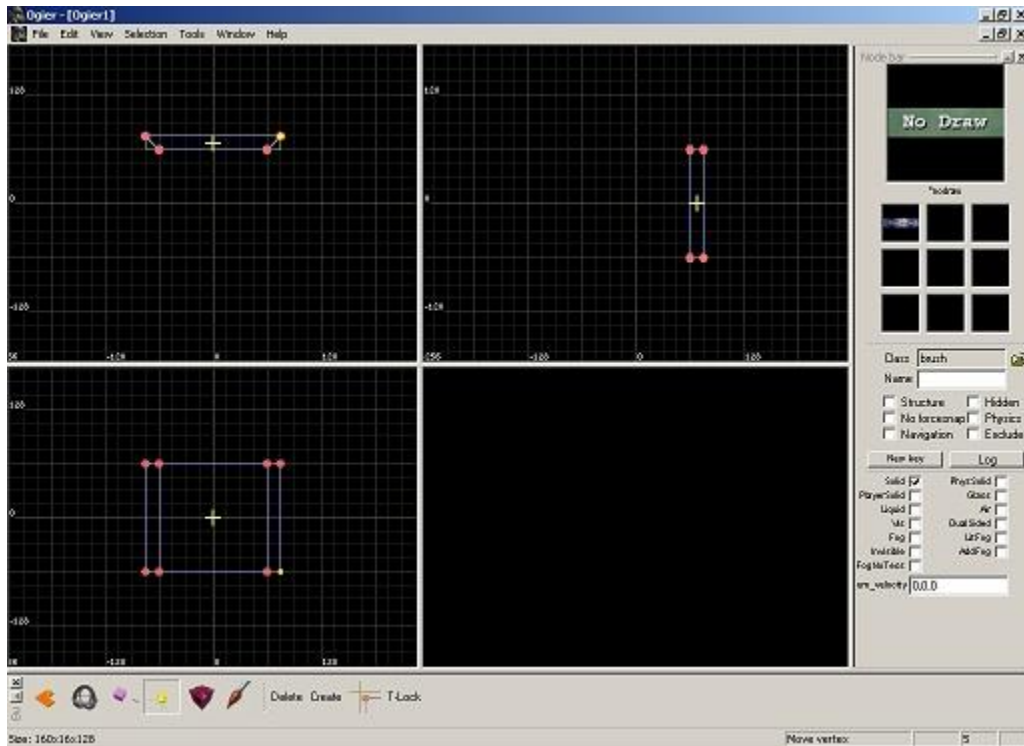


As you can see, the brush has now turned red due to the way one vertex is projecting out. (It will either not compile, or will compile but have problems. I am unsure how the Enclave engine handles this.)

You must fix this by dragging the other corner over as well.

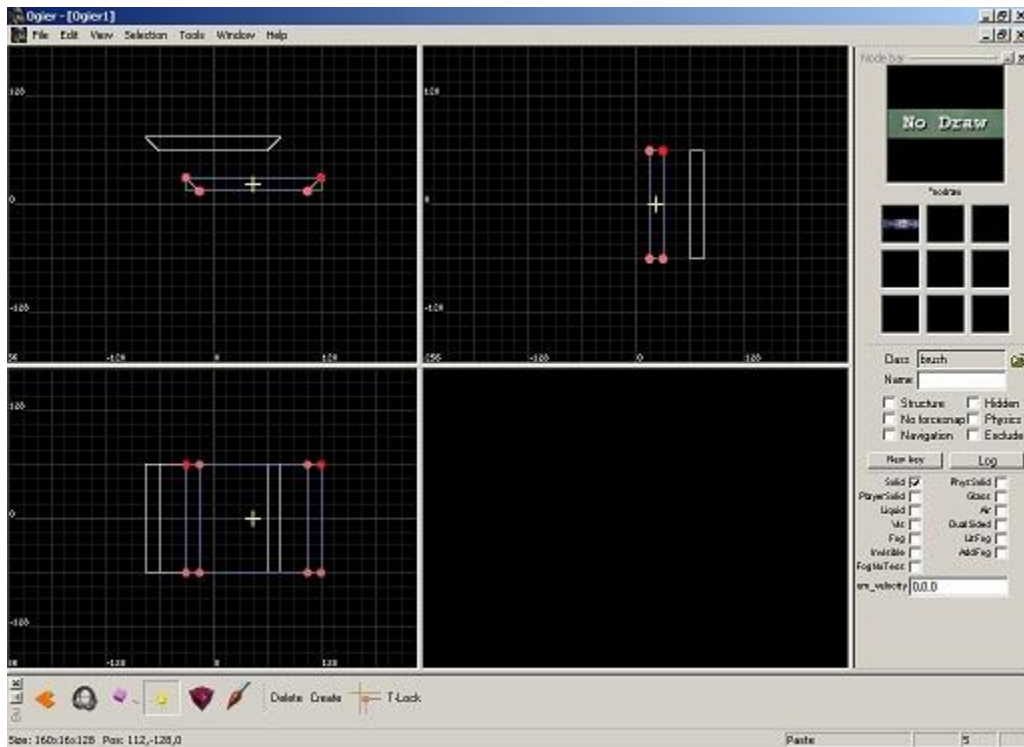


Much better, now repeat these steps for the other side of the brush.

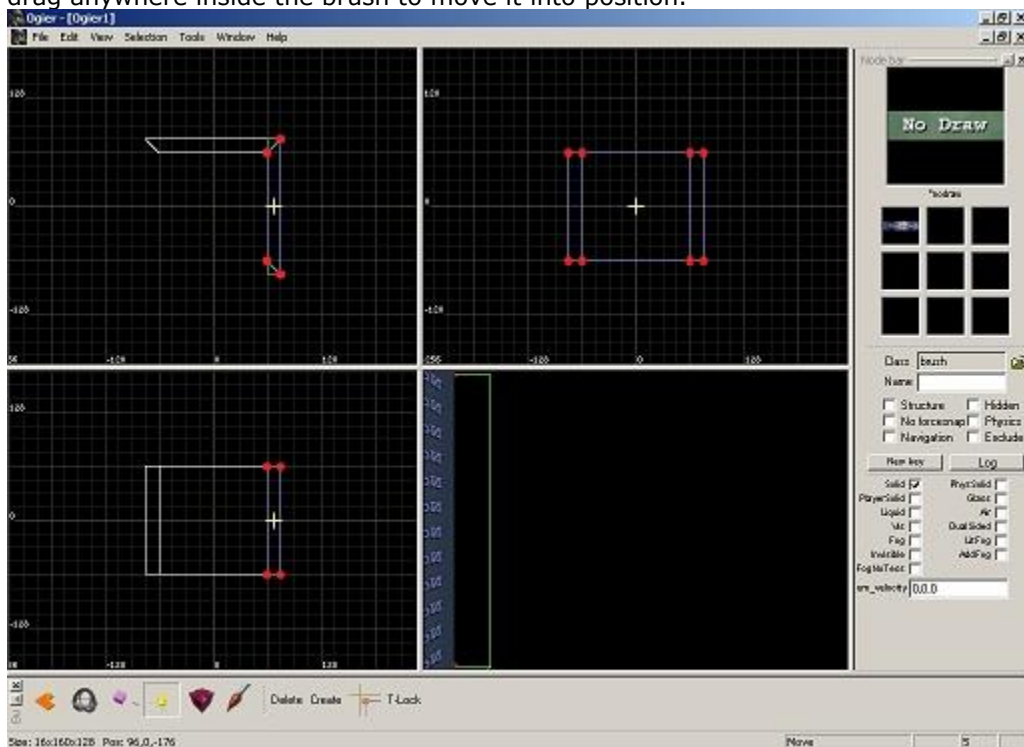


## ----Duplication----

Next, we will duplicate this brush to make the walls of our room. With the brush still selected, hit Ctrl-C then Ctrl-V to create another copy of it.

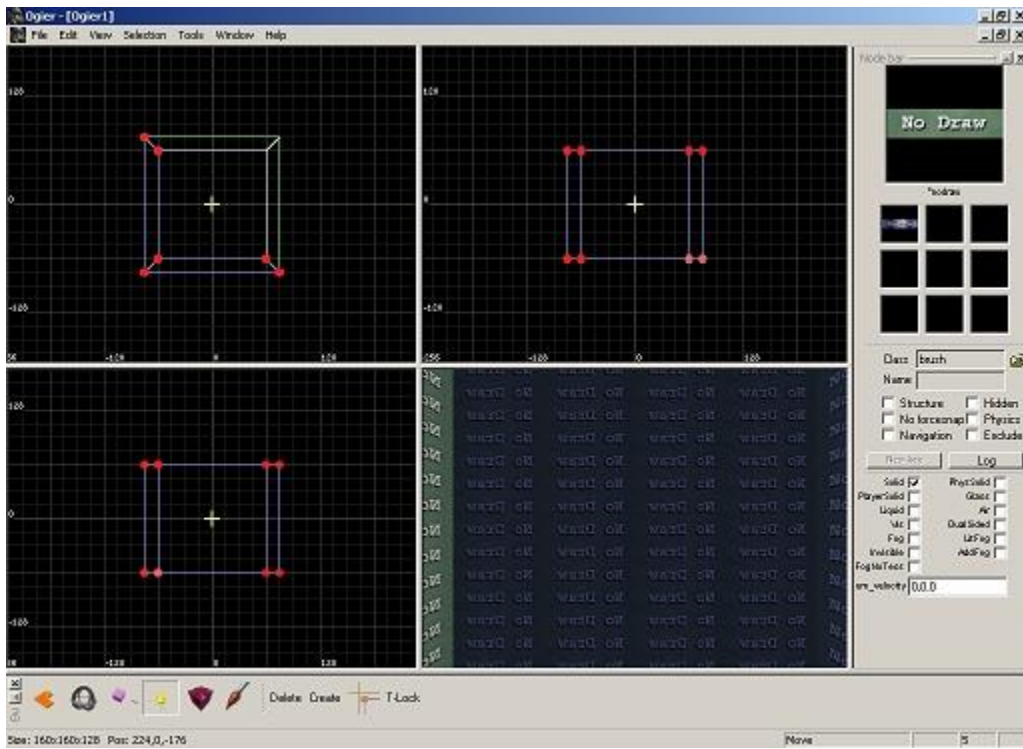


With the mouse cursor over the top view, tap the "V" key twice to rotate it 90' then click and drag anywhere inside the brush to move it into position.

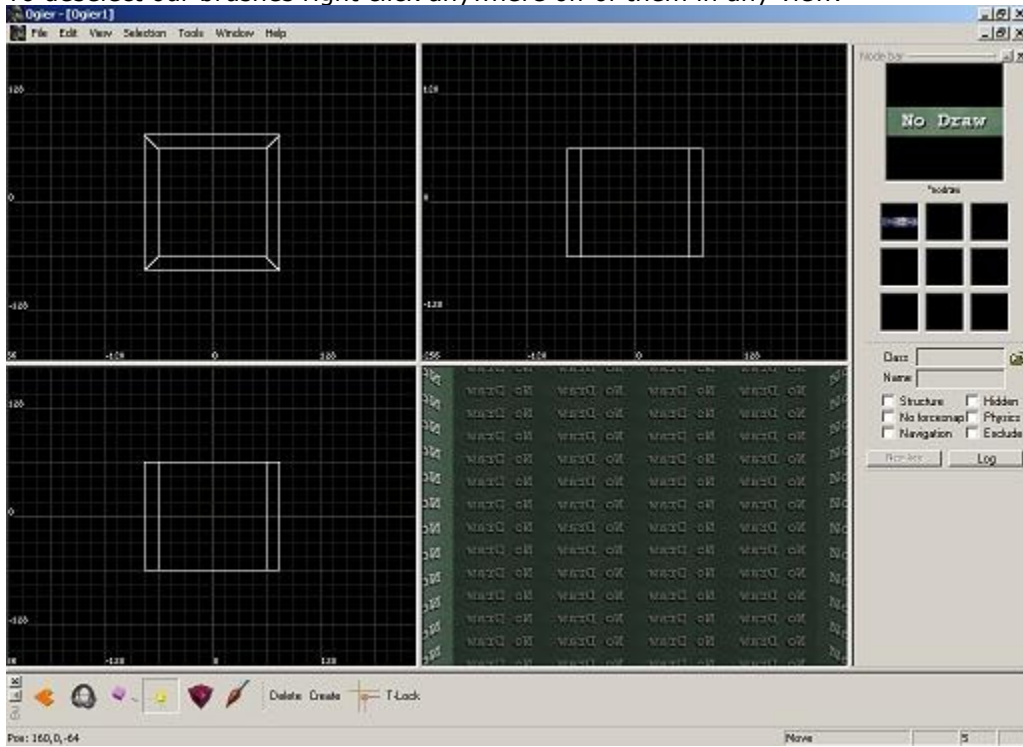


Holding down Ctrl, click on the original brush. This will select both at the same time. Now do another Ctrl-C Ctrl-V and with the cursor over the top view tap "V" four times. Move these two duplicated sections into place like below.



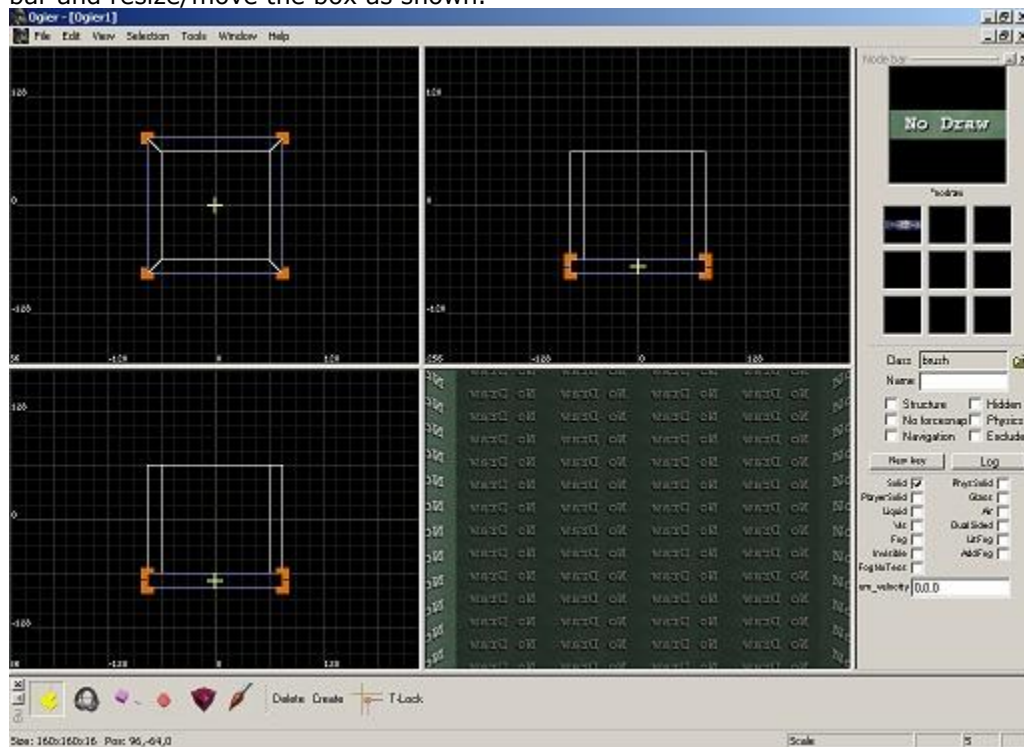


To deselect our brushes right click anywhere off of them in any view.

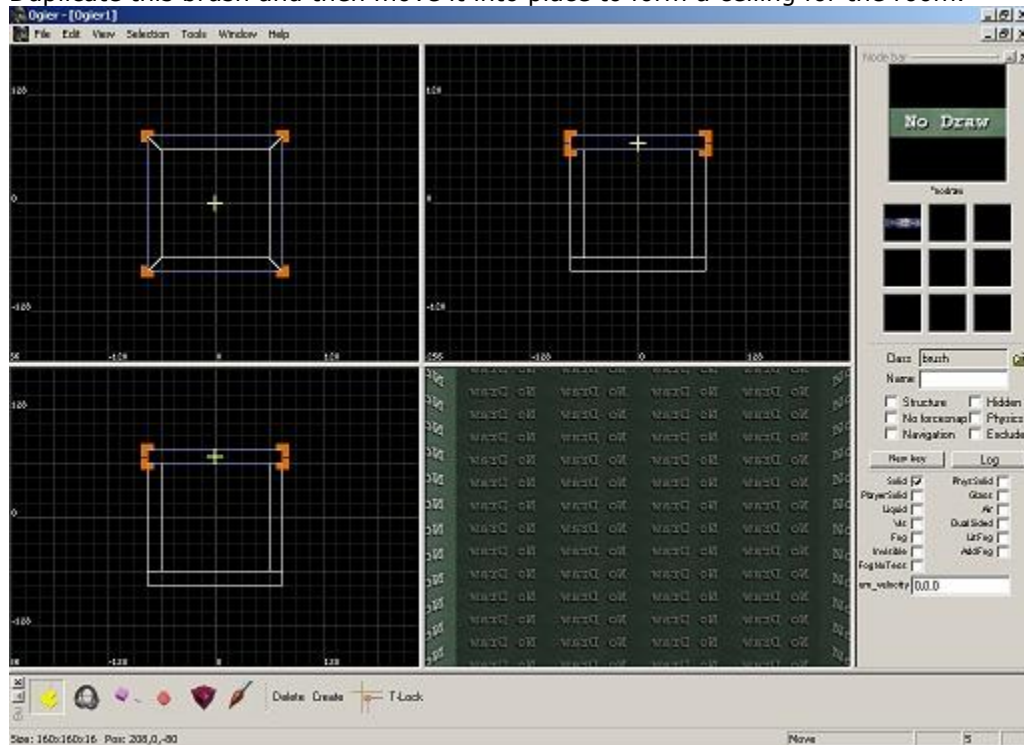


----Floor/Cieling----

Let's now add a floor and ceiling to our room now shall we? Hold the cursor over a 2D view window again and hit the "1" key to create another box. Re select the resize tool in the Button bar and resize/move the box as shown.



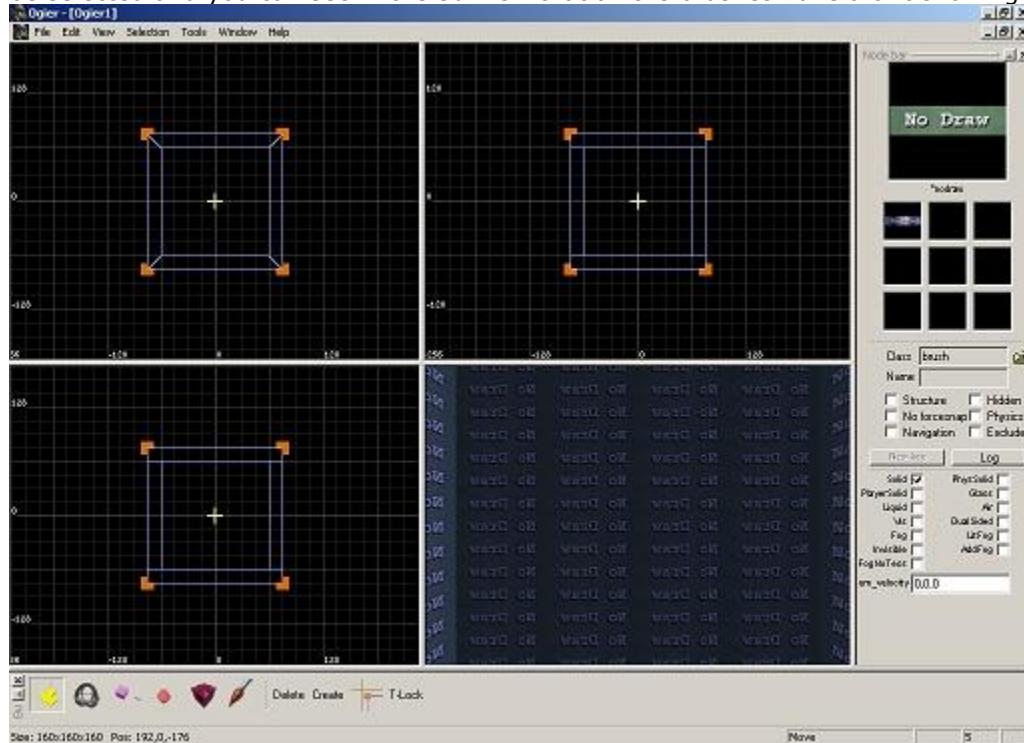
Duplicate this brush and then move it into place to form a ceiling for the room.



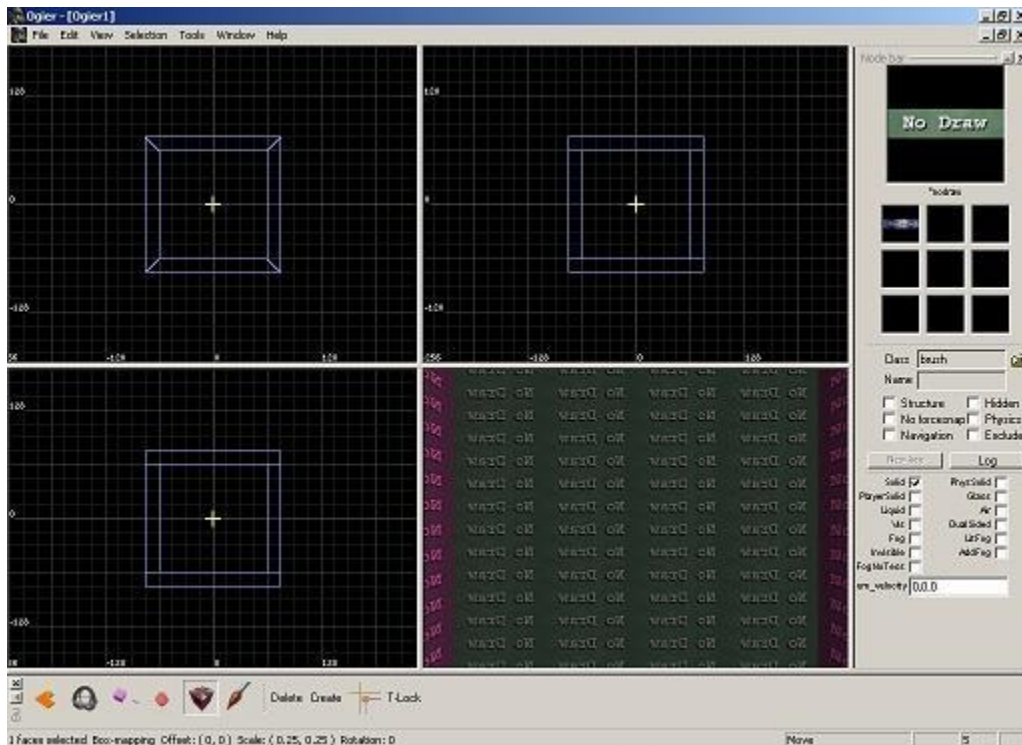


## ----Texturing the Room----

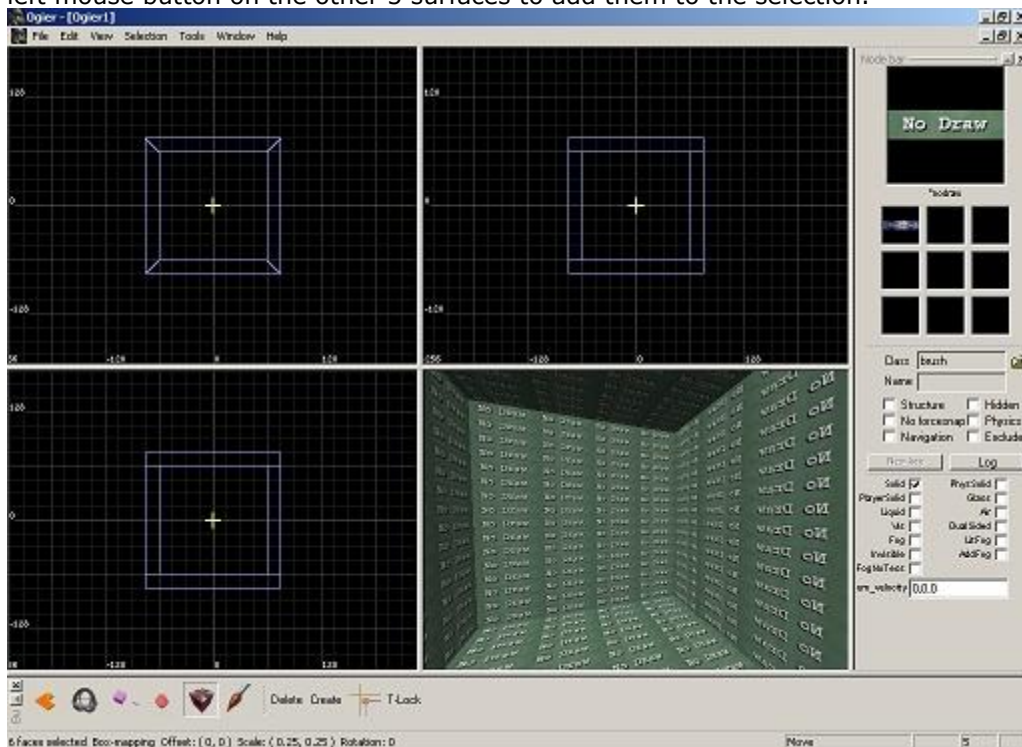
And now we have a simple room made with some of the basic tools. Let's add a few textures. First, right click off your ceiling to deselect it, then holding the right mouse button, drag a selection box over your entire room and release the mouse button. Now your whole room will be selected and you can see in the 3d view that all the brushes have a tint showing this.



Click on the texture tool in the Button bar (the dark purple box) and you will see the brushes in the 3d view are now purple. Click once with the left mouse on the surface in front of the 3D views camera and it will return to having no tint. In texturing mode this means that that surface is now selected.



Place the cursor over the 3D view and hold shift. Using the mouse and W,S,A,D look and move around the room so that you can see the other surfaces. Now, while holding Ctrl, click with the left mouse button on the other 5 surfaces to add them to the selection.



We now have all of the surfaces on the inside of our room selected for texturing. You'll notice how they currently all have the NoDraw texture applied by default which means that the engine will not render them, and they will appear invisible in game. Let's find a texture and

change that.

Place the cursor over top of any of the 4 view windows and hit the ~ key to bring up the texture browser. The various texture collections are in groups listed on the tabs on the top, and the textures themselves are displayed in the window below. Lets use DM3 for now. Click on the DM3 tab and you will see the textures for that group.



Click once with the left mouse button on the third texture shown (DM03\_01\_02) to select it causing the texture browser to close and this texture to appear up in the Node bar's currently selected texture window.



To apply the texture to all selected surfaces, tap the "K" key once. You should now see the inside of your room with the brick texture applied to it.



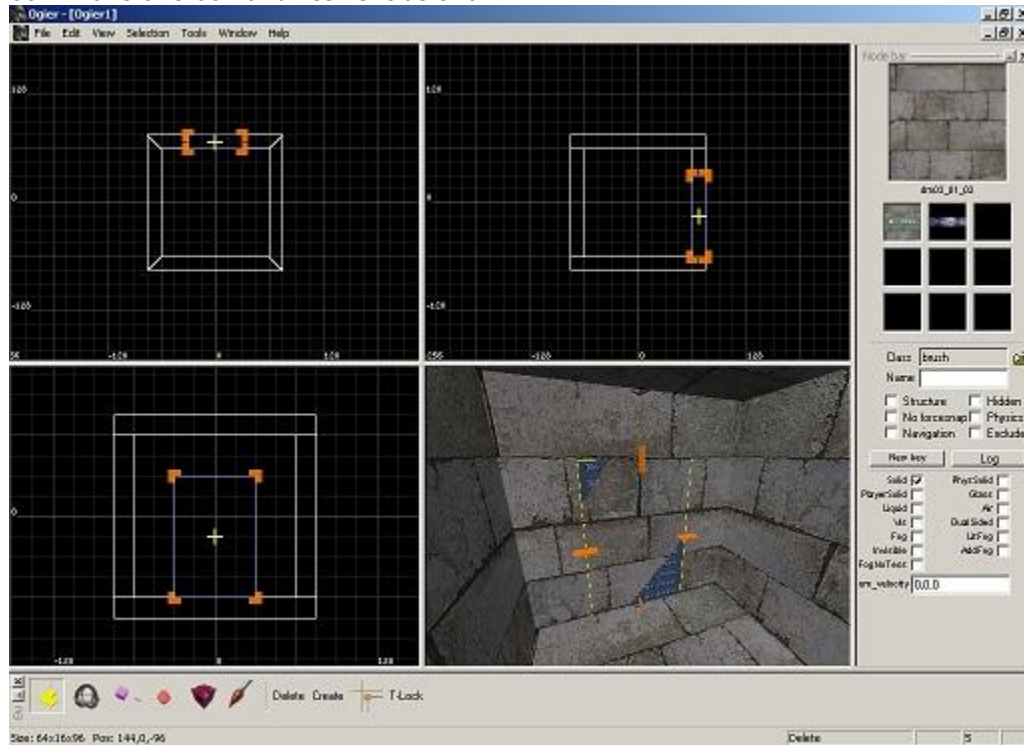
Right click anywhere off your room to deselect it, and move around in your 3d view to see that the texture was only applied to what you selected, and not the outside or edges of the room where the player will not be able to see.



## -----Adding Some Detail: Carve and Grid Size-----

Time to add a little detail and demonstrate some more features. Lets add an alcove with an arch above to the upper most wall.

Select the resize tool, and move the cursor over a 2D window then hit "1" to create another box. Move this box and resize it as shown.

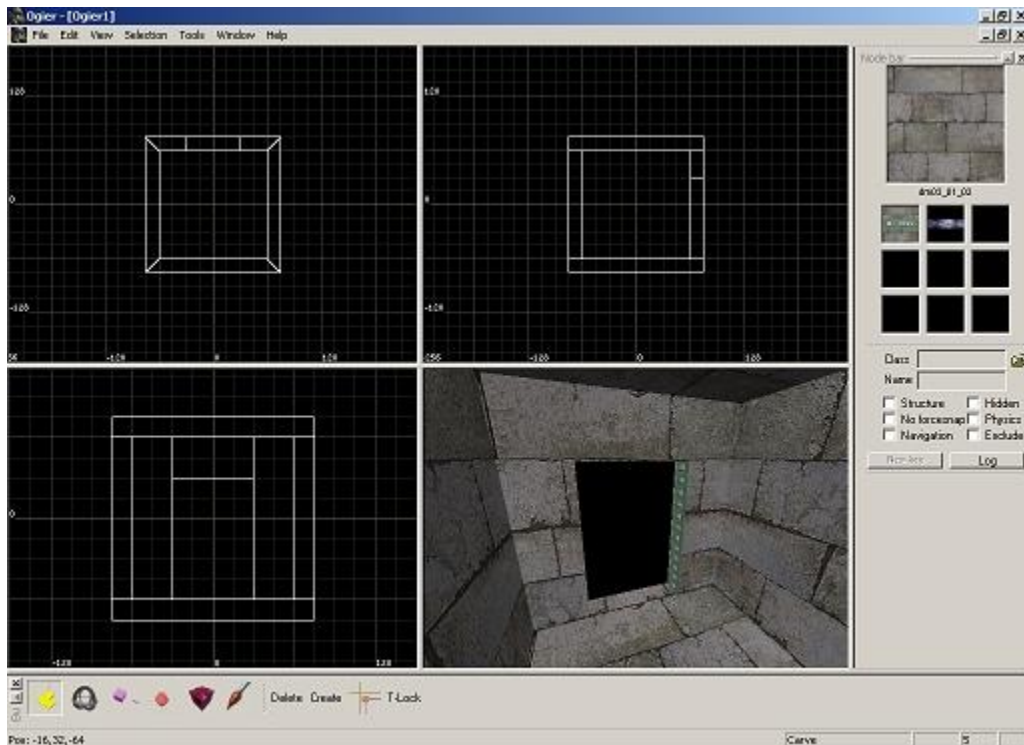


At the top of the screen, click on the selection menu, then click "Carve" (Ctrl-X)



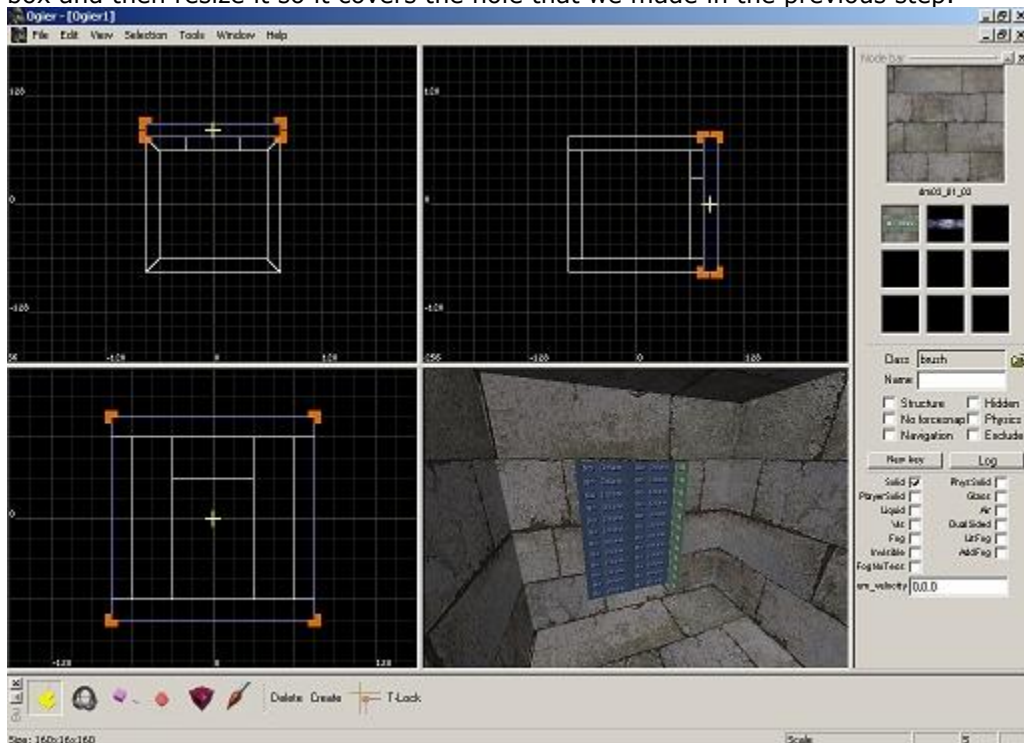
The box will disappear, and split up the wall leaving a doorway.



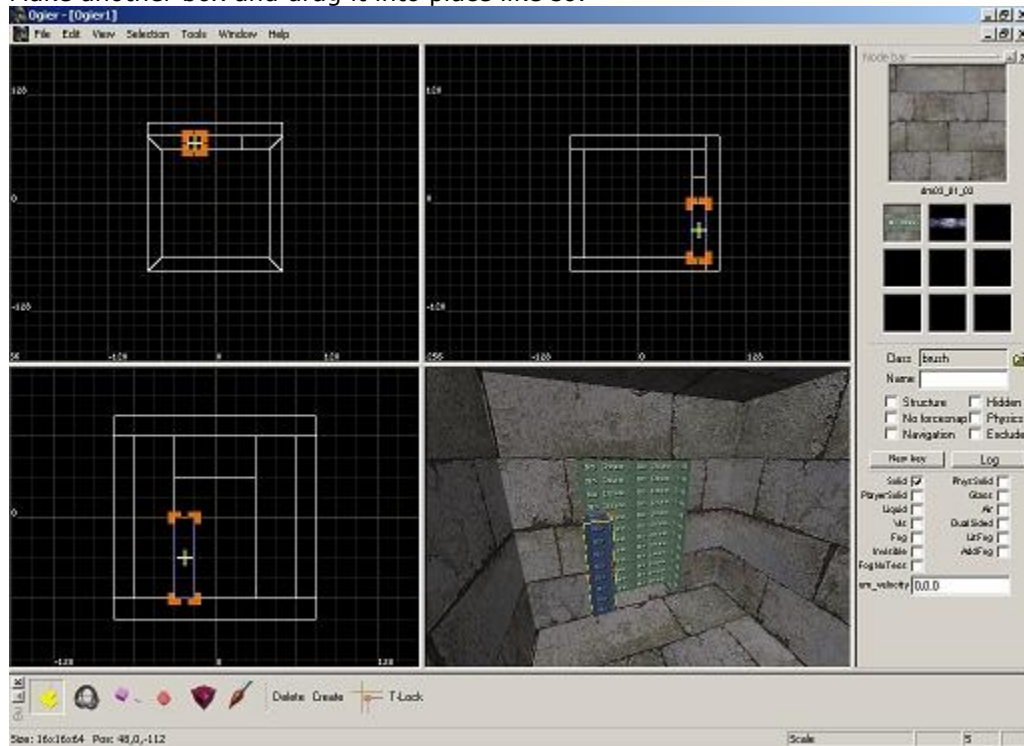


(\*Note\* Personally I don't use the carve function for creating doorways and the like, I prefer to create them "by hand" to be sure there are not any unnecessary brushes created in the splitting of the surface. The reason why I use it in this tutorial is mainly to show different features of Ogier, and ways of accomplishing things.)

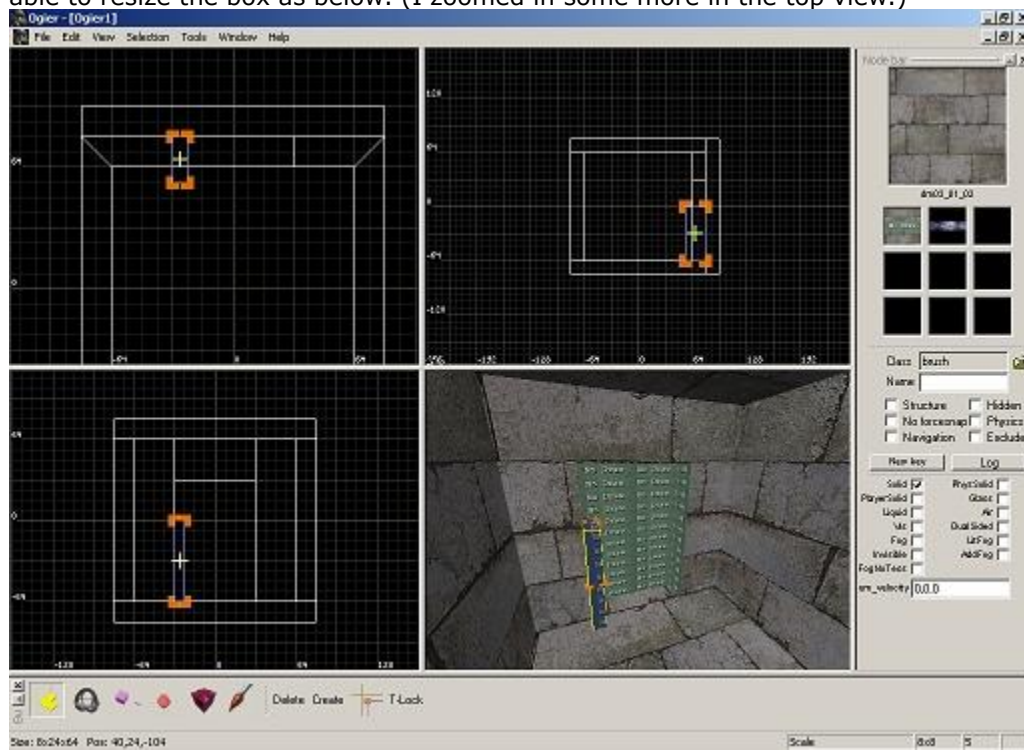
Let's block off the back of this hole with another box. Repeat the usual step of creating a new box and then resize it so it covers the hole that we made in the previous step.



Make another box and drag it into place like so.



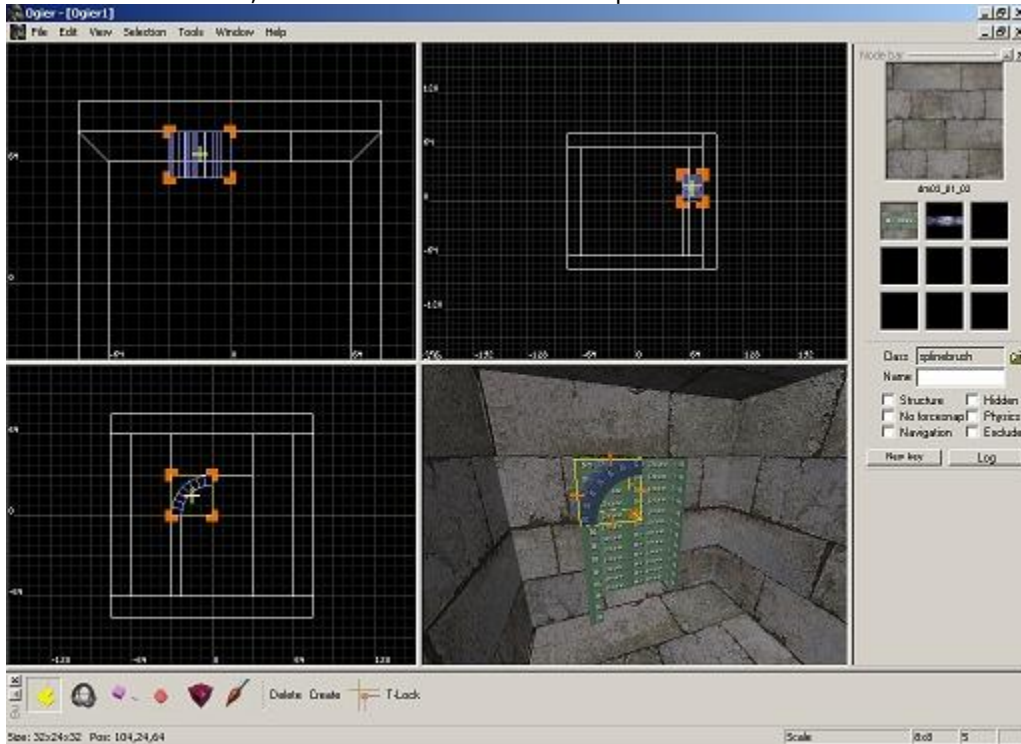
Now we want to adjust the size of this a little more finely, but we can't with the current grid. To decrease the grid size hit " Q " once. You will see this change immediately, and are then able to resize the box as below. (I zoomed in some more in the top view.)



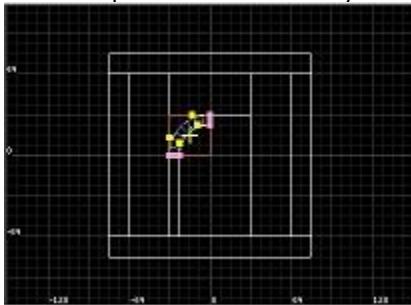
## ----Adding Some Detail: Splines----

The next step is to make half of the arch that will go above this.

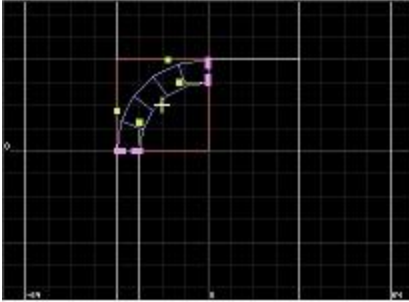
Put the cursor over a 2D view and hold shift then hit "4". This creates a type of curve or spline for us to use as the top of our arch. With the cursor over the top view, hit "V" twice to rotate it the correct direction, then resize and move it into place.



For some fun, lets decrease the number of sides in this spline. With it still selected, click the bend tool. (the purple box with a line in the bottom right corner.) You will now see a series of points appear allowing you to modify this spline. Hold the right mouse button and drag a selection box over top of the four points within the curve in the front view to select them. (The selected points will become yellow.)



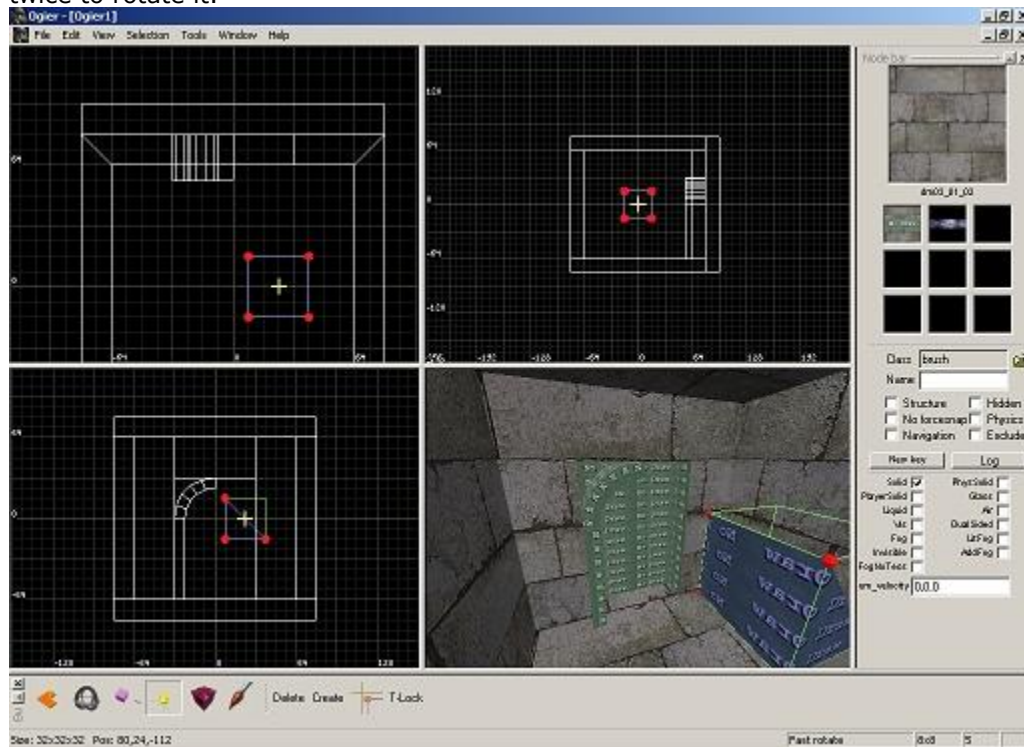
Hit the KEYPAD + sign to change the number of sides in the spline. We'll hit it 5 times, giving the spline 5 sides.



Right click off of it to deselect.

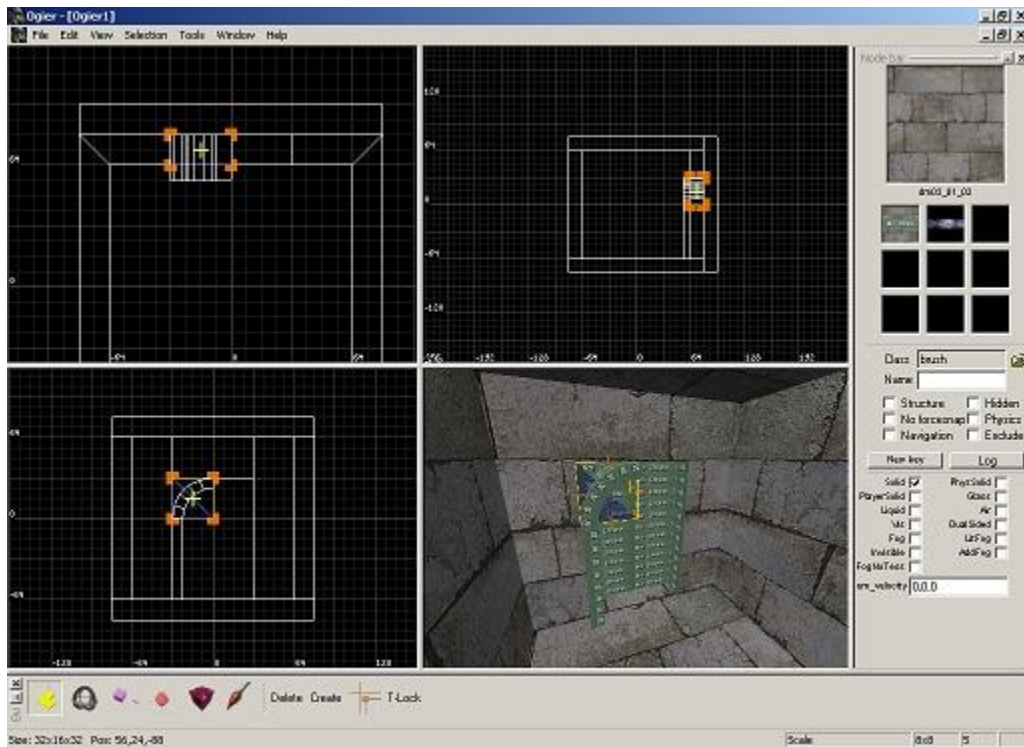
## ----Adding Some Detail: Filling in the Gaps----

We will need to add some wedges to fill in the gap between the arch and wall. Put the cursor over a 2D view and hit "2" to create a wedge. Hold the cursor over the top view and tap "V" twice to rotate it.

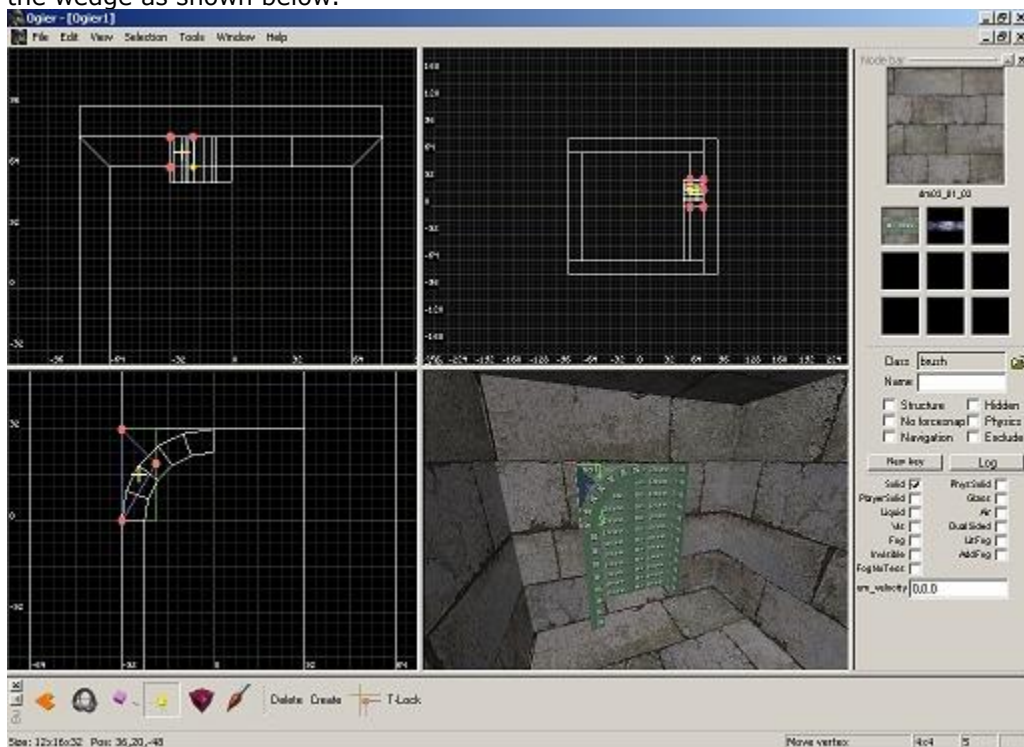


Click on the resize tool, and place it like so.

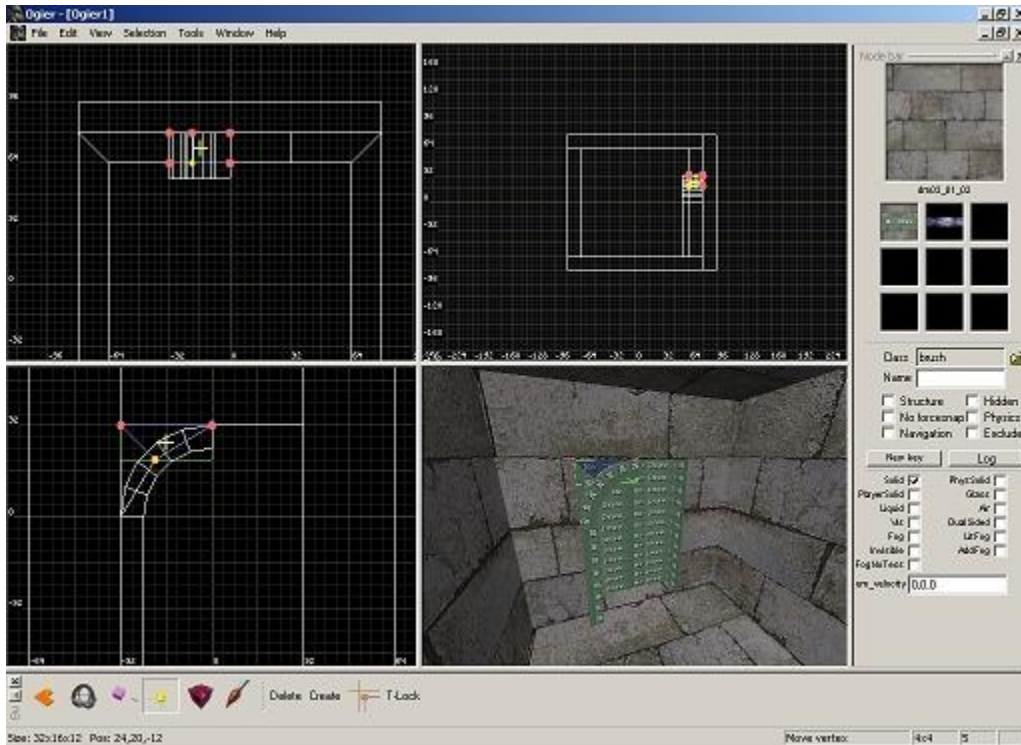




Click the vertex editing tool, hit "Q" again to decrease the grid size, and edit the vertices of the wedge as shown below.



Duplicate it with a copy and paste and again, vertex edit like so.



Right click off of the shape to deselect.

## ----Texturing Additions: Alignment----

Now let's texture these additions. Select the two wedges and the wall placed to block the back off (Ctrl click), then click on the texture mode icon.



Left click the surface of the back wall facing towards your room, then Ctrl left click on the two faces of the wedges facing towards the room so that all three surfaces are selected. As the brick texture we want is still selected, hit "K".

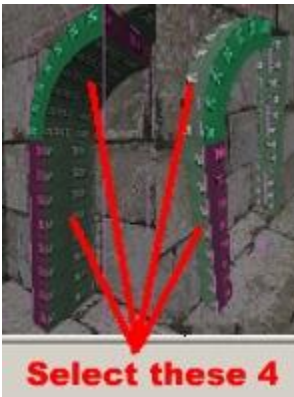


Right click to the side to deselect.

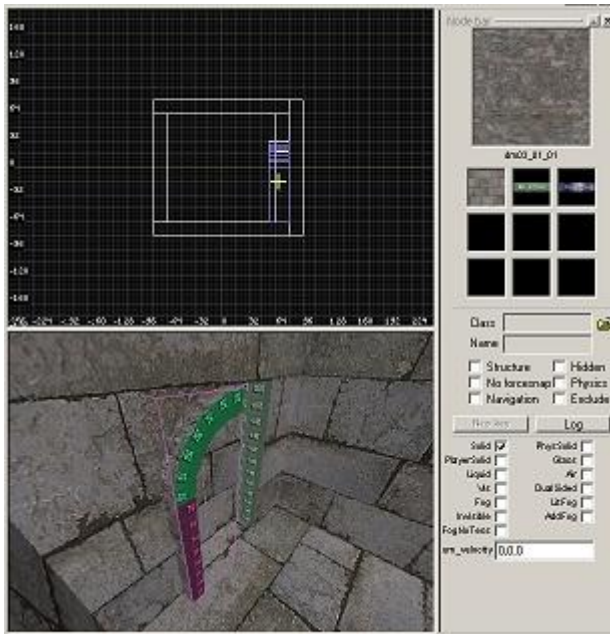
Next, select the side and top pieces of the arch. (Left click the side piece, Ctrl left click the top piece)



Adjust your 3D view and left click on the insides of the side piece to select the surface, then Ctrl click the other three surfaces shown:

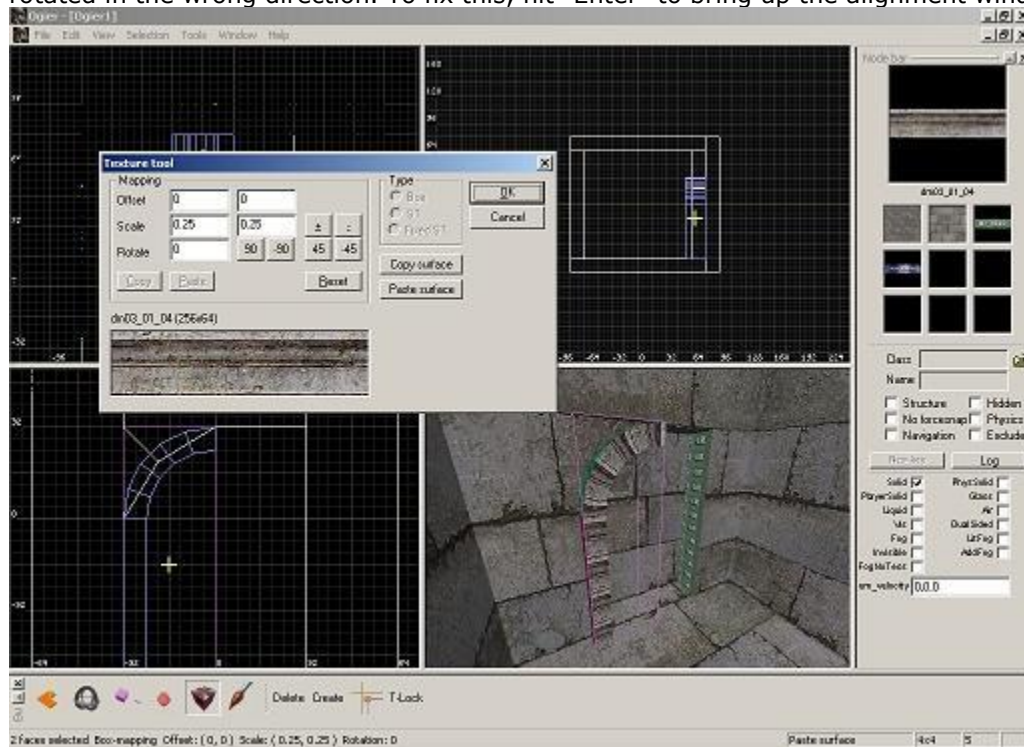


Put the cursor over a view and hit "~" to open the texture browser again. Click on the texture DM3\_01\_01. The texture browser will close like before, and the texture will be added to the Node bar. Hit "K" to apply this texture to the four selected surfaces.



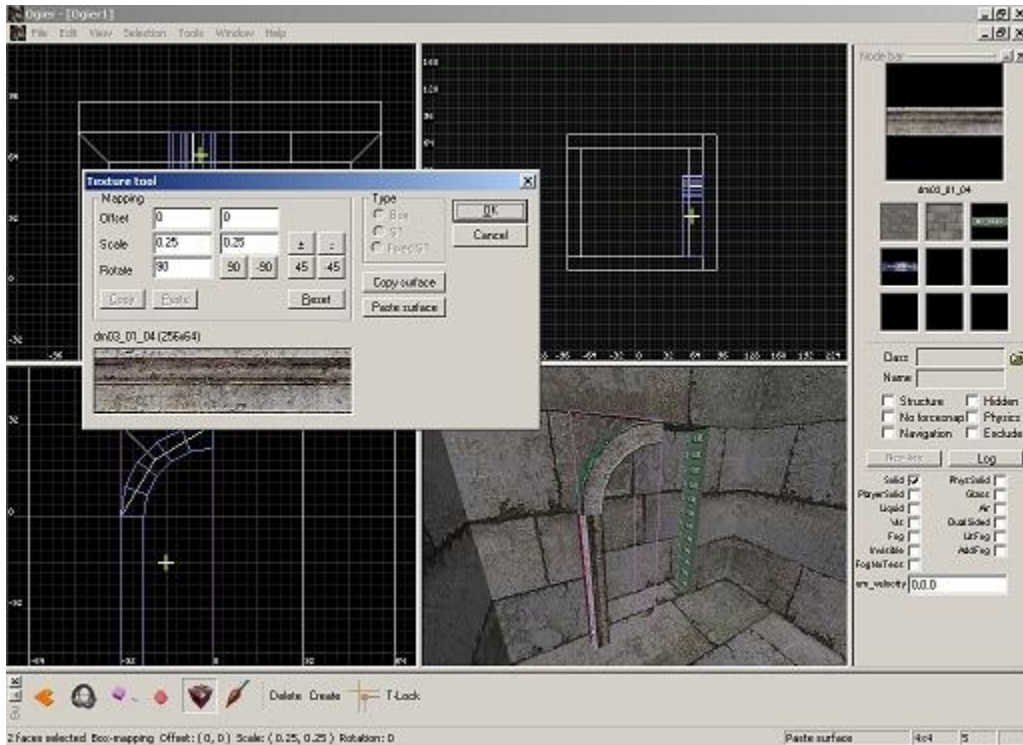
Left click on the front of the side piece and then Ctrl left click on the front of the top piece. The other four surfaces will be automatically deselected, and you will now only have two surfaces selected. Reopen the texture browser, and left click on DM03\_01\_04. Hit "K"

The way this texture will apply won't look very good as you will see. This is because it is rotated in the wrong direction. To fix this, hit "Enter" to bring up the alignment window.



To the right of "Rotate", click "90" once.





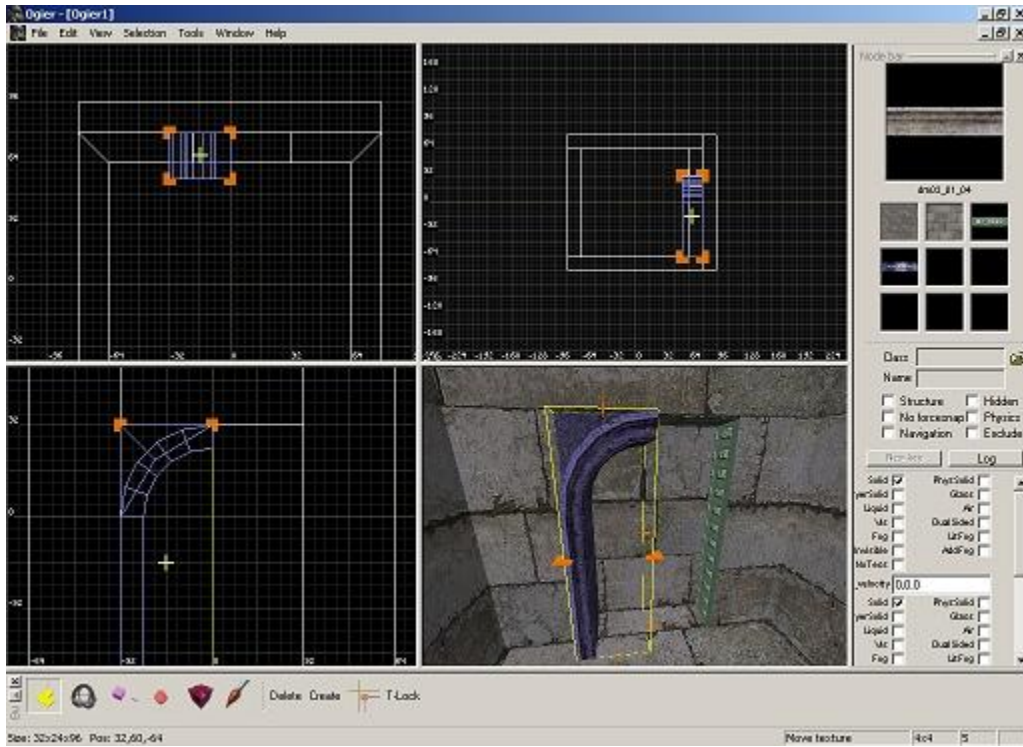
Well with that done, it looks better, but they still are not aligned properly. Let's adjust the arches texture some more. Hit "Ok" on the alignment window, and Ctrl left click the texture on the lower part of the arch to deselect it leaving only the texture on the curved part selected. Left click on the selected surface and holding the mouse button down, drag it to move the texture and align it with the other.



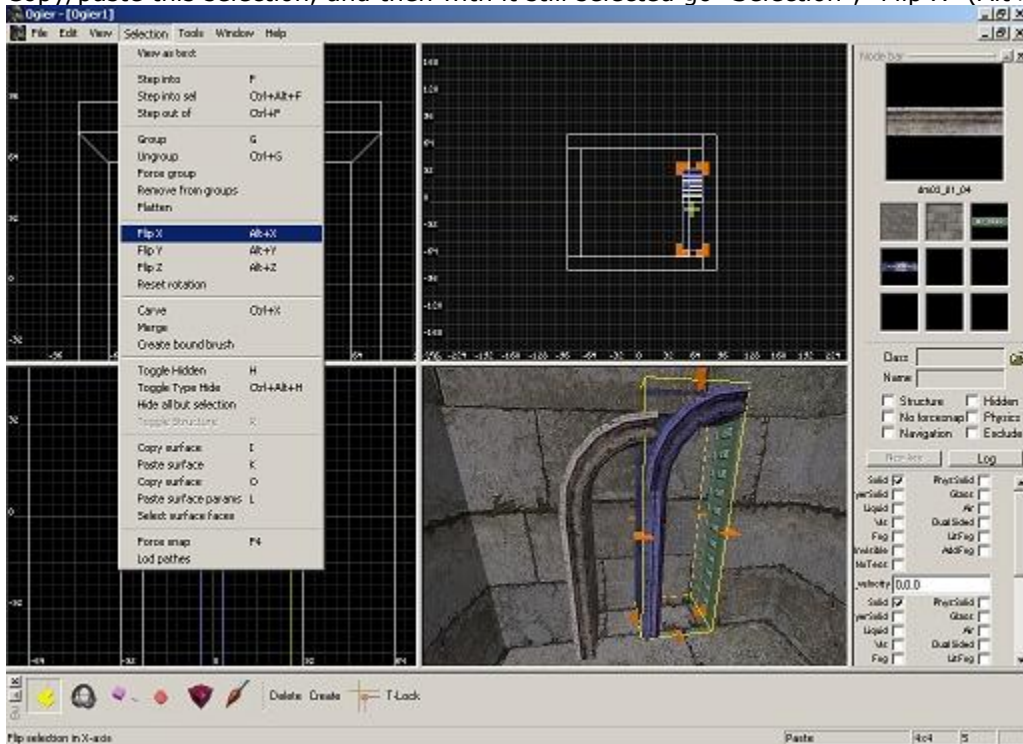
right click off of the brush to deselect.

## ----Texturing Additions: Duplication and Alignment----

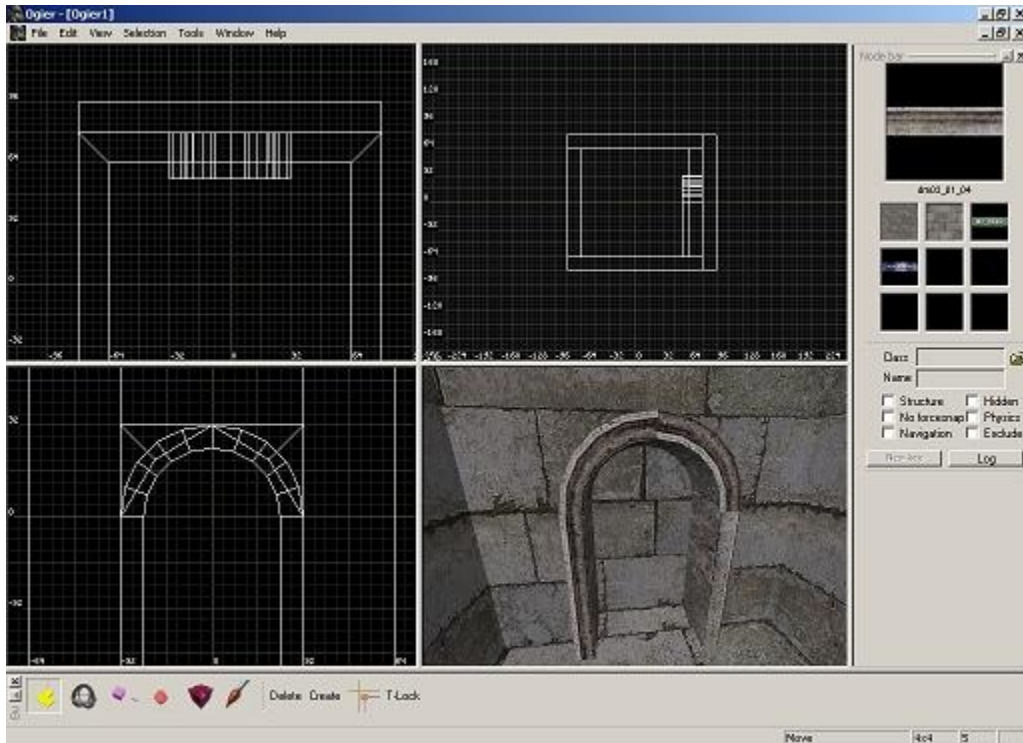
We will now duplicate the half arch we have created and flip it to create a whole one. Click on the resize tool and then select all the pieces of the arch we have made so far.



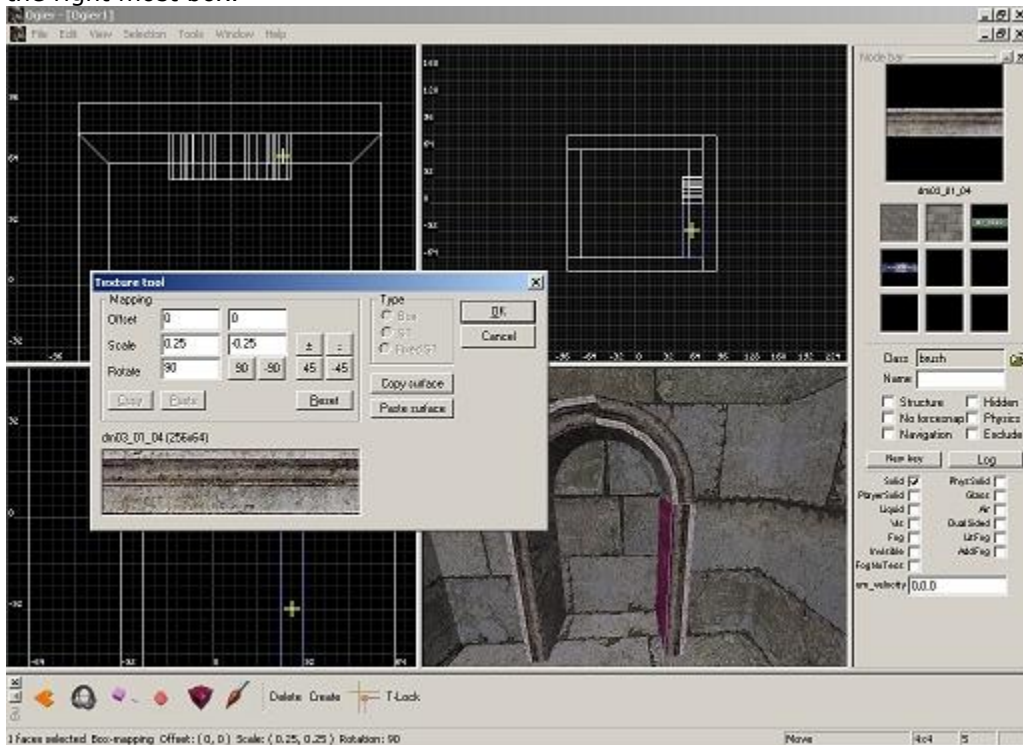
Copy/paste this selection, and then with it still selected go "Selection", "Flip X" (Alt+X)



Move it into position so it lines up with the first. Right click off the brushes to deselect.



Select the side part of the right half of the arch and then enter texture mode. Click the misaligned texture on the front and hit "Enter". To the right side of "Scale" there are two fields with the current scale of the texture. To flip this texture, we must add a - before the 0.25 in the right most box.



That's now fixed, so lets move on to the curved section. Hit OK and right click off of the brush, then click the curved section, select the face with the misaligned texture. This time around lets



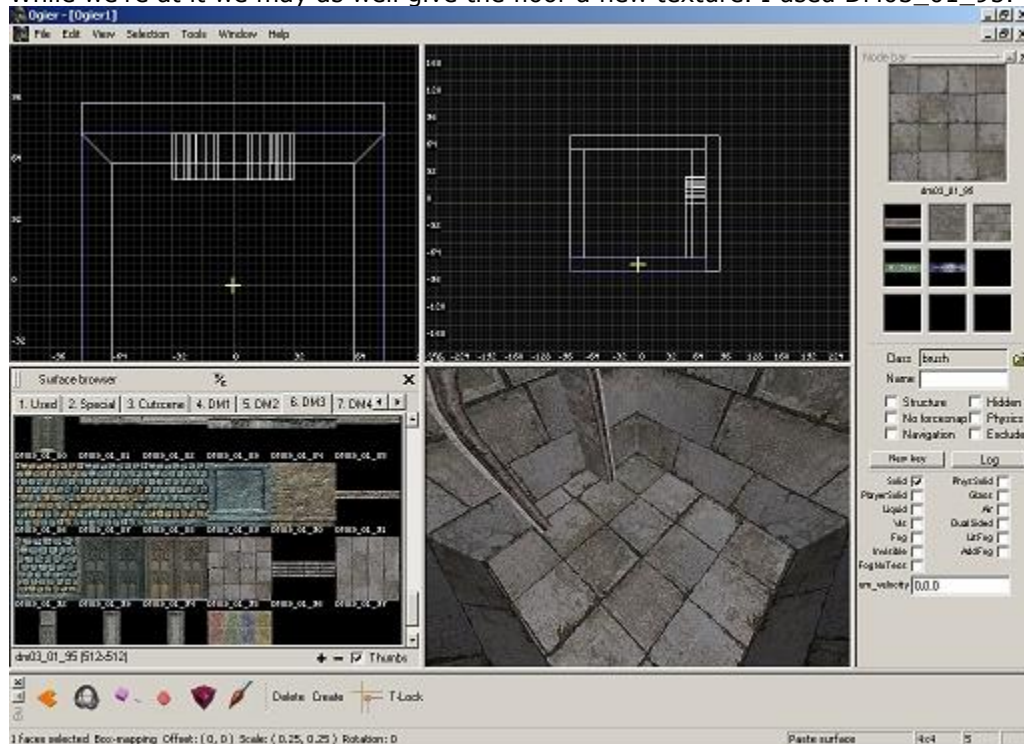
try it with a different method. Hold down the right mouse button and click/drag on the surface to rotate it by hand. Do this to rotate the texture 180' on the surface.



Hold the left mouse button and drag the texture so it lines up with the others. Deselect it.



While we're at it we may as well give the floor a new texture. I used DM03\_01\_95.



## ----Structure----

OK so now we have a room with a nice little alcove attached. The next thing we'll need to take



care of is Structure. I'll just paste in the section from the faq on this as it explains it well.

"This is an art form. I'll give you a quick summary, but this section would benefit from more detail by someone better than me.

Structure is what the engine uses to find out what polygons to exclude when rendering the world. Basically, if the polygon is behind a Structure block it won't be rendered.

The thing is that a Structure brush is extremely taxing for the cpu so unless it covers a lot of polygons it's cheaper to just render the polygons.

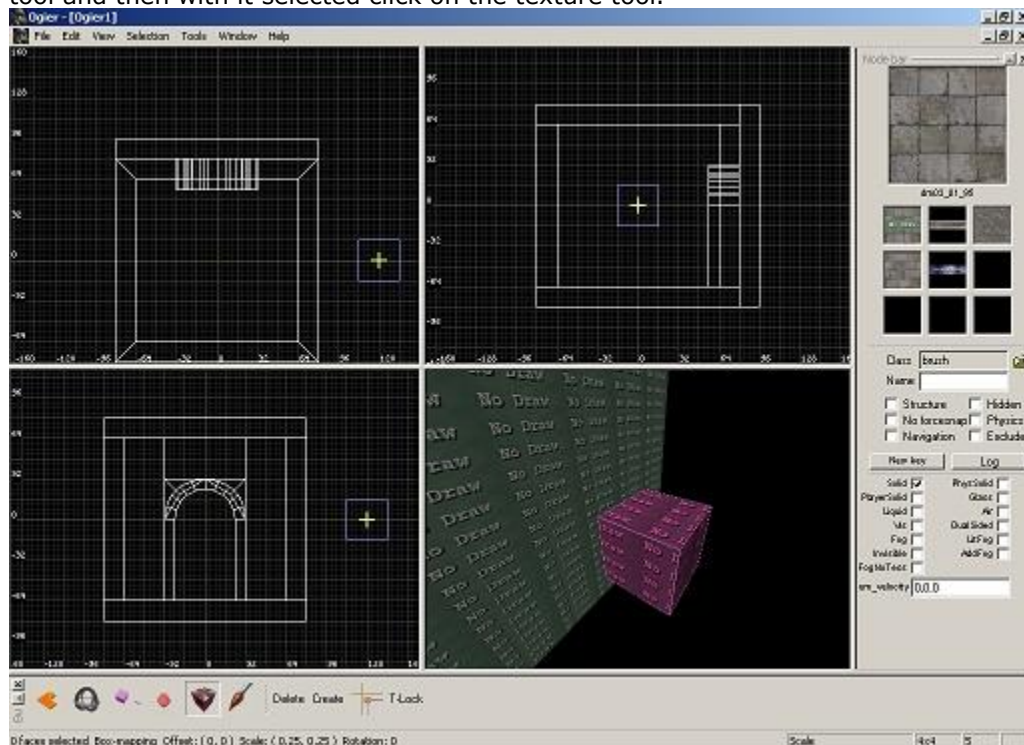
STRUCTURE SHOULD BE BUILT AS PRIMITIVELY AS POSSIBLE WITH AS FEW BRUSHES AS POSSIBLE. Basically, if you build a room you should encase it in a box of structure that has as small holes as possible for the doors.

Never make structure at an angle (unless you really really know what you're doing). Always let the brushes follow the grid lines.

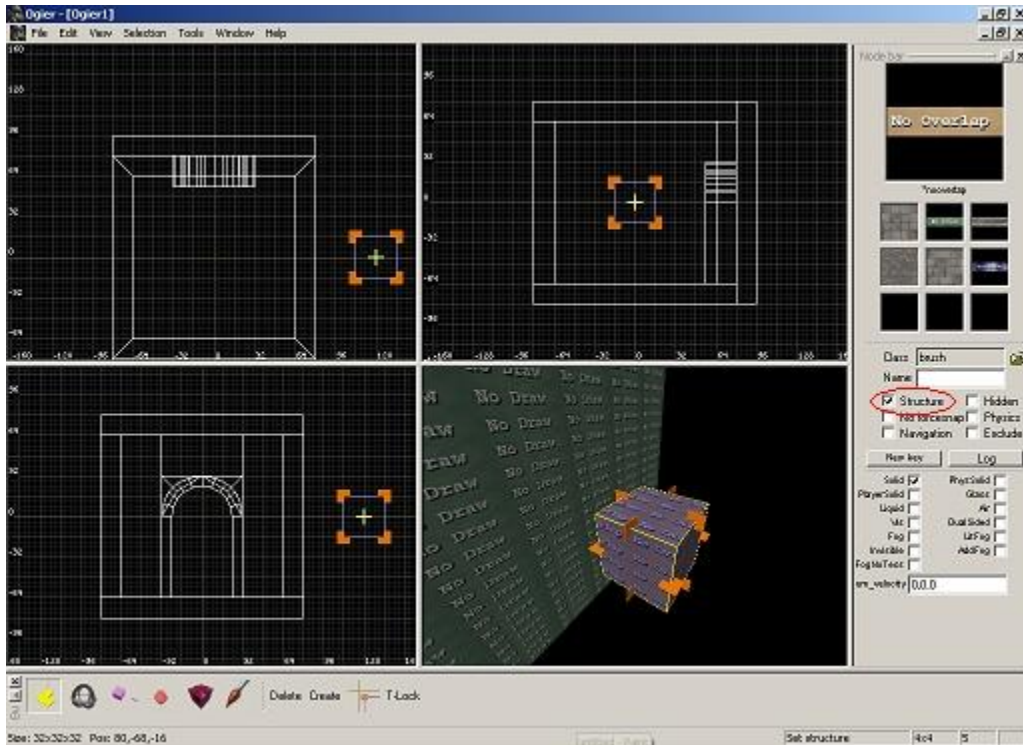
A Structure brush should always have the "NoOverlap" texture on all faces. This will make it invisible in the engine as long as it is inside of a regular visible brush (the face of the structure block can be in the exact same space as a regular visible face if you wish)."

Now as this is only a test map we'll just add a structure bush outside the level so it will compile. (If one of these isn't in the level it will not compile or run.)

Create a new box and place it outside the room we've created. Adjust it's size with the resize tool and then with it selected click on the texture tool.



Hit "X" to select of all the faces of this brush at once, then open the texture browser with "~" and click on the "Special" tab. Click \*Nooverlap and hit "K". Now switch back to the resize tool and on the right side of the screen in the Node bar, check off the box that says "Structure". Press Ctrl-R to show only structure brushes to make sure everything is correct, then hit Ctrl-R again to go show all.

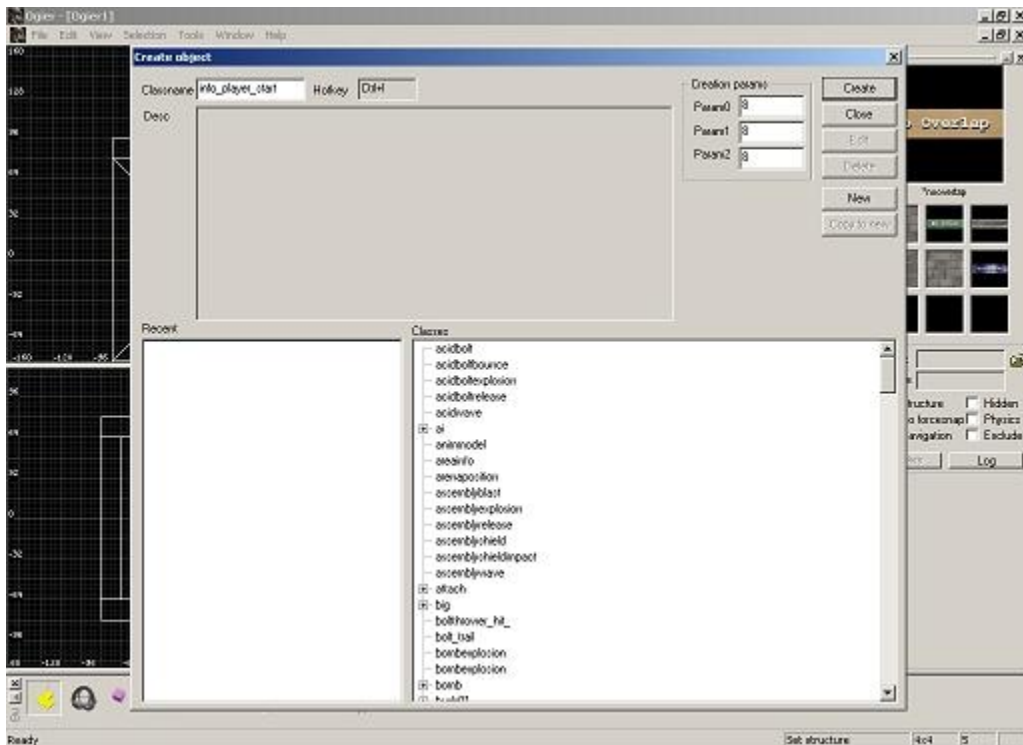


Deselect the brush.

## -----Objects and Lighting-----

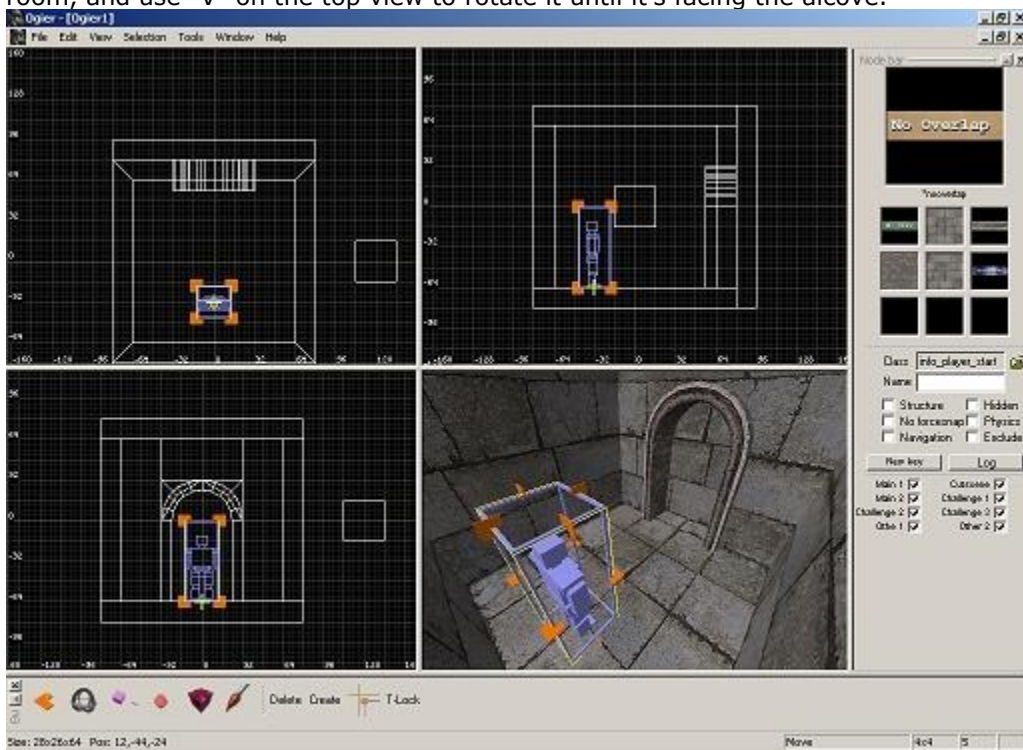
All of our geometry is now complete, lets add some lighting and objects.  
Go "Edit", "Create Object"





Scroll down through the list and expand the group "info". Click on player\_start then click "Create" in the top right corner. (Alternatively, you can just double click on player\_start.)

A player start will now be inserted into your map. Move it into position on the floor of your room, and use "V" on the top view to rotate it until it's facing the alcove.

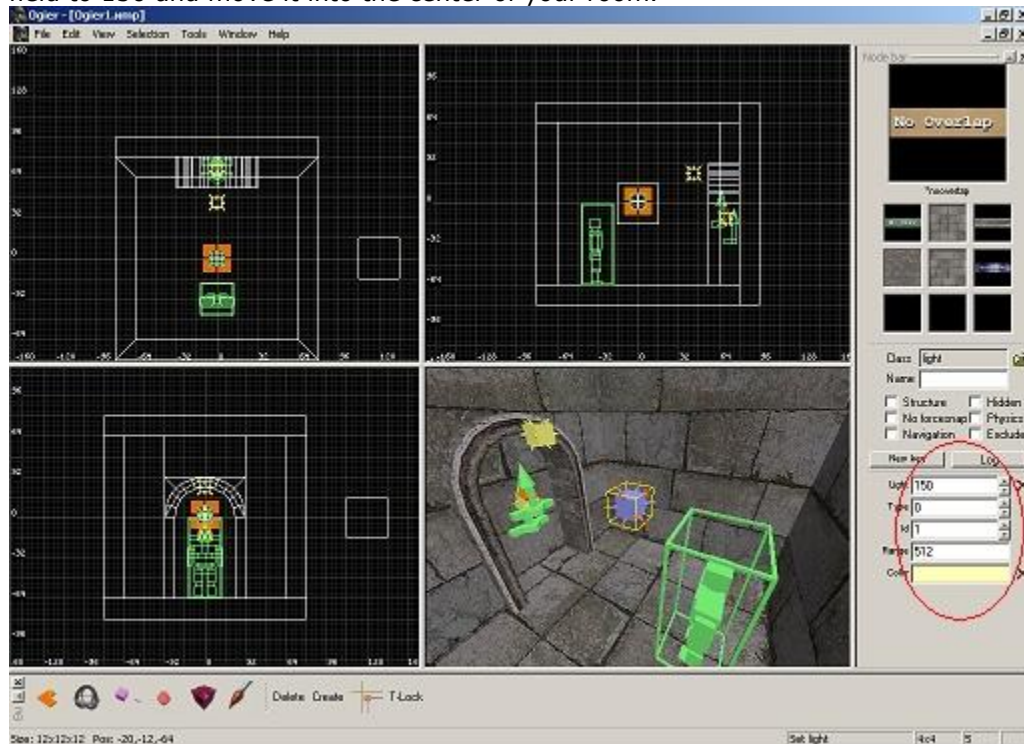


Go back into create object and this time expand "light" and select "light\_metaltorch" then





field to 150 and move it into the center of your room.



(\*Note\* You can preview the lighting by going into "View", "Preview Lighting". I suggest saving before this though as with version 1.02 of Ogier it has been giving me troubles.)

We may as well increase the grid size by one again now. Do this by holding Ctrl and hitting "Q". You can always change this as you please.

## ----Saving----

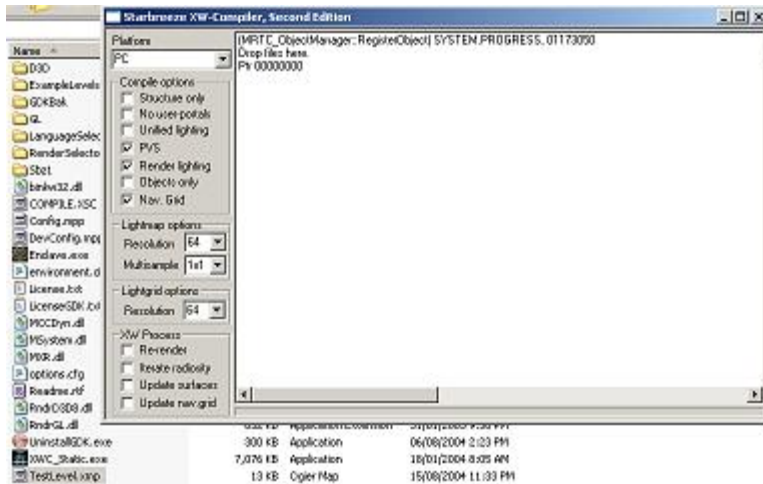
That's it! We can now compile and see how it plays. "File", "Save" the map as TestLevel.xmp (For the sake of this tutorial, I saved it in the Enclave folder.)



Close Ogier.

## ----Compiling the Map and Playing----

Open up "XWC\_Static.exe" (Installed with the GDK) and then DRAG the TestLevel.xmp file from the windows explorer window into the area containing the text in the XW-Compiler window.



The map will now compile, and place TestLevel.XW (the compiled level) into ...Enclave\Sbz1\Worlds.

Ok, it's time to check this sucker out, pop open enclave and when you reach the main screen hit the "~" key to bring down the console. Type "Map TestLevel" (Without the quotes) and then hit enter.

*(Use ctrl-alt-~ if you're building a map for Chronicles of Riddick)*





After it loads, hit the "~" key again to remove the console and then use the arrow keys and enter to select a class and begin.  
*(Use ctrl-alt- "~" if you're building a map for Chronicles of Riddick)*



If your esc key miraculously has stopped working at this point, bring the console back down and type Quit. 😊

Well that's all! You now have a small test level running in Enclave. Hopefully this tutorial was understandable enough to help you, and good luck with future mapping projects!

-Written by Shawn "AmranX Davison & Edited by Curtis "Methril" Hielema

## Ogier Tutorial: Ledges and Paths (Chronicles of Riddick)

Written by Daniel "dnadns" Schieber

So what I wanted to do were basically some ledges since Riddick refused to start climbing some boxes I made and I had to start playing around with the editor.

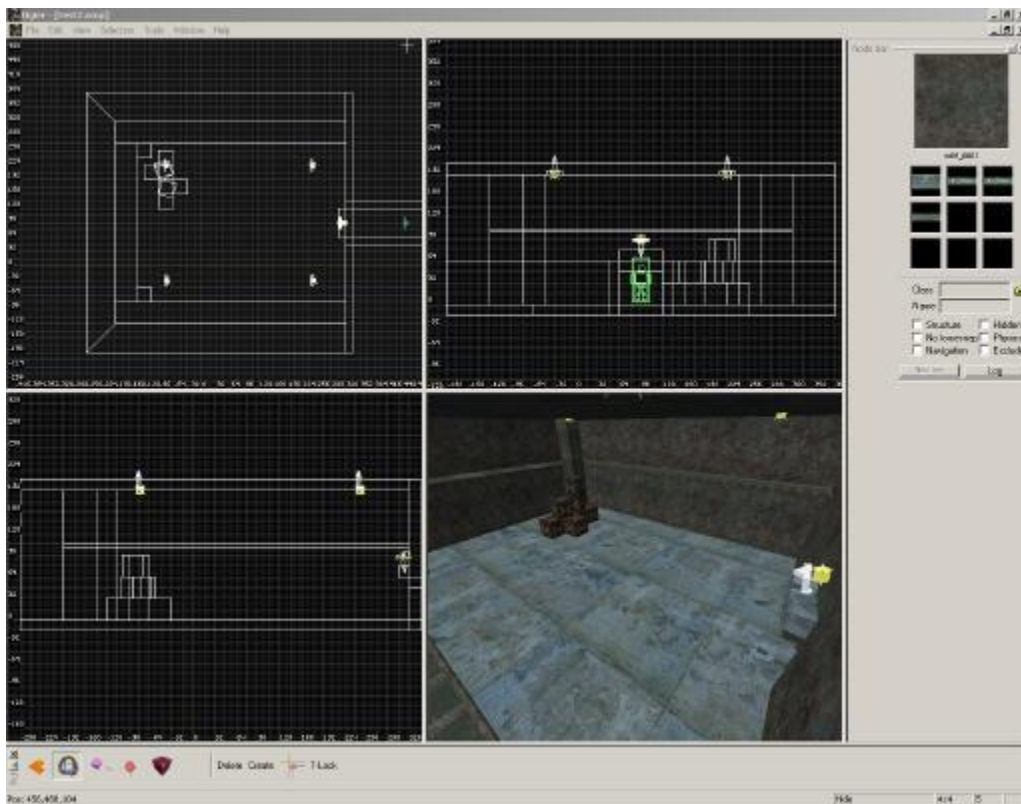
If it happens that you are as new to mapping as me, I am providing this little guide, so you don't have to start searching like I did.

By the way, since I am from Germany and English is not my native language you might stumble across some errors....uhm...strange accent (hehe) which I want to apologize for.

So well, let's get it on...

### Ledges

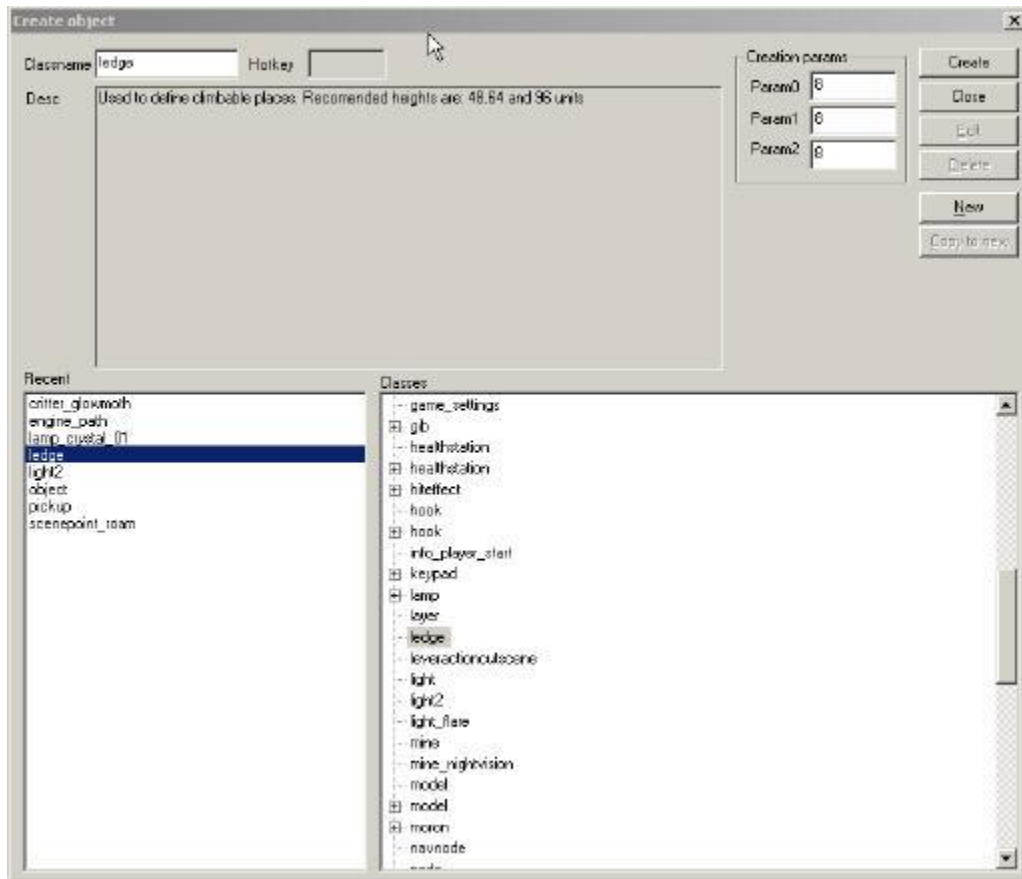
I guess everyone is quite familiar with the creation of brushes by now and managed to build a warehouse-like scene as I did here:



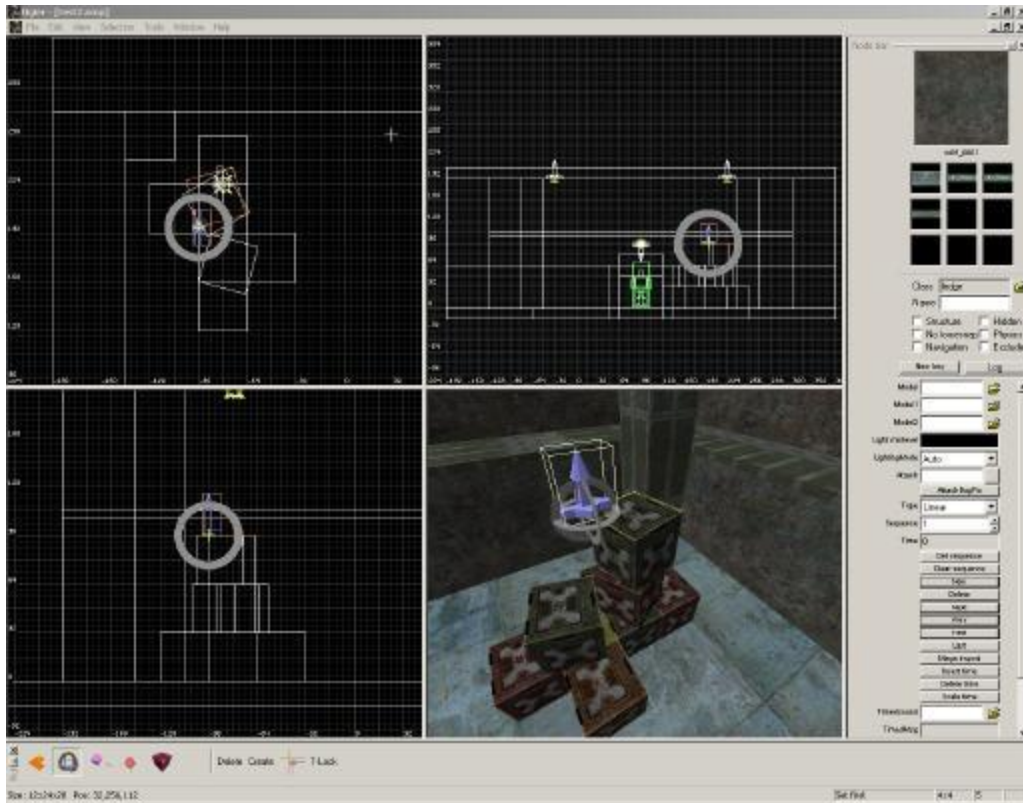
As the topmost box in the corner is too high to jump up, Riddick shall be able to climb it. What we need is a "ledge" object, so hit <Insert> on your keyboard and choose "ledge"



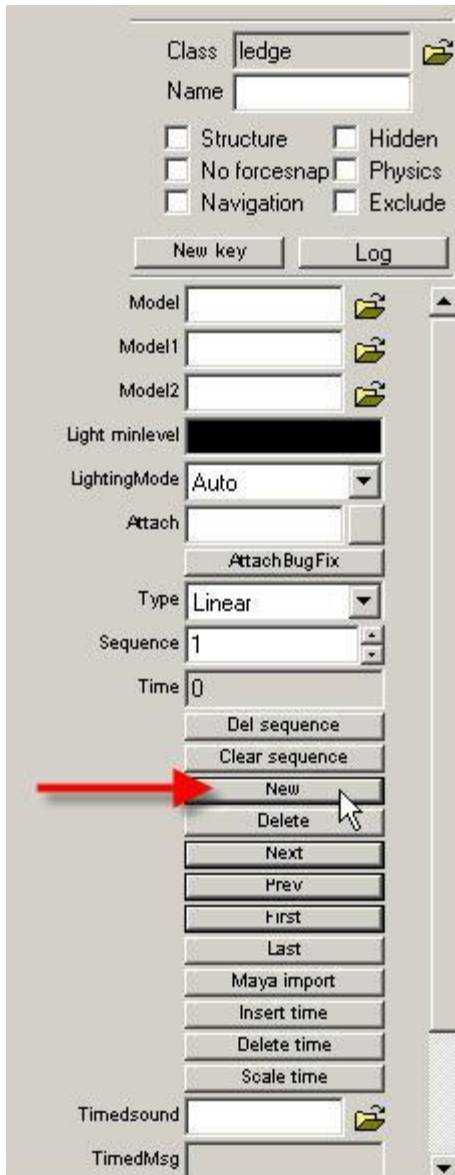
from the list...



Place the freshly created object right on top of one of the box's corners where you want to make climbing possible.



What we will need now is the actual ledge which is defined by a "path" of our object. Creating one is fairly easy as soon as you know where to click 😊  
 Make sure the "ledge"-object is selected and have a look at the Node bar to your left. To draw a "path" you have to set waypoints where the ledge will be.  
 The first waypoint is the position of our object right now.  
 In order to create another waypoint you click on the "new"-button within the Node bar.



This will pop up a small dialog where you are able to define a time index.  
 For our ledge this is pretty uninteresting right now, but we will come back to this later, while having a look on patrol paths for some guards.  
 So hit <Enter> or click on the "OK" button to create your waypoint.

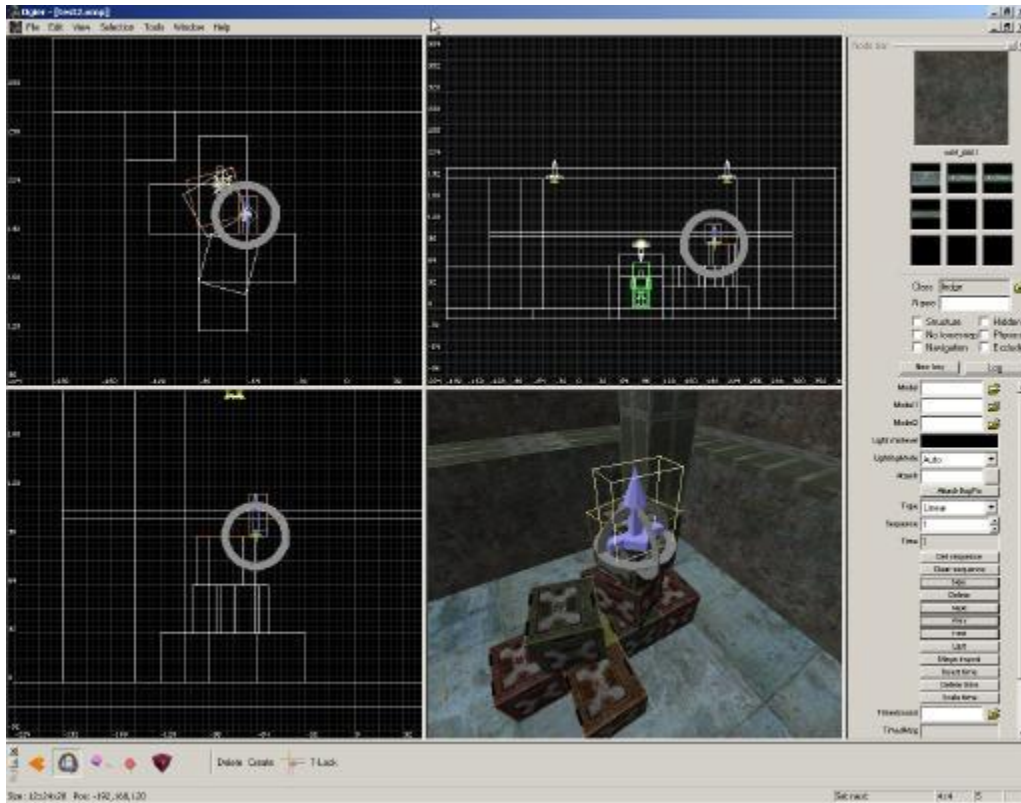


You can see which waypoint is currently selected by looking at the "Time" field in the Node bar.  
 To make things easier, the new waypoint gets selected automatically.

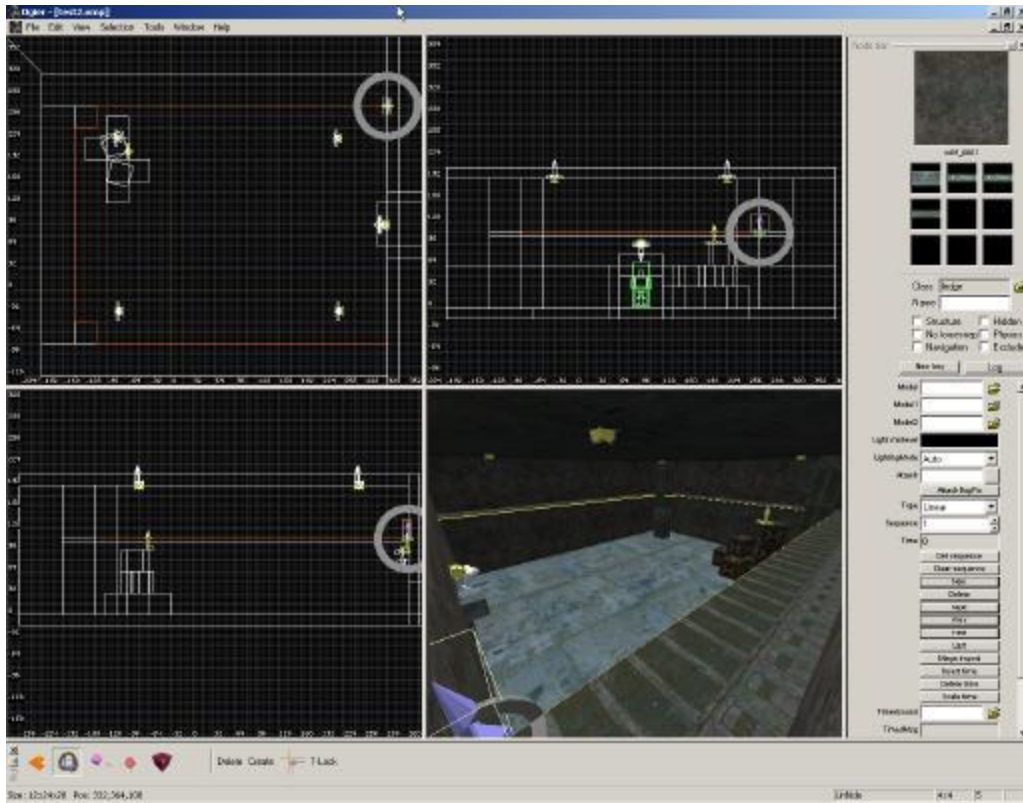
If you want to adjust things or select a specific waypoint just use the "Next", "Prev", "First", "Last" buttons in the Node bar.  
Deleting a waypoint is simply done by hitting "Delete" there.

So move the "ledge" object to the next corner of the box and see how a slight yellow/orange path is drawn between both positions.

To make all sides ready for climbing, just repeat above steps by inserting new waypoints and adjusting the position of the "ledge" object until you have drawn a complete square on top of the box.



The walkway on the next image is done in the same manner using a new "ledge" object.



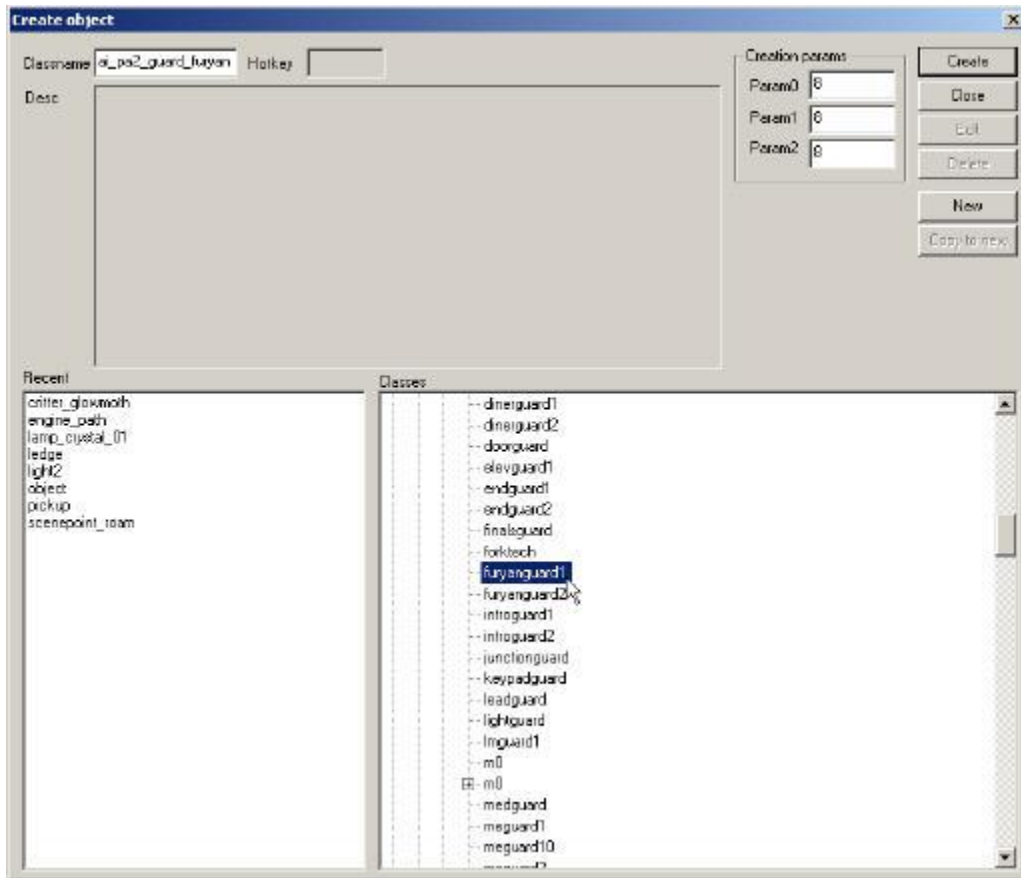
## Patrol Path

Just climbing around is way to easy, so we want to add two guards, form a team and let one of them patrol the upper walkway.

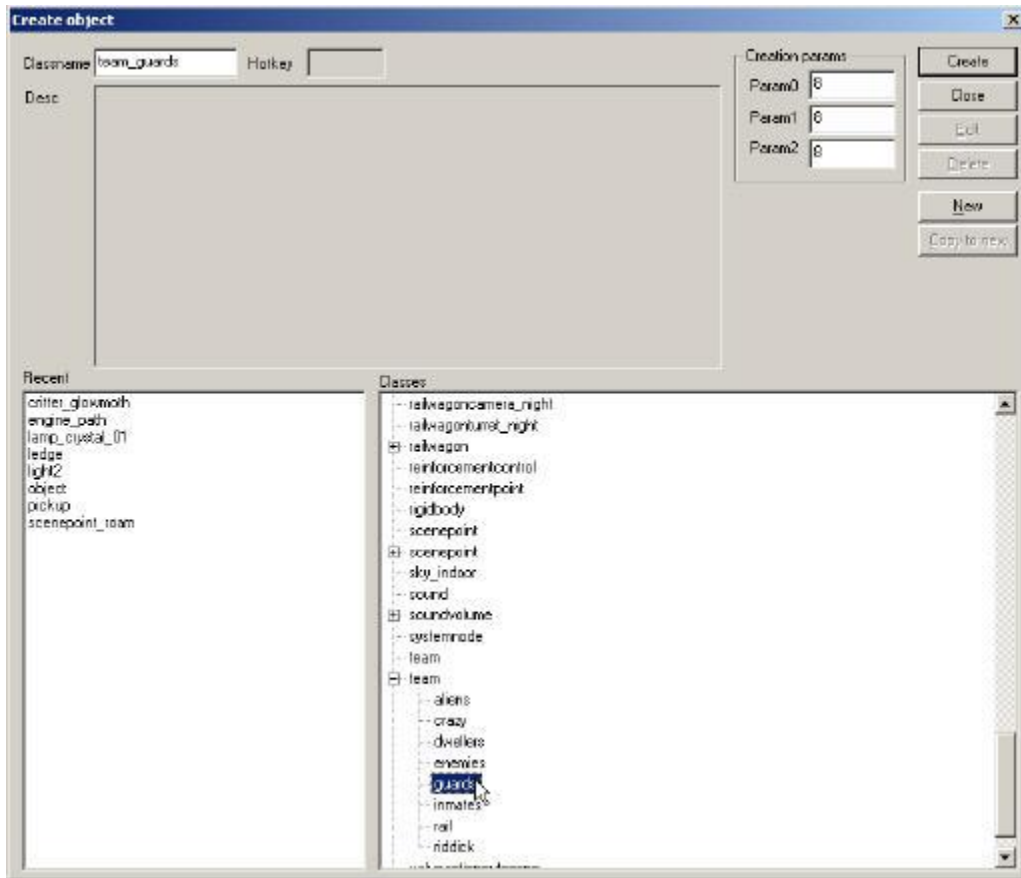
For adding the guards proceed as with any object.

Hit <Insert> on your keyboard and select any guard you desire.

(I chose ai->pa2->guard->furyanguard)



Also we want those two guys to act as a team, so we add another object to the room called a "team" marker.

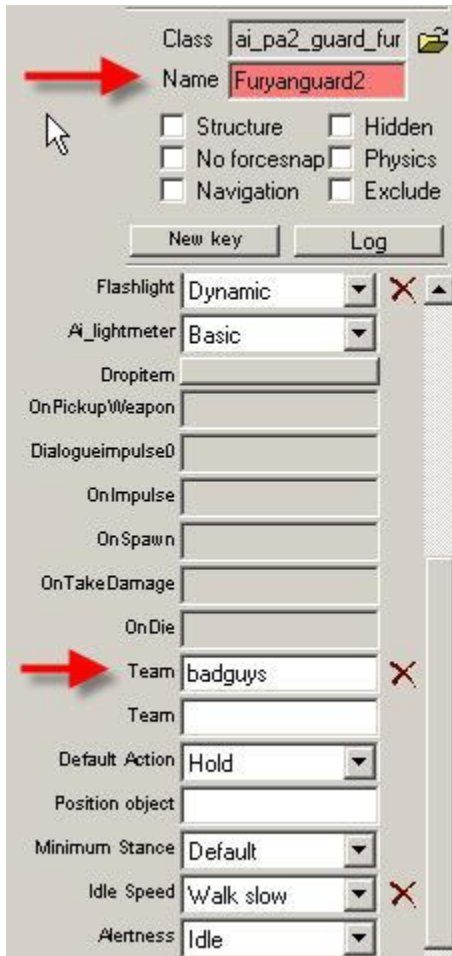


Let's give our team a name and tell it to communicate via voice.  
You can do so by selecting the "team" marker and editing the indicated fields in the Node bar.



To add both guards to the team just select one of them and enter a name as well as our team name (badguys in my example) in the Node bar,

after that proceed with the other guard in the same way using the same team name, but a different name for the guard itself.



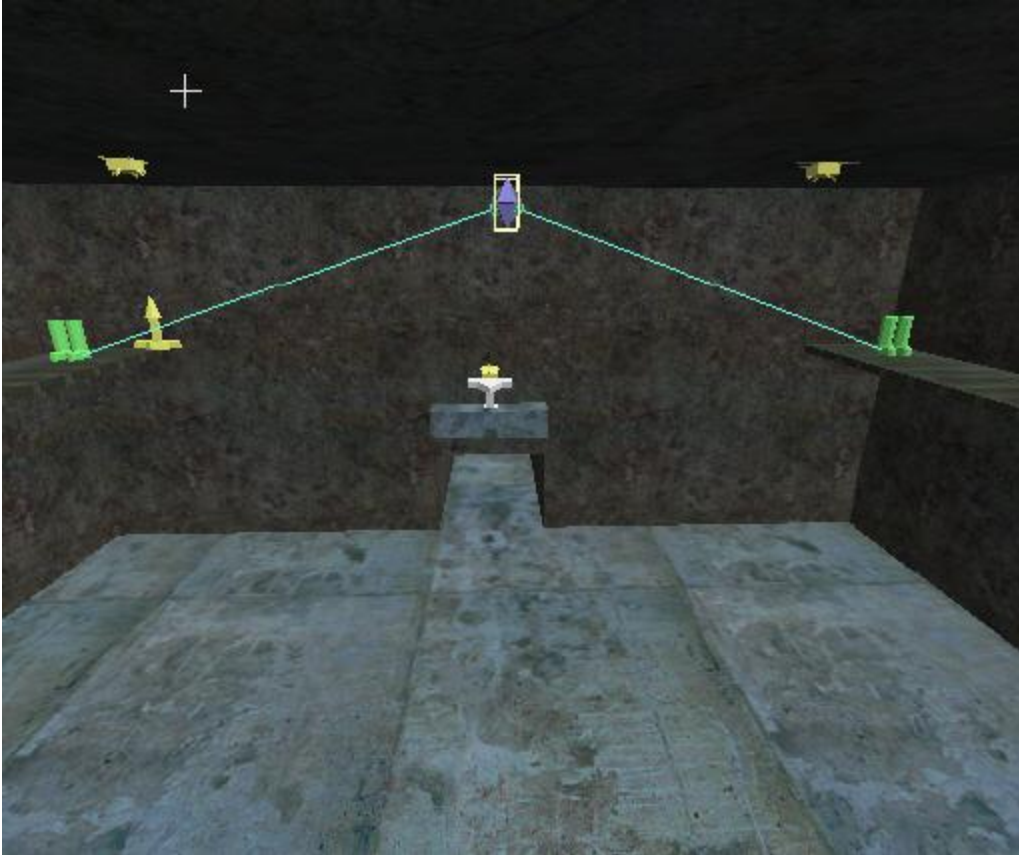
Here I placed both guards on top of the walkway.

While selecting a guard, you will see all objects that it is linked to.

In our case this is a line connecting the guard to our team marker.

The following image shows both guards in position on the walkway and the team marker being selected (which brings up both links to the guards).





Just standing around is a bit boring, so I wanted to let one of those guys walk around. This is done by adding a new object, the "engine\_path". To add it, again hit <Insert> on your keyboard and select engine->path from the list. Place the object in front of the guard you want to patrol around and give our path a name in the Node bar. (I called it path01. Yes, very creative work here)

Also you might want to change the type here from "linear" to "spline" which results in giving us a curved route (looks more natural) instead of straight lines.

To draw the path, just use the same technique as we used for the ledges. Create new waypoints with the "new" button in the Node bar and start dragging the object to build the desired route.

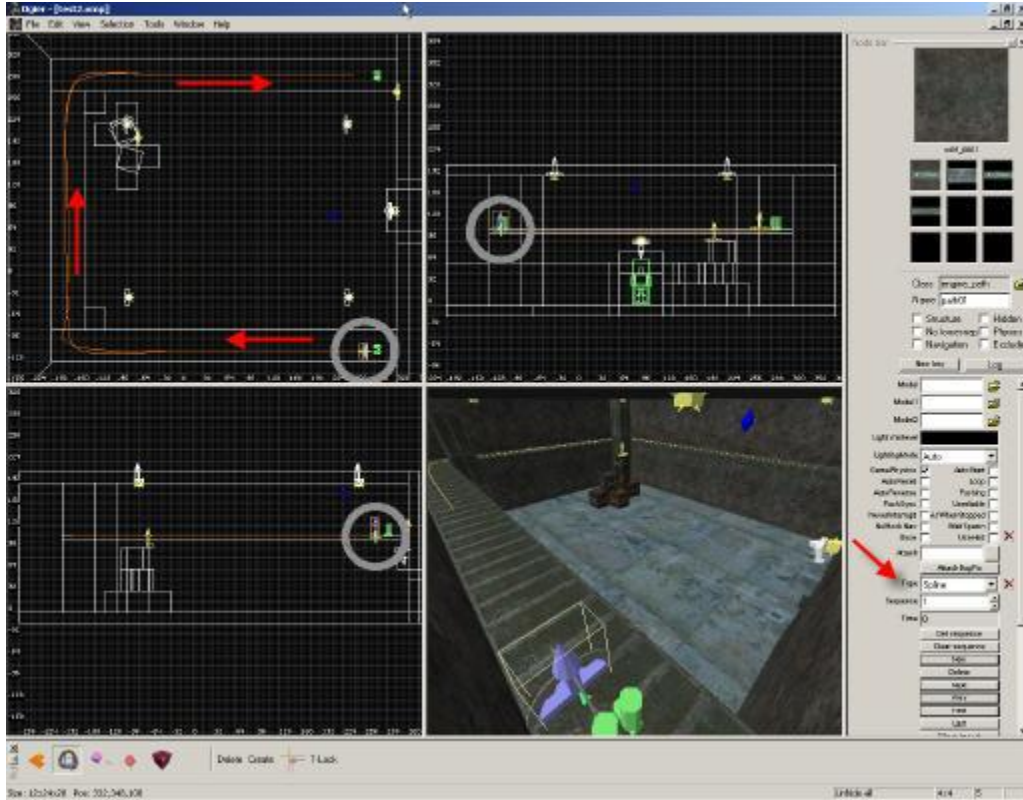
As mentioned above, the time index you are asked for at waypoint creation is of some use here.

Of course you can make the guard "hold" on its way by adding 2 waypoints to the same place.

If the first waypoint had a time index of "10.0" and the second one of "25.0" the guard will stand there for 15 seconds.

The next picture shows my patrol path from the patrolling guard to the other guard where

he will wait for 10 seconds and walk back.



The last thing we have to do here is to attach the guard to the path. Select the guard and give it a "default action" called "Patrol" in the Node bar. As for the Path param you just type the name of the path we have created. (that was the creative "path01" in this example)

Class  

Name

☐ Structure ☐ Hidden

☐ No forcesnap ☐ Physics

☐ Navigation ☐ Exclude

OnTakeDamage

OnDie

Team  

Team

Default Action  

 Path param  

Minimum Stance

Idle Speed  

Alertness

I hope it saved you some time with the ledges and paths.

Cheers,  
Dan

# **Ogier Tutorial: Building an elevator (Chronicles of Riddick)**

**Written by Daniel "dnadns" Schieber**

Hello again,

this time we are going to build a simple, but fully functional elevator from scratch.

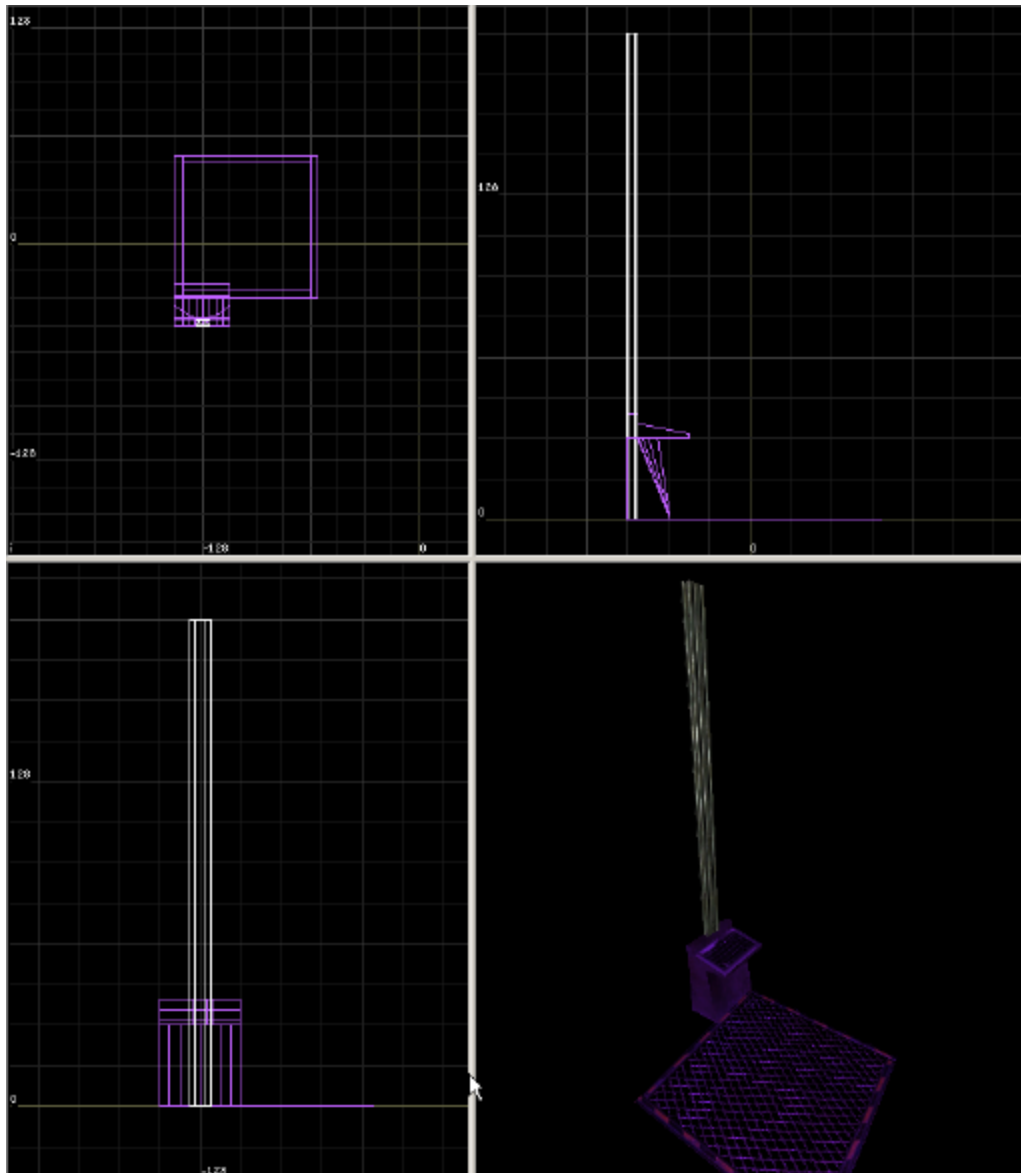
Before going on, please make sure to read these tutorials so you know the essentials needed:

[Ogier Tutorial: Creating Your First Map \(Enclave](#)

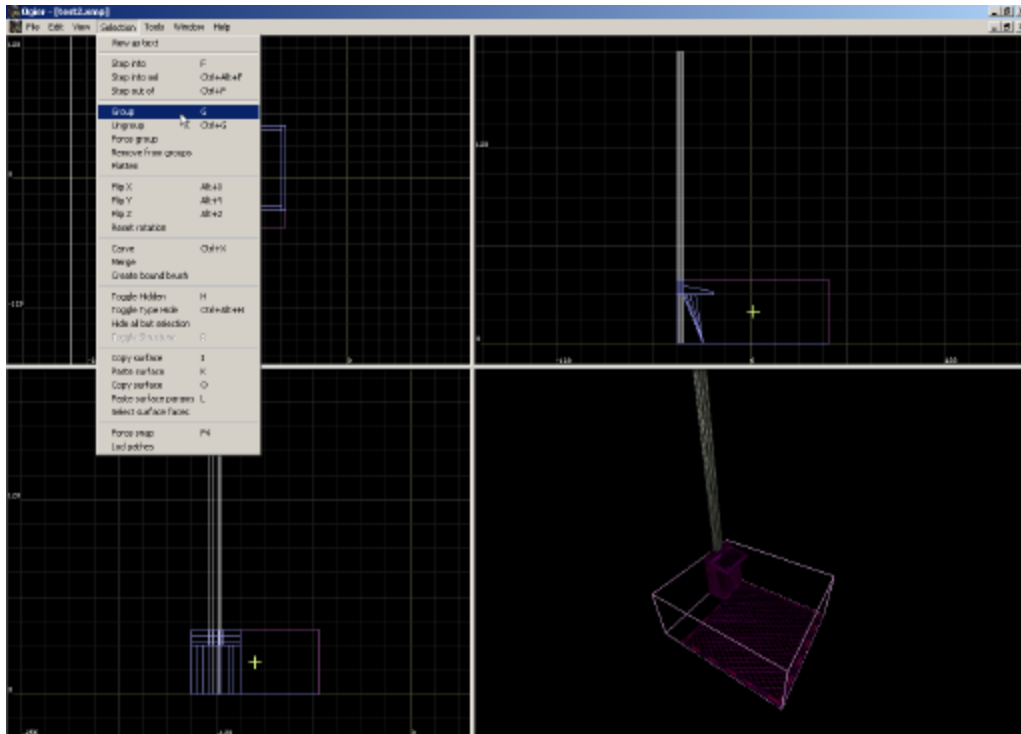
[Ogier Tutorial: Ledges and Paths \(Chronicles of Riddick\)](#)

## **Building the Material**

First of all we place some brushes and texture here to something that looks like an elevator. Mine consists of a platform, a control pad and a railing that it will travel along.



When everything looks like you want it to be, select all the moving parts (in my case, the platform and the control pad) and group them. This can be done by hitting "G" on your keyboard or by clicking on "Selection->Group".

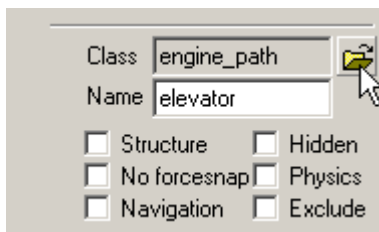


## Scripting the Elevator

The group will be part of the Class "Node" in the Node bar.

To make it movable at all, we have to change this to "engine\_path".

Click on the "browse" button to bring up the class selection dialog and select "engine\_path".



As you can see now, there are lots more options to change in the Node bar. Be sure to select the following flags there:

GamePhysics (This makes it able to travel in the world at all)

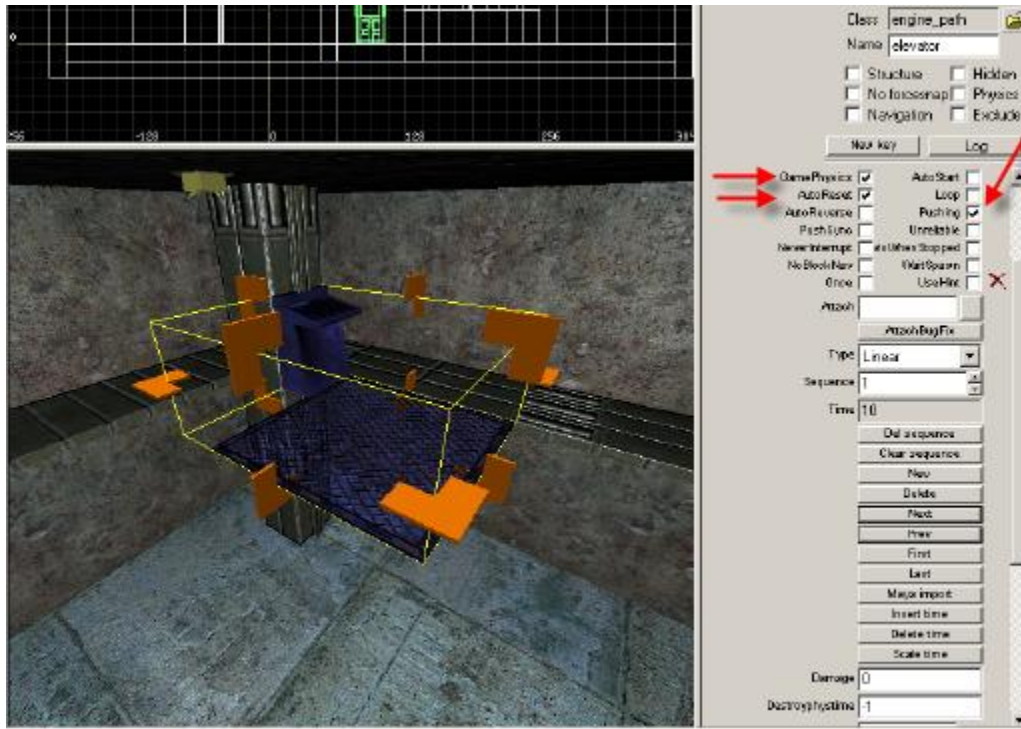
AutoReset (We want our elevator to be used as often as the player wants to later, so it has to reset it's position to Start after finishing the path)

Pushing (With this flag the path is allowed to move things around. That means, a player standing on the elevator)

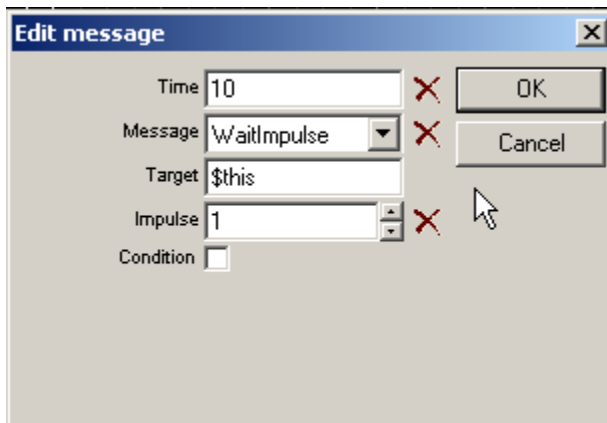
Also name the path to "elevator" here.

The next step should be familiar from building paths in general.

Start by clicking on "new" in the Node bar to create a waypoint.  
I gave mine a "time index" of 10, and moved the elevator to the position where it shall take the player.



We want the elevator to stop there.  
To achieve this, click on the empty "TimedMsg" field in the Node bar and make it a "WaitImpulse 1".



So the elevator will pause it's path there and wait until it receives the Impulse 1.

After that, add another waypoint to the path, giving it a "time index" of 20 and move the elevator back to its start position.



What we have now is an elevator that waits for a signal, moves to the top within 10 seconds, waits there again for a signal and moves back to the ground where it will reset it's position to start.

## Update

### Hair wrote:

You can also create multiple movements for the engine path by adding an other sequence.

To activate the other sequences that an engine path has you just send the sequence number as a impulse.

So you could use impulse1/sequence1 to send a enginepath up and impulse2/sequence2 to send it down and so on.

To send a signal to the elevator we need some sort of button, so let's add a new object the "actioncutscene".

Since we built a control pad, we move the object there and name it to "elevator\_btn".

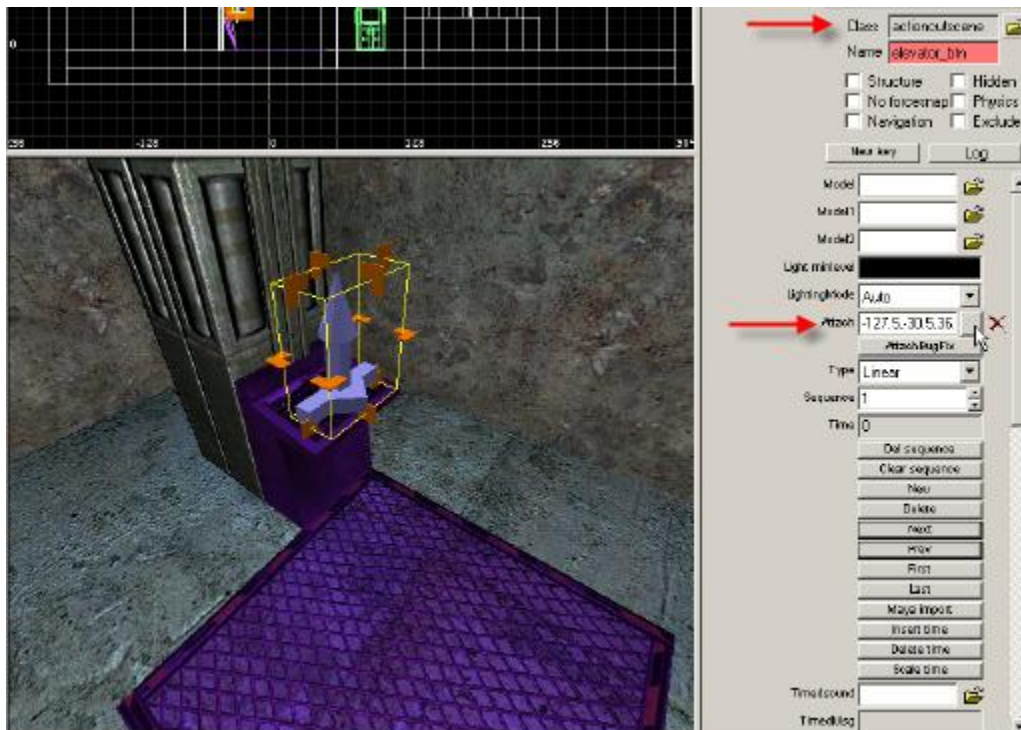
Now for an important part:

We want the button to move up with the elevator, so we have to attach it to the path.

Click on the blank button right next to the "attach" field.

This fill insert the current coordinates of the object there, looking like this: -127.5,-30.5,36.5

It still needs to know which path to attach to, so enter a comma after the coordinates and enter the name of the path ("elevator" in our case): -127.5,-30.5,36.5,elevator



So let's make this acs a real button.



Set the flag "UseThirdPerson", so Riddick will be shown how he pushes the button from an external view.

This isn't necessary at all, but looks much better 😊

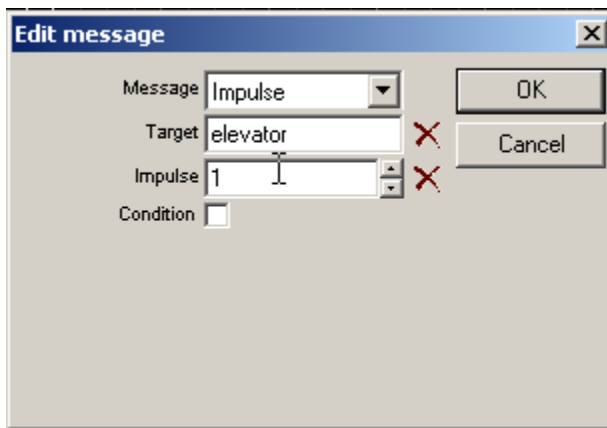
Then click on the field "SuccessType" and change it to "USEBUTTON", so the game how the player will interact.

Also set the "Retrycountdown" to something practical so the player cannot use the button a hundred times in 2 seconds.

Remember the signal for the elevator? Right, we are doing this now.

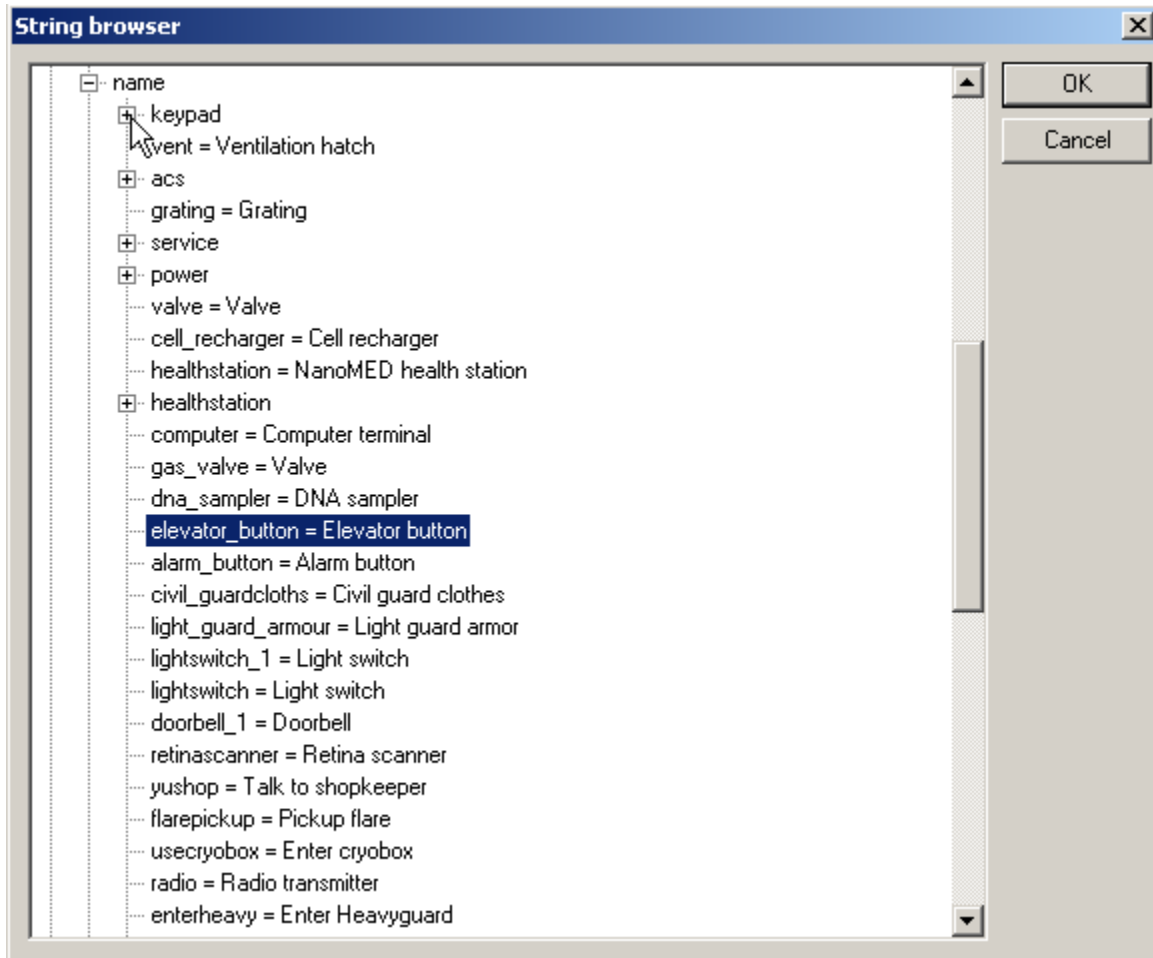
Click on the empty field "TriggerSuccMiddle" and make it an "Impulse 1" send to the target "elevator".

You could also use "TriggerSuccess" for that, but I want to use that for something else later in this howto.




Yet the player will not see a message in-game that there is a button.

So click on the browse symbol right next to the "Focusname" field and select the string for "elevator\_button". (This will translate automatically, depending on your localization)




So that's it!

Class  

Name

☐ Structure
 ☐ Hidden  
☐ No forcesnap
 ☐ Physics  
☐ Navigation
 ☐ Exclude

☐ Rotate
 ☐ Fall  
☐ Waitspawn
 ☐ CrouchAtEnd  
☐ UseThirdperson
 ☒ Depends On Item
 ☐  
 Success Set Start Position
 ☐ s Set End Position
 ☐  
 Fail Set Start Position
 ☐ ail Set End Position
 ☐  
 Locked
 ☐ sable On Success
 ☒

Depend Item   
 DependMsg   
 Startposition   
 Endposition   
 SuccessType  ☒  
 FailType  ☐  
 Actioncutscenecamera   
 TriggerSuccess   
 TriggerSuccMiddle  ☒  
 TriggerSuccMiddle   
 TriggerSuccessEnd   
 TriggerFail   
 TriggerFailMiddle   
 TriggerFailEnd   
 Minactivationtime   
 Retrycountdown  ☒  
 Focusname   ☒

A fully functional elevator to go:



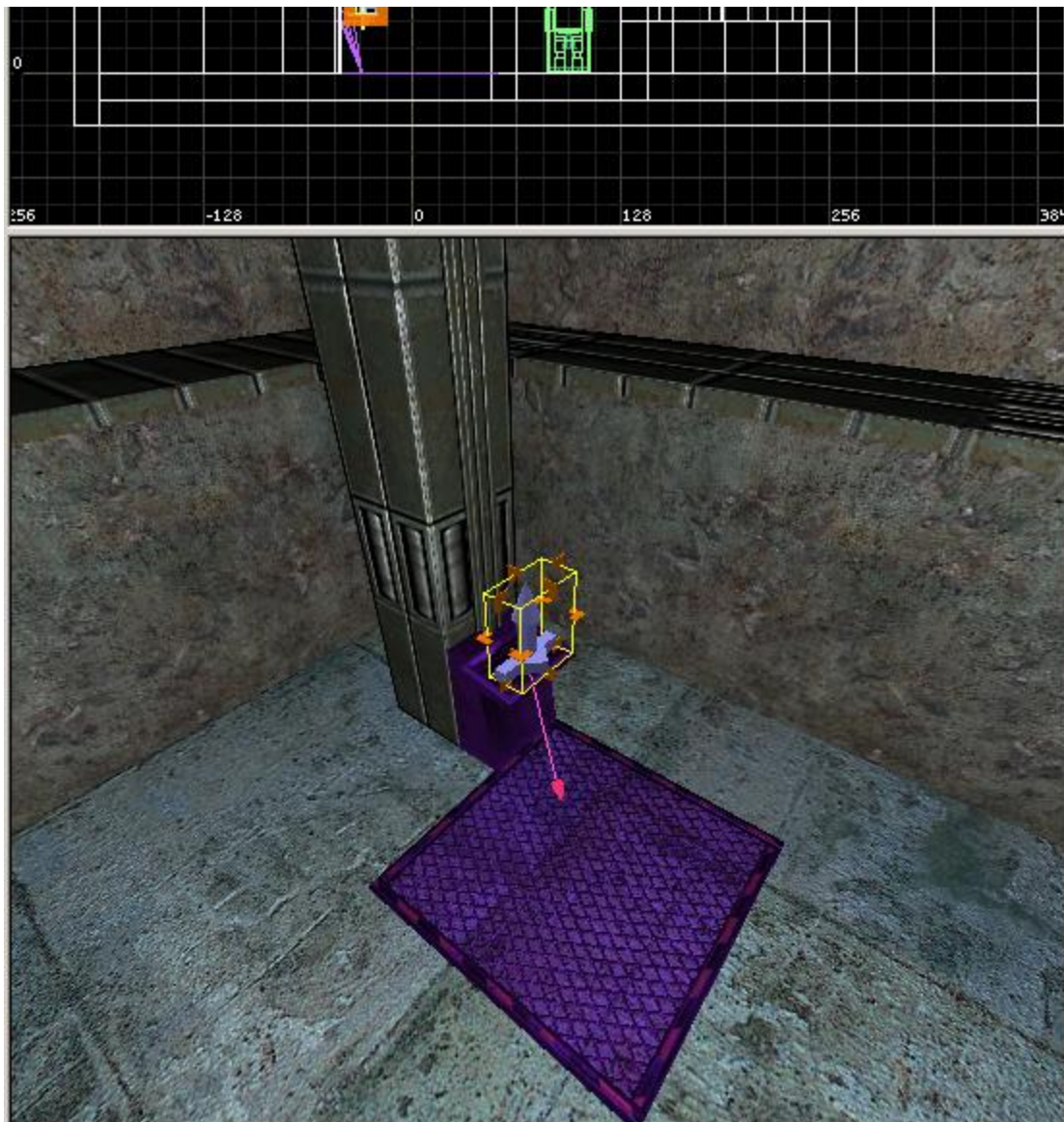
Still lacks a bit of atmosphere here, so we will add some...

### **Sounds**

Before I said the "TriggerSuccess" for the elevator button has some other use. I want to play this a sound, so click into the field to bring up the Message dialog. Select "PlaySound" here and browse for a sound that you thinks fits here:  
(Loading the soundbrowser takes a while, please be patient here)



So now our button will trigger two events when being used, tell the elevator to move and a sound to be played:

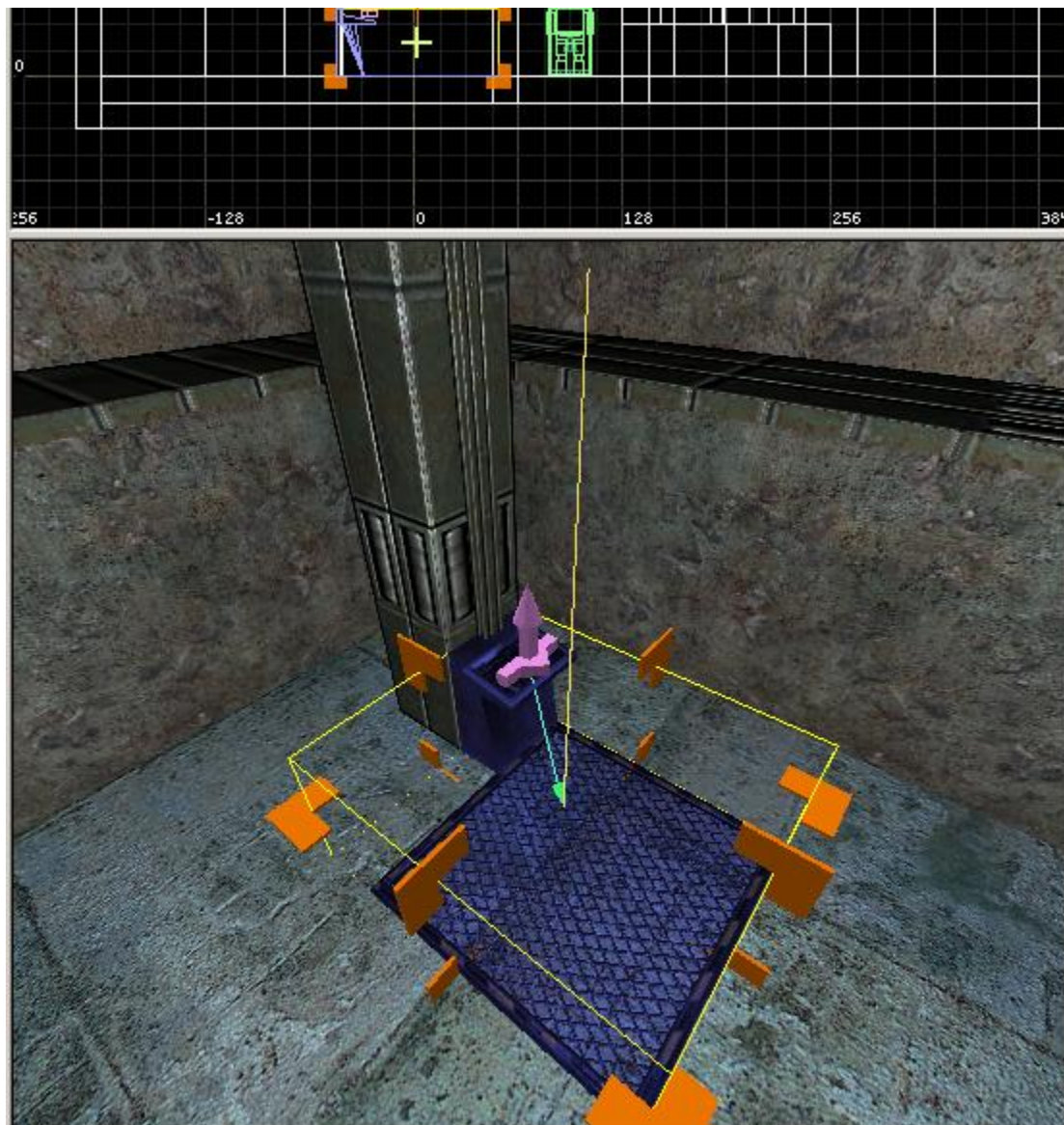


Class	
Name	
<input type="checkbox"/> S	
<input type="checkbox"/> N	
<input type="checkbox"/> N	
New	
DependItem	
DependMsg	
Startposition	
Endposition	
SuccessType	US
FailType	NO
Actionoutscecamera	
TriggerSuccess	Pla
TriggerSuccess	
TriggerSuccMiddle	Im
TriggerSuccMiddle	
TriggerSuccessEnd	
TriggerFail	
TriggerFailMiddle	
TriggerFailEnd	
Minactivationtime	0.0
Retrycountdown	7
Focusname	\$L
Focusdesc	
Focusframeoffset	
Light Name	

Proceed in the same manner with the elevator.  
 Since it is an engine\_path you can place sounds by adding "TimedMsg" to it and select "PlaySound".

I added sounds for start, travel and arrival in both directions.





## Conclusion

What we have now is a traveling elevator with a button which both make sounds when used.

So the script does this:

1. The player pushes button (ACS: elevator\_btn)
2. The elevator\_btn plays a sound as "TriggerSuccess"
3. The elevator\_btn sends "Impulse 1" to the elevator
4. The elevator starts it's "engine\_path" at "Time Index 0"
5. On "Time Index 0.1" it will play a sound (unlocking some bolts)
6. On "Time Index 9.5" it will play a sound (pressure hiss)
7. On "Time Index 9.9" it will play a sound (locking some bolts)
8. The elevator sends "WaitImpulse 1" to itself and therefor pause the path.
9. The player uses the button again.
10. The elevator continues it's path down again where it will finish and reset to start.

That's all folks 😊

Hope you enjoyed!

Dan



# Ogier Tutorial: A simple sky (Chronicles of Riddick)

Written by Daniel "dnadns" Schieber

Hi again,

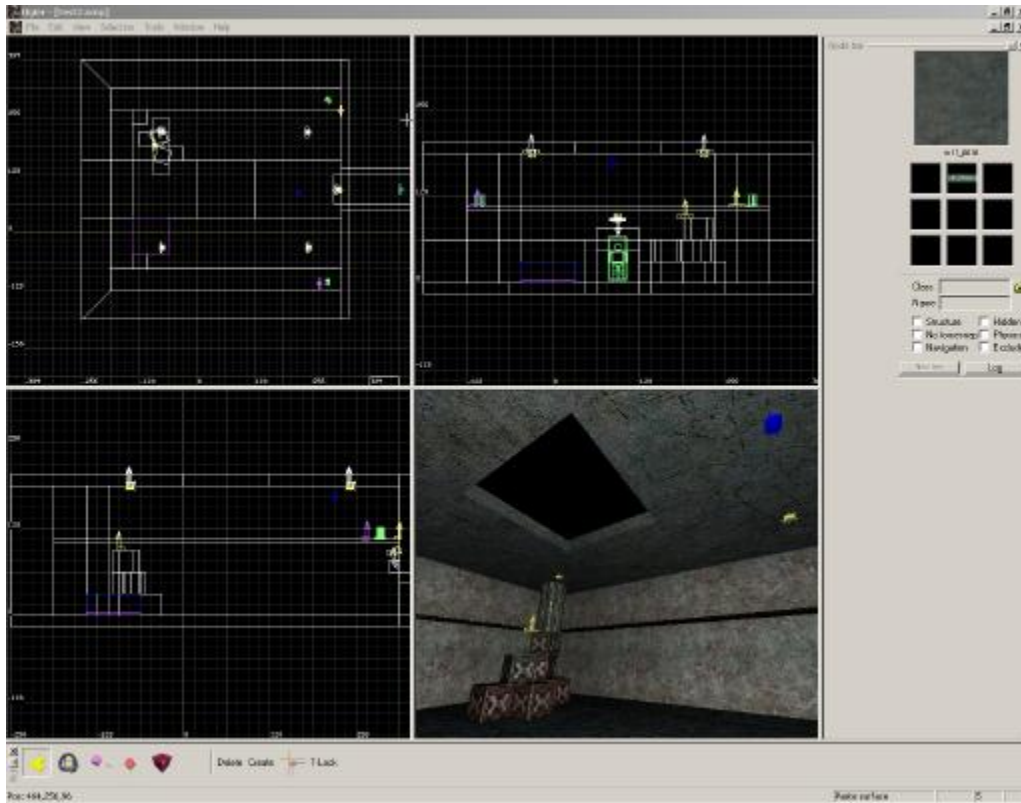
as promised I'll give a short introduction on creating a sky.  
It is really simple to do and creates much atmosphere to a scene.

Here we go:

## Skies

using the scene already introduced in the path HOWTO, we will carve a light hole into the roof of the warehouse.

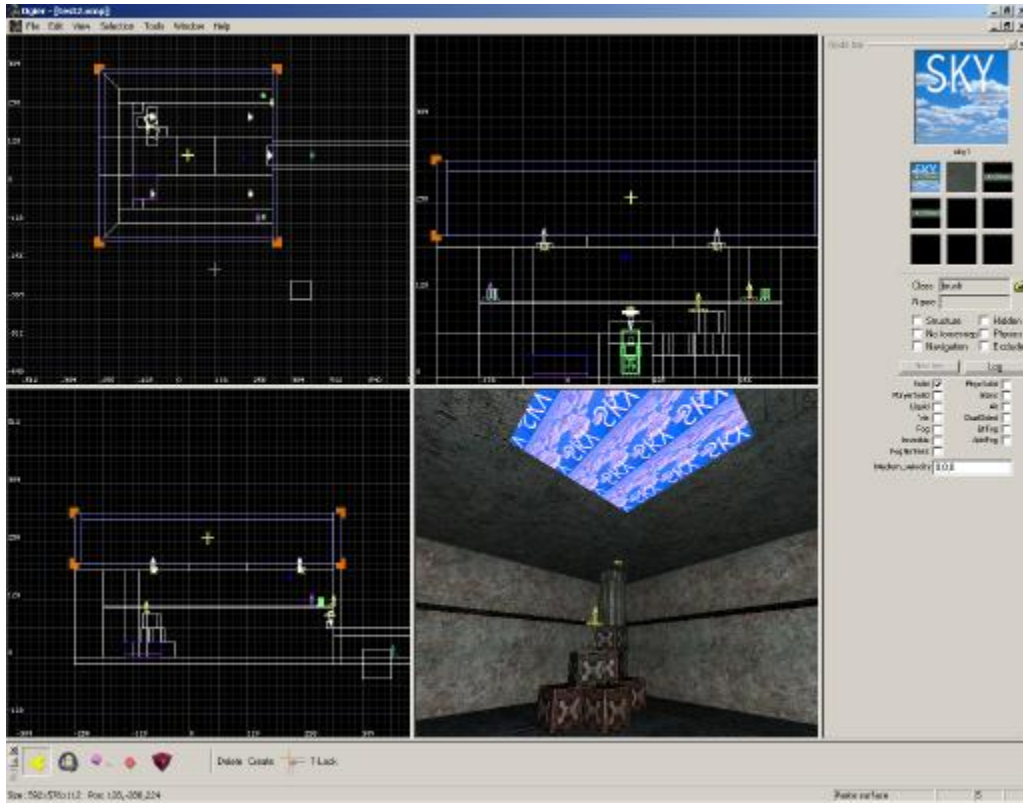
Create a brush of the desired size, place it in the roof and hit "CTRL+X" on your keyboard to carve the hole.



In the next step, we will need to create a second room above the warehouse; a second story. After doing so, select all of the meshes that are going to be our sky and assign the "sky" texture to it.

(You can find that in the "special" tab of the texture browser)

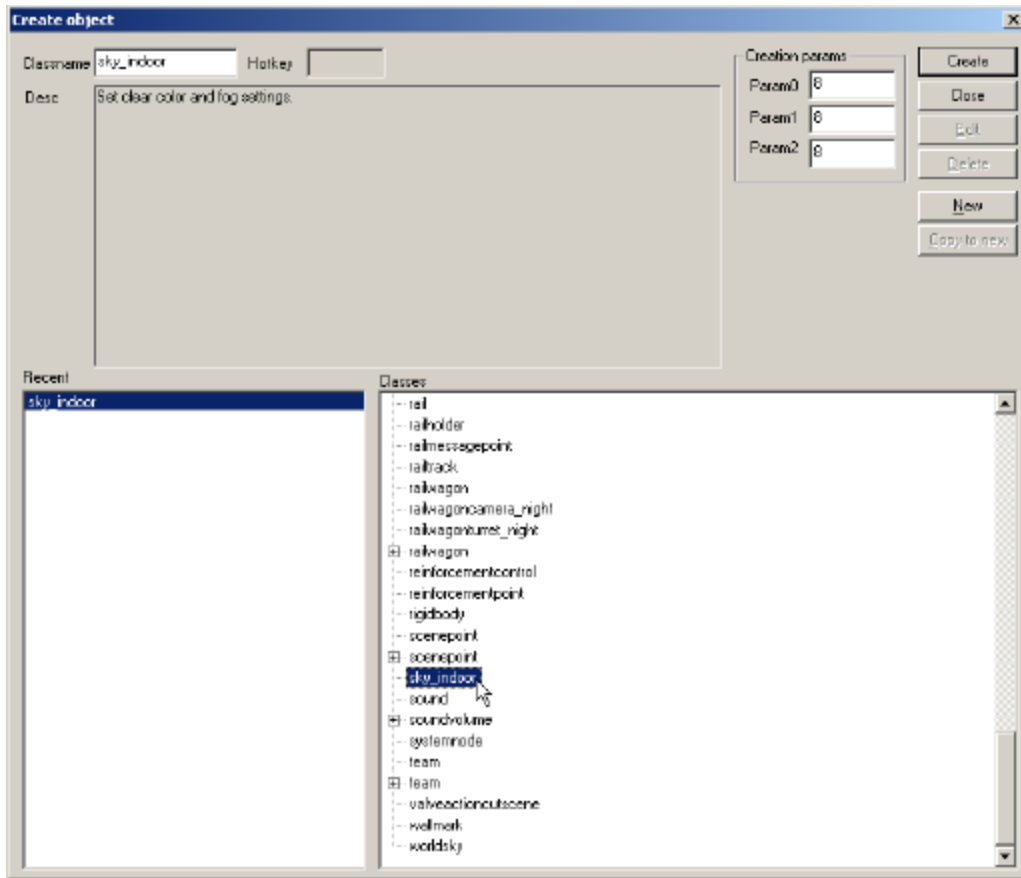
So our scene will look similar to this:



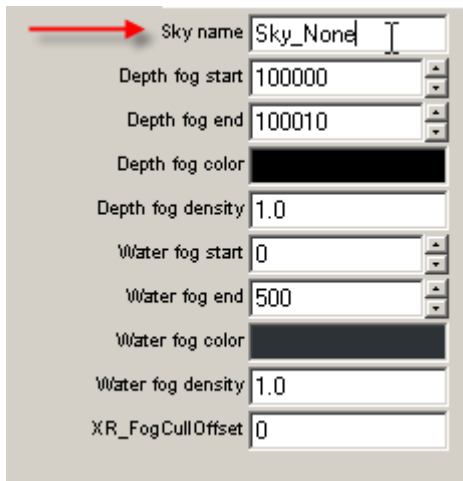
So where do we choose what sky will be shown in the game?

We need a special object to define this and hit <Insert> on the keyboard to bring up the object dialog.

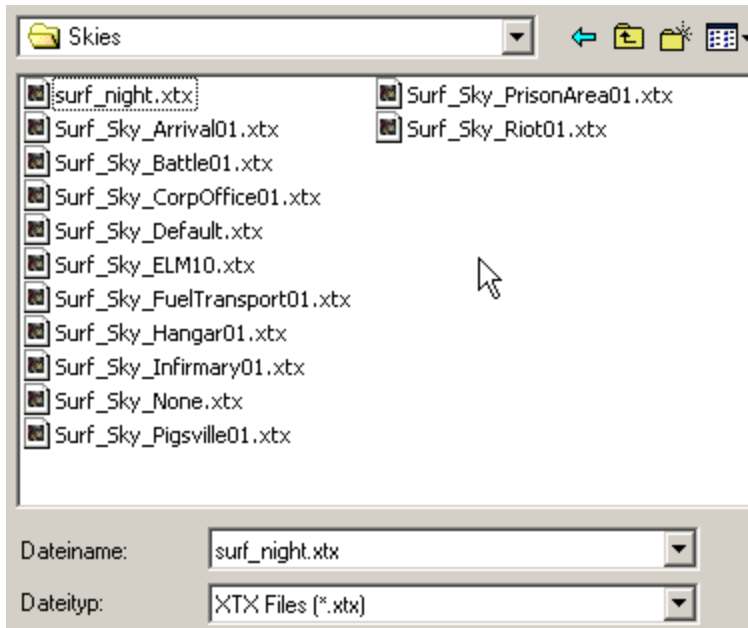
The desired object is called "sky\_indoor", so we select and add it to the scene.



Select the "sky\_indoor" object, so we can edit it's properties in the Node bar. The "Sky name" field needs a valid entry to show the sky on all brushes with the corresponding texture.



Unfortunately we won't get a dialog to choose from when clicking on the field. So a quick look into the folder "Riddick EFBB\Content\Skies" will show us the available skies.



You can also open these files and have a quick look at them in Ogier or even change additional properties.

For this example I chose surf\_night.xtx and therefore entered "night" (just leave out the surf\_ of the filename) in the "Sky name" field of the "sky\_indoor" object.

And that's it!



Still there could be some sort of window brush/texture added to make it look even better, but I will leave that to your imagination. 😊

EDIT:

In order to get a sun, just add a light2 under the sky and adjust its values to your needs. 😊

Cheers,  
Dan

# Ogier Tutorial: Multi-Part Door (Chronicles of Riddick)

Written by Daniel "dnadns" Schieber

Hi again,

this time I will show how to build a multi-part door, meaning several parts will move.

What you should know by now:

- Creation of brushes
- Engine paths (optional)
- Timed Messages (optional)

The last two parts are optional, because I will not go into full detail on those points.

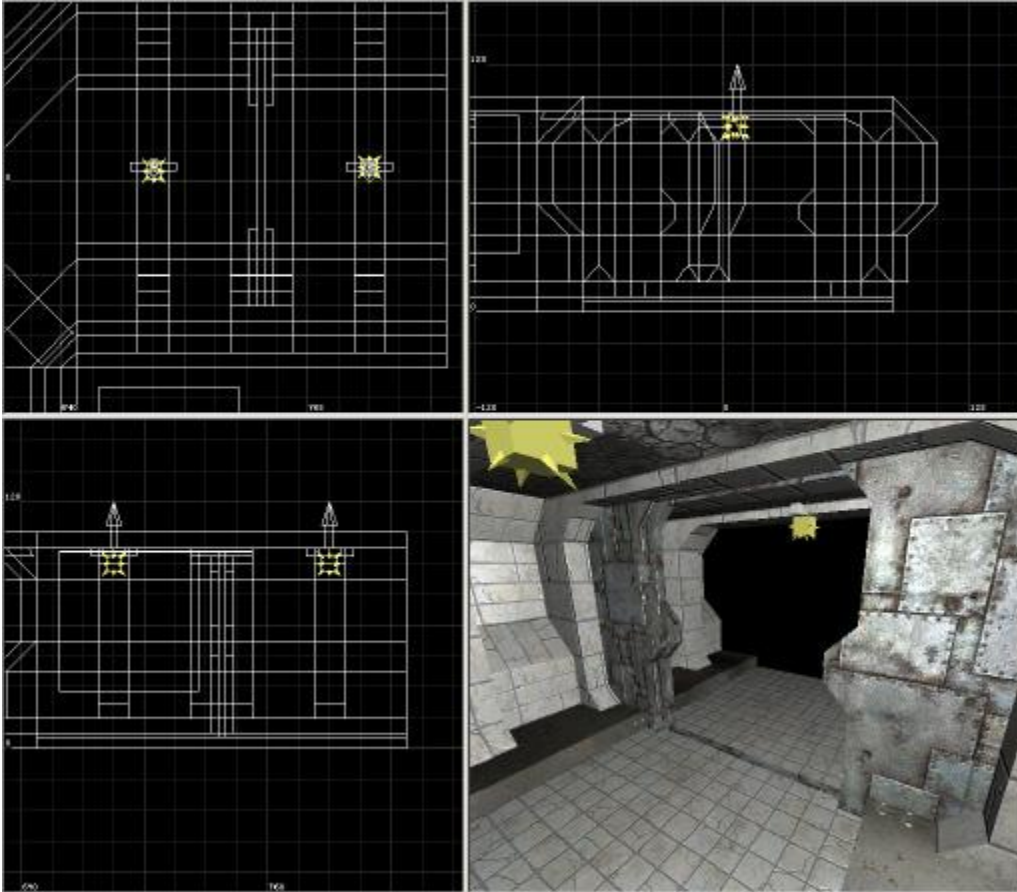
(If you want to get further information on these topics, please have a look at the Elevator and Paths Tutorials.)

- [Ogier Tutorial: Ledges and Paths \(Chronicles of Riddick\)](#)
- [Ogier Tutorial: Building an elevator \(Chronicles of Riddick\)](#)

Let's get started:

## Brushwork

What we see here is a doorway our door will be placed in:



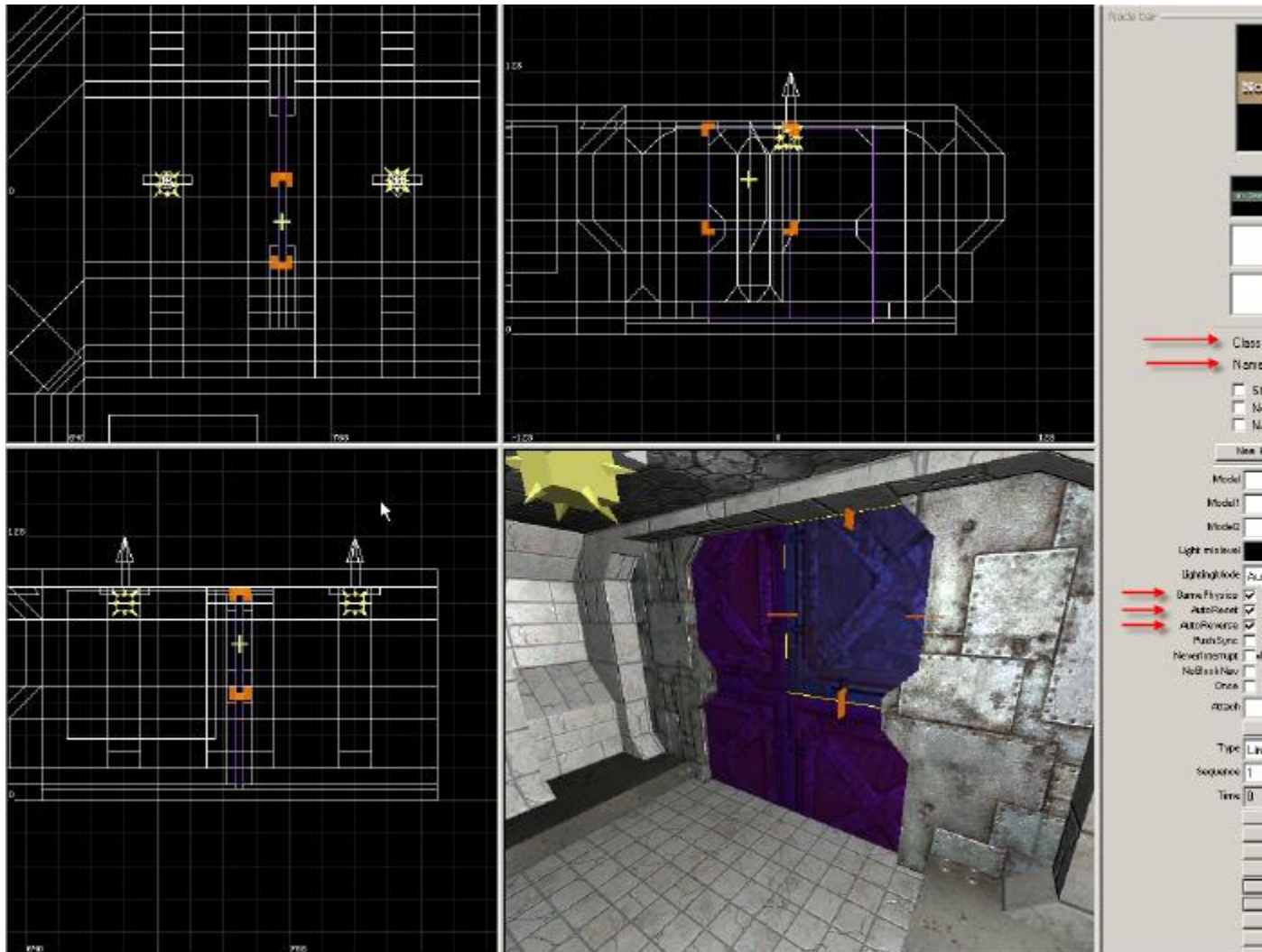
In the next step I created 4 brushes that the door will be made of.  
While doing so I named all parts individually (door01\_ur, door01\_ul, door01\_dr, door01\_dl)  
and selected  
a new object class for it (engine\_path).

Also the following flags:

- GamePhysics (to make it move at all)
- AutoReset (so it can be used more than just once)
- AutoReverse (this will make the door open again if something is blocking it, i.e. the player's head)

You can easily set these values with having all 4 brushes selected (hold STRG and click on every part).





## Pathfinding

What follows next is the movement of the door.

This depends on what your door looks like since every part may move at it's own speed and time.

In my examples all 4 parts will move at the same time and speed, but in different directions (the following pictures will show this)

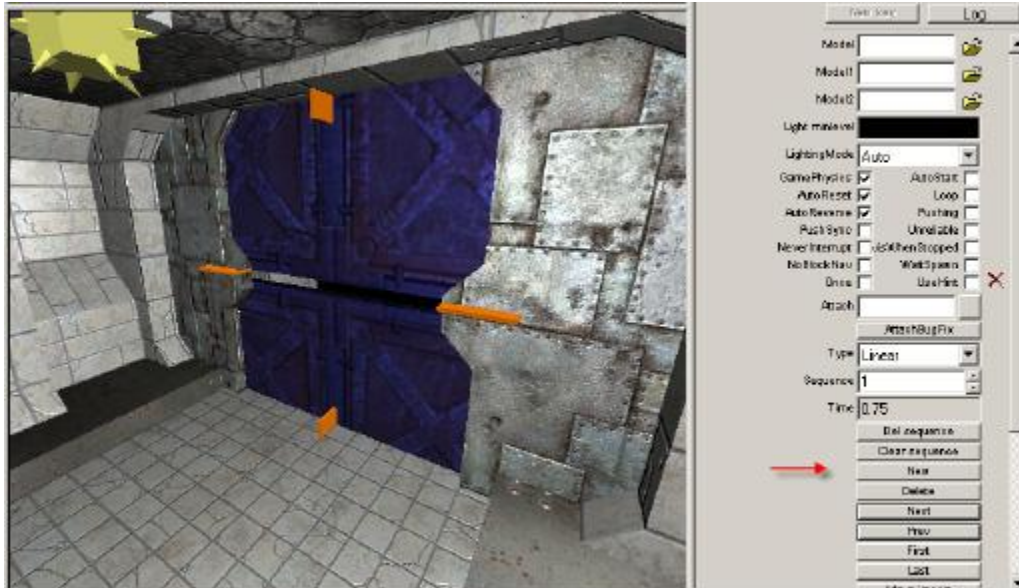
Keep in mind that we are setting something like waypoints by clicking on the "new" button in the Node bar.

For the parts to move at the same time and speed, watch out that the time index on every doorpart per waypoint is identical.

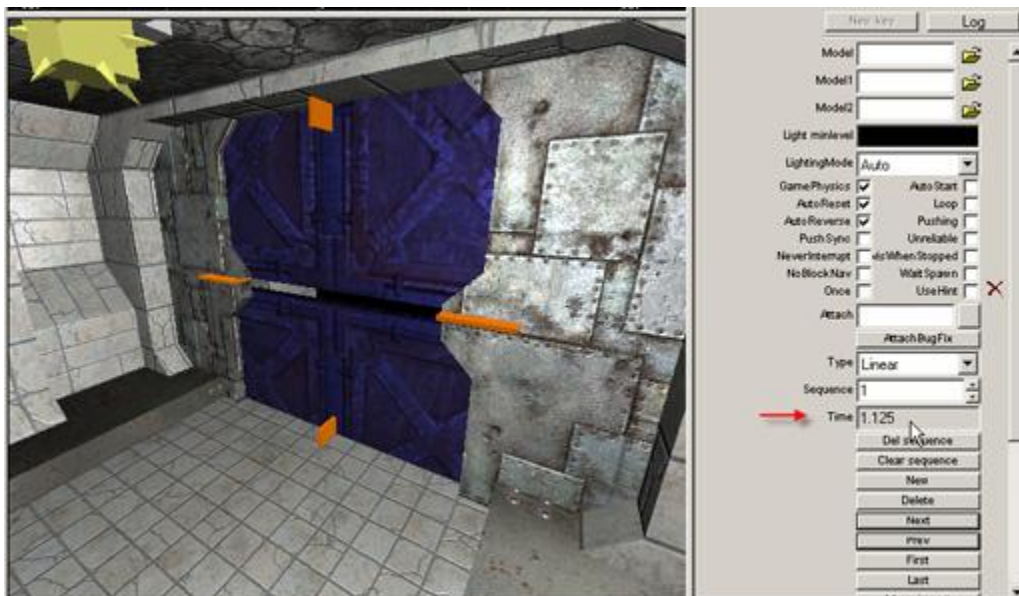
Otherwise the upper-left part may start moving before the upper-right part...etc. (except you want it this way)

This is the door on timeindex 0.75:

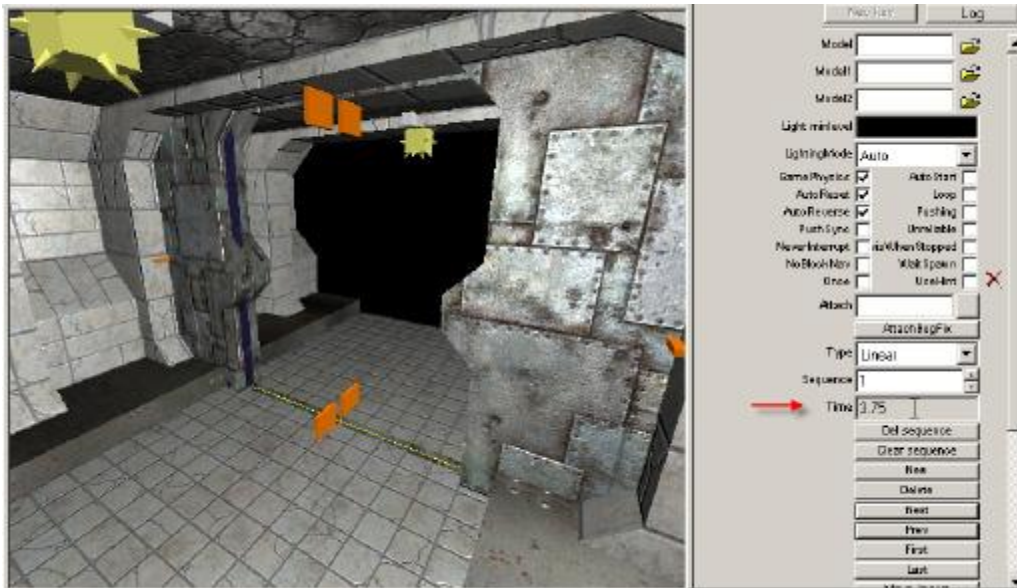




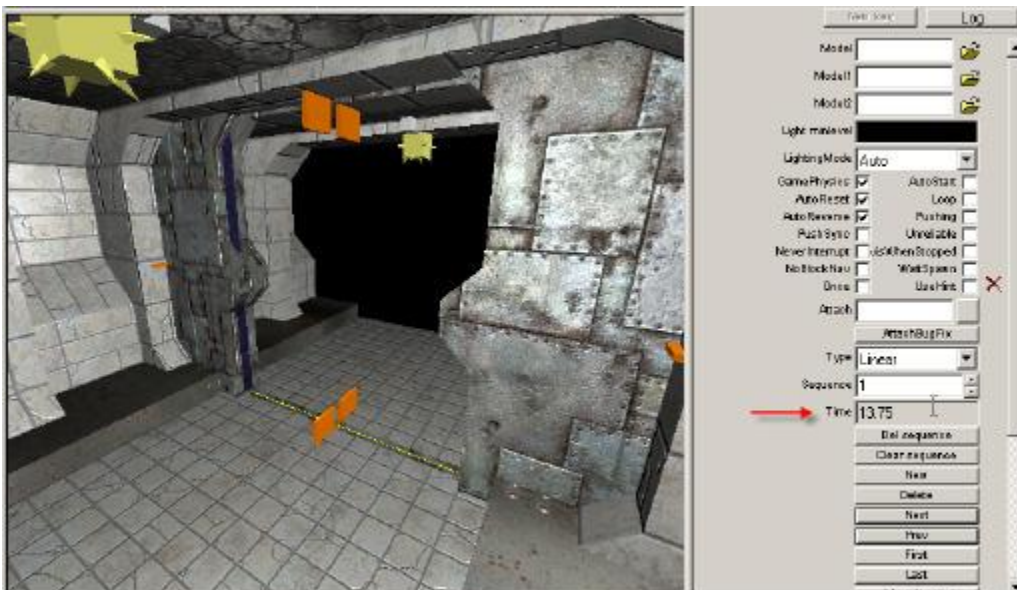
So it opened a bit and I want it to stay there for a glimpse.  
 Next I will add another waypoint at timeindex 1.125, but I will not move the door at all.  
 The result is that the door will wait in that position for 0.375 seconds.



Adding another waypoint at timeindex 3.75, I move the door parts into the full-open position:

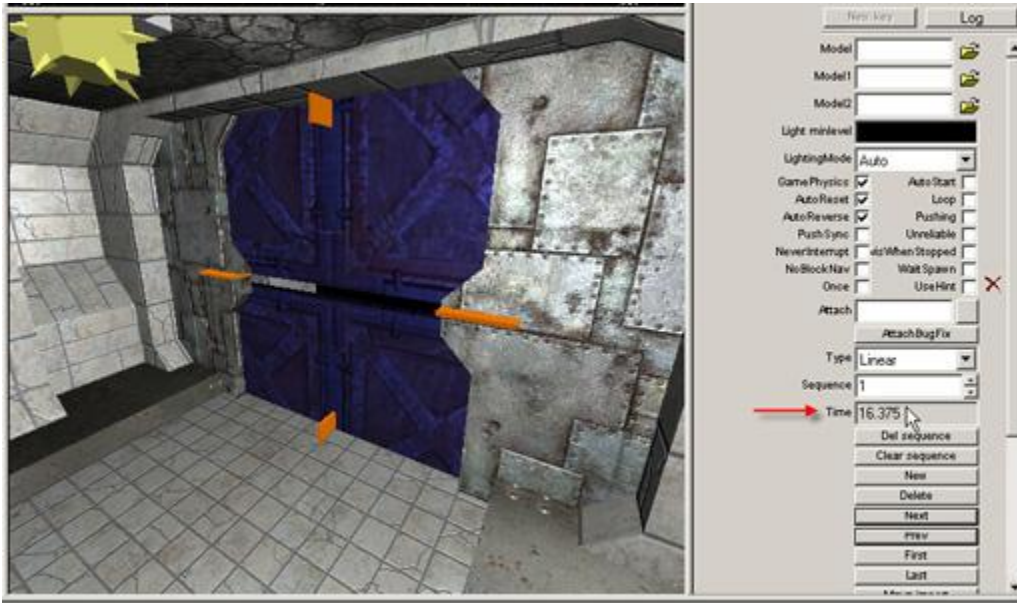


There it shall wait again for 10 seconds.  
So, as before, another waypoint is made, with the same positioning at TI 13.75



What comes now is pretty simple, since it is the closing animation.  
Basically it gets the same positions as in the opening animation and the timing should also be the same,  
meaning that you should use the same delays and times for closing as opening, otherwise it might appear strange ingame.

At TI 16.375 we have:



Then again another waypoint at TI 16.75 without moving (I don't post this again, it looks the same way as on TI 0.75) and it will be fully closed again on TI 17.5.

## Scripting

Everything is set and just waits for being triggered.

We will do that right now by adding a button on either side of the door, but wait...

The Buttons would need to trigger each doorpart on its own which is not really the best practice.

If you would want to add or delete something, you would have to edit both buttons all the time.

What we need is something that will trigger the doorgroup for us, so let's add a new object, the "engine\_script".

The "engine\_script" is actually something like a little brother of the "engine\_path". It is not able to set paths where things can move around (like our doorparts), but it can send "Timed Messages", like it's bigger brother.

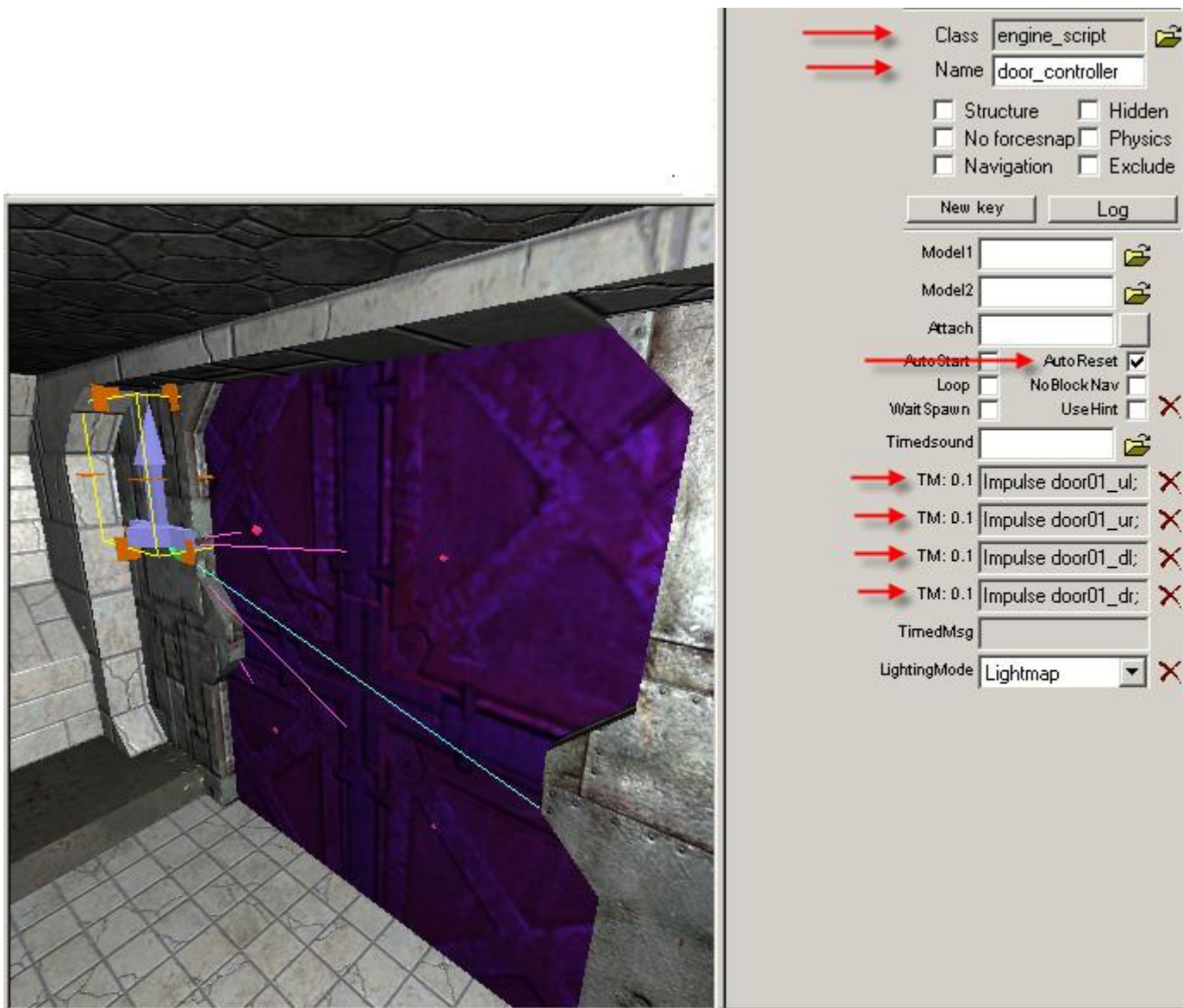
So we put it next to our door, name it "door\_controller" and switch on the "AutoReset" flag. (so it will send it's "Timed Messages" as often as we push the button later)

Next we click on "TimedMsg" in the Node bar and add an "Impulse 1" on TimeIndex 0.1 with "door01\_ul" as target.

Click on ok and repeat this with the other 3 doorparts (door01\_ur, ...).

If you want to add anything to the door that shall be triggered when somebody pushes the button, all you have to do is to add it to "door\_controller" and it's done.





Next we finally add the two buttons that will activate our "door\_controller".

So hit <Insert> on your keyboard and select "acs\_doorbell".

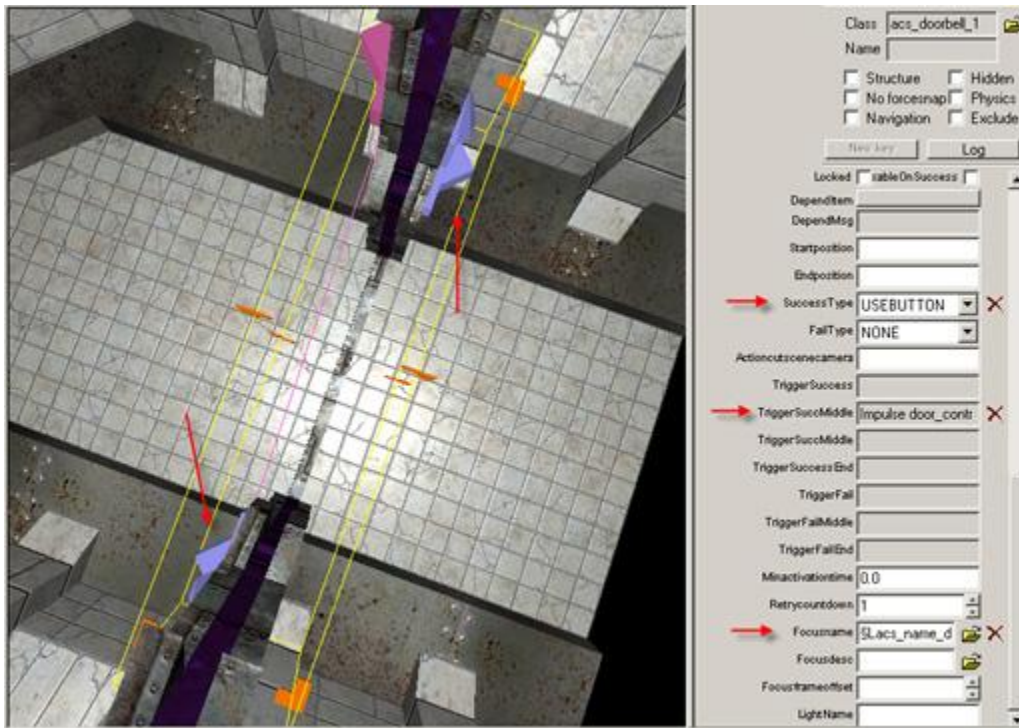
(I just liked the look of it, basically you can use a standard "acs" object and assign any model you want to it)

Set the SuccessType to "USEBUTTON" and as TriggerSuccMiddle an "Impulse 1" to "door\_controller".

Be sure to select a Focusname that fits your needs. (I took "\$Lacs\_name\_door\_button")

Copy the whole button and move it to the other side, so you don't have to enter all the params again.

(I took the roof out and shot the pic from the top so you can see both buttons here)



Well, done 😊

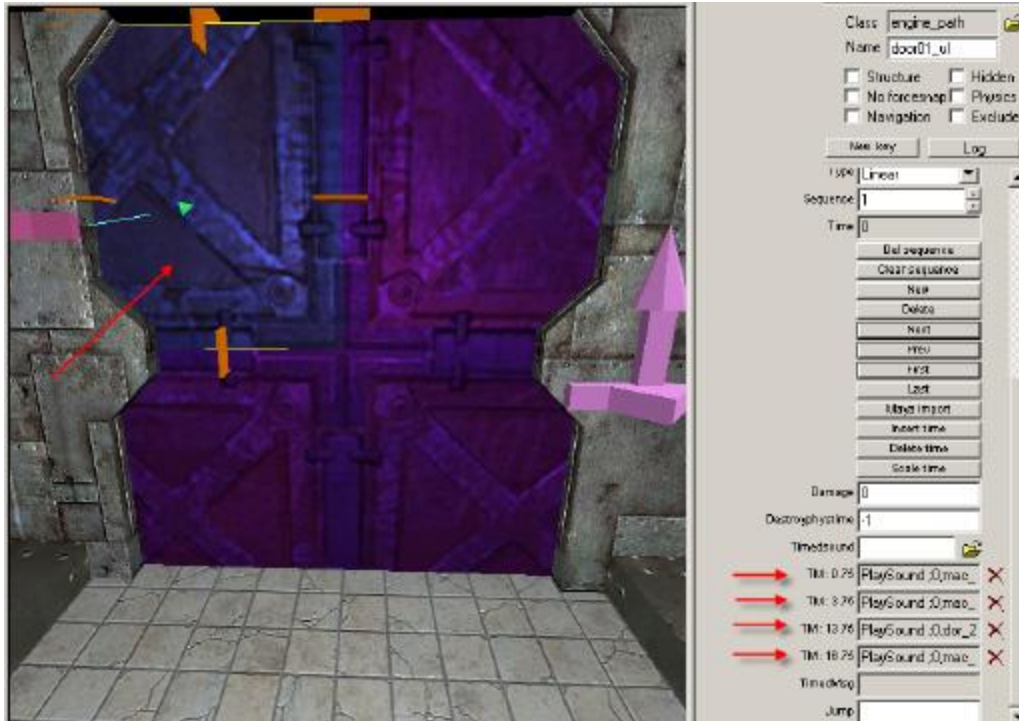
## Sound

Some fine tuning is what the door still needs now.

I selected one of the doorparts and added some TimedMessages that do "PlaySound" and bring up some metallic noises according to the moves.

If you would assign the sounds to all 4 parts, then you really have 4 sounds playing simultaneously which is sometimes irritating to the player if he is using dolby-surround and gets 4 different sources. Apart from that, it's just a waste of system resources to play the same thing 4 times at once.

As you can see, I used the same TI's as the path-waypoints. These just fitted into the animation, basically you can choose any time you want the sound to be heard.



## Conclusion

- We have a door that consists of 4 moving parts (engine\_path).
- Every part has some waypoints that know where the part has to be at that time.
- The engine\_script "door\_controller" will trigger all 4 parts if it gets an "Impulse 1"
- Two buttons (acs\_doorbell) were added that are able to send an "Impulse 1" to the "door\_controller" when the player pushes it.
- Sounds will add some atmosphere to the animation as the parts move.

Hope you enjoyed this tutorial and could find some useful info.

Greets,  
Dan

# Ogier Tutorial: Converting Maps from Worldcraft/Hammer to Ogier

Written by "Nemesis"

Hi there, I just wanted to provide this community with a quick tutorial concerning mapping with Worldcraft/Hammer for Chronicles of Riddick. First, there are some things that need to be mentioned:

**Versions** This Tutorial is written for users of Worldcraft/Hammer 3.4, I still have no idea if map-exporting can be done with the Hammer 4.0 that shipped with HL2-SDK. If you know, please drop a line here. The GDK-Radiant should also be capable of saving maps in HL-style though i have not tested this yet.

**Size** Mappers, who are quite familiar with Hammer will soon notice, that Richard is somewhat smaller than Gordon, e.g. if you plan to make stairs, be aware that Richard is capable of climbing heights of 12 max, Gordon could do 24, i guess. (With the use of ledges this can be easily compensated though). Be also aware that in most cases Richard will not feel like a dwarf when wandering floors with the height of 96, Gordon somehow needs ceilings of 128 to feel comfortable (I built many maps that way, but if you built cs-maps with floor height of 96 and done well with that, go on) To conclude, try to build somewhat smaller, this will save time.

**Entities** Concerning entities? Erase everything. If your favorite HL/CS is to run in Riddick, no class or world entities have to remain in the map once it is exported. Also delete or replace this `r_speeds` saving `func_walls`. You know what i mean with "everything".

So, once you have scaled your map and deleted all entities, we can go on. I presume that Worldcraft/Hammer is running and you can see your beautiful map that you want to export to Riddick. Just click on File --> export to .MAP  
That's it for Hammer. Open the .MAP with Ogier and notice that all Textures are messed up.

(1) Right click and draw a selection box around the whole map. Or press F3 to do the same.

(2) Then press "the key under ESC" to open the Texture browser. Select the "nodraw" Texture from the "3.Special" Texture container.

(3) Switch to the Texture tool, (Alt+5)

(4) hit "x" to select all faces and then "k" to paste "nodraw" on all of them. You should notice that the formerly grey brushes turned to somewhat green now.  
But still, the Texture scaling is pretty messed up.

(5) Hit ENTER to open the Texture tool.

In the Texture tool, write "0" into the two offset spaces, "1" into the two scaling spaces and "0" in the rotating space. Click OK.

(6) Nothing seems to happen. Hit ENTER again, notice that the "1" in the scaling spaces changed into "-2147483.75". Weird. Replace them once again with a "1", click OK, and everything will be fine. (For a rather cosmetic approach you can also set Texture scaling to "0.25", because Riddick makes often use of 512x512 sized Textures, but this can be done another time)

(7) You should now see the "nodraw" Texture on all of your faces. All in-game faces should now be textured. (Remember Zoner? He invented the "NULL" texture for the use with Worldcraft. "Nodraw" does the same). But texturing is covered elsewhere in this forum.

(8.) Save the map as .XLS (Rename it, if its now named mymap.map.xls)

(9) At least we have to get rid of some HL-referring strings in the map data, which can't be seen IN Ogier but in a Text editor:

Open a Text editor and look at the beginning strings of the MAP data

**Quote:**

```
{
"CLASSNAME" "BRUSH"
"OGR_CAMERA0" "-0.294922,0.5,-0.132324"
"OGR_TRANSLATE0" "5.781945,71.026421,52.415737"
{
"CLASSNAME" "WORLDSPAWN"
"SOUNDS" "1"
"MAXRANGE" "4096"
"MAPVERSION" "220"
"WAD" "\\half-life\\valve\\halflife.wad"
{
"CLASSNAME" "BRUSH"
"BRUSHFLAGS" "32"
"BRUSH_PLANE_0" "2, 216,492,32, 216,436,32, 208,436,40, *NODRAW, 0, 0,0, 0, 1,1"
"BRUSH_PLANE_1" "2, 208,492,16, 208,436,16, 216,436,16, *NODRAW, 0, 0,0, 0, 1,1"
"BRUSH_PLANE_2" "2, 216,436,16, 208,436,16, 208,436,40, *NODRAW, 0, 0,0, 0, 1,1"
"BRUSH_PLANE_3" "2, 216,492,32, 208,492,40, 208,492,16, *NODRAW, 0, 0,0, 0, 1,1"
"BRUSH_PLANE_4" "2, 208,436,16, 208,492,16, 208,492,40, *NODRAW, 0, 0,0, 0, 1,1"
"BRUSH_PLANE_5" "2, 216,436,32, 216,492,32, 216,492,16, *NODRAW, 0, 0,0, 0, 1,1"
}
}
```

In the second line the "CLASSNAME" "BRUSH" must be changed to "CLASSNAME" "WORLDSPAWN"  
"OGR\_CAMERA0" "-0.294922,0.5,-0.132324" remains unchanged, this is just the view position and angle of the 3D-view in Ogier stored in the map. Same with "OGR\_TRANSLATE0".  
GET RID of the next Strings:

**Quote:**

```
"CLASSNAME" "WORLDSPAWN"
"SOUNDS" "1"
"MAXRANGE" "4096"
"MAPVERSION" "220"
"WAD" "\\half-life\\valve\\halflife.wad"
```

These should be deleted, for they are for HL.

Save, compile, enjoy.



PS: Still not seeing your beloved map in Riddick? Don't forget to put a "structure" brush into the map with Ogier. Common mistake, if you ask me 😊

CREDITS go to dnadns who has helped me much with my first steps with Ogier, thanks mate!