Level Editor Basic Guide

Version 1.0

By Daniel Grayshon

Table of Contents

Introduction

Welcome to this guide to the Turok level editor!

This guide will be a basic introduction to the editor. It will get you familiar with the interface, and how the editor functions.

We will create a small space, laying out meshes and creating a play space for your player to run around in. You will also learn about sectors, navigational meshes or 'Navmeshes', Events, and how to trigger them as well as things like secret locations, warping your player around the level, and potentially in-between levels as well as getting the player back to the Turok title screen once they have finished your campaign.

You'll also learn about the Steam Workshop, and how to package up your mod into something that can be uploaded to the Workshop. People will be able to download what you create.

This editor works very much like other editors you may have used in the past like Unreal or Unity. The "world" the player will run around in initially starts off as a big empty void of nothingness. It is down to you to fill that void with an environment, place enemy locations, create events, item pick ups, and many other things.

The editor has been built from the ground up to allow the community to create levels for Turok, and along the way whilst it was being created, we found that the original developers had to take some short-cuts in order to fit the game on to a Nintendo 64 cartridge. These short-cuts ended up making things confusing when trying to determine what does what. As a result of this, we couldn't work out what some of the values actually meant to the game engine.

Instead of giving these values names, they are simply known as "Arguments" or, in the editor, "Arg" which you will see later in the guide. They are numbered separately, and the guide will be as clear as it can in informing you about what values to put into what Arguments.

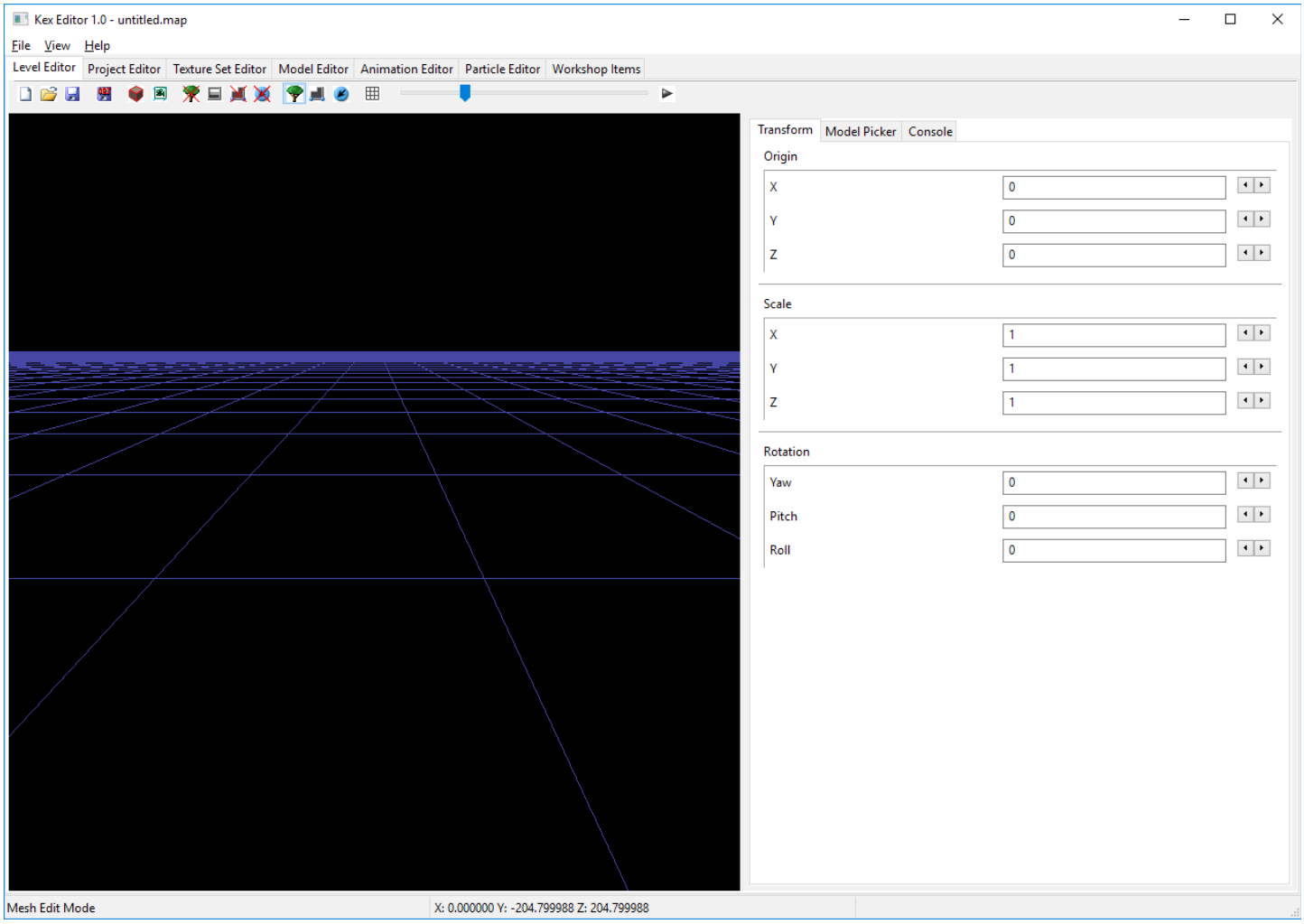
This is only a basic guide. For more advanced information about scripting, importing models, and customized sound effects, please visit the Turok: Dinosaur Hunter Steam community hub page. The "Modding" section of the hub will be filled with all the information you could need.

I have also provided a tutorial level file that is provided with the editor. **Tutoriallevel.map** is the end result of the creation of this guide. If you ever get stuck, take a look at the provided tutoriallevel.map file to get a clearer idea of what to do.

Have fun with the tools that have been provided. We can't wait to see what interesting levels you come up with!

With all this in mind, let's dive right in and get making our very first level with the editor.

The Level Editor



This level editor is all you need to get started with building your own custom campaigns for Turok: Dinosaur Hunter. This editor will not allow you to make models for the game. Making models is achieved by using a 3D modelling package such as 3DS Max (http://autodesk.com/3dsmax) or Blender (https://www.blender.org)

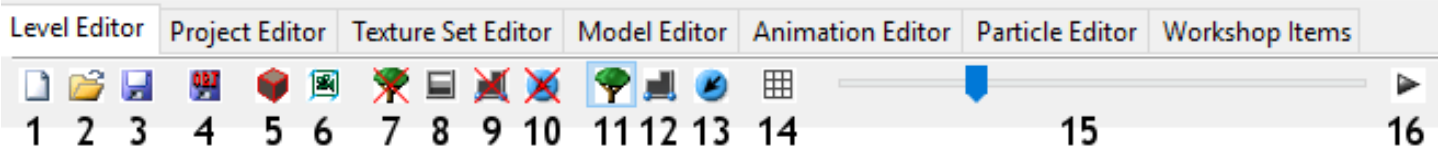
For the editor to function properly, you will need to set the "seta developer" value to 1 in your config.cfg file. You can do this by opening the file in NotePad or NotePad++ (https://notepad-plus-plus.org) , finding the line "seta developer" and setting the corresponding number to 1

You will also need to extract the contents of your game.kpf file into the Turok directory. This can easily be done with a Zip file manager tool such as 7-Zip. http://www.7-zip.org

(Night Dive Studios, DreamWorks Interactive, Valve Corporation or any individual employee from those companies is NOT and CANNOT be held responsible for any software and/or hardware damage that could (but probably won't) happen as a result of installing 3rd party applications on your computer. By reading this, you agree to this.)

Simply extract the contents, and you'll be much better equipped now for editing game files.

"Level Editor" Interface



1. **Create a New Level** -> Clears everything from the editor, providing you with a fresh environment to work in.
2. **Open a Map File** -> Will open a previously saved map file for editing.
3. **Save Map** -> Saves your current project. User will be prompted to save their map to a location every time.
4. **Export Scene to .OBJ** -> Waveform Object Model File.

5. **Toggle Wireframe** -> Removes graphics, allowing you to see just wireframe.
6. **Toggle Ortho View** ->

7. **Hide Static Meshes** -> Toggles all Static Meshes in the editor view window.
8. **Hide Fog** -> Toggles fog in the editor view window. Use this to see how your map looks with no extended draw distance.
9. **Hide Sectors** -> Only works when Sector Edit Mode (12) is enabled. Toggles sector viewing, allowing you to see edges more clearly.
10. **Hide Actors** -> Will hide Actors. Actors consist of anything that moves: enemies, destructible trees, pick-ups, blue portals and other things.

11. **Edit Mesh Mode** -> This will turn on Edit Mesh Mode.
12. **Sector Edit Mode** -> This will turn on Sector Edit Mode. Hide Sectors (9) will only work in this mode.
13. **Actor Edit Mode** -> This will turn on Actor Edit Mode.

14. **Toggle Grid Snapping** -> This will toggle grid snapping, allowing for a less precise but much easier alignment of floor and wall tiles.

15. **Draw Distance** -> This will affect how much you can see in the Editor View window. This is not related to fog.

16. **Test Level** -> This will save the currently open map to a 'editorTemp.map' file in the root game directory and will proceed to launch the game. No original files will be overwritten. For this to work properly, you must have "Seta_developer" set to "1" in your config.cfg file.

Controls

Here is a list of some of the controls for moving around the editor as well as some Quick Key presses for enabling certain modes.

In the Editor window:

Left Click and Hold	-	Will move the camera around, sticking to its current height.
Right Click and Hold	-	Take control of the camera to look around “First Person” style.
Left Click + Right Click and Hold	-	Will move the camera up and down as well as left and right.
Control+Click	-	Will select something that the user clicks on. You must be in the correct “mode” to accomplish certain selections (e.g. selecting a tree will only work in Mesh Edit Mode). You can only select one item at a time.
Control+Shift+Click	-	The same functionality as Control+Click instead you can select multiple items instead of just one.
Escape	-	Will deselect anything you might have selected. Getting into the habit of pressing Escape regularly is a good thing if you are unsure if you have something selected or not.
End	-	Will snap anything currently selected to the floor. Very useful for grounding objects such as trees, rocks, bushes, collectables and meshes.
Space	-	When something is selected, it will change the Gizmo Axis to a different mode. Modes are “Move/Rotate/Scale” in that order.
Control+D (Othro Mode only)	-	Enables Shape Draw Mode.
F1	-	Enables/Disables Vertex Plot Mode.
F2 (When a Sector is selected)	-	Enables/Disables Sector Creation Mode.

Building your First Map

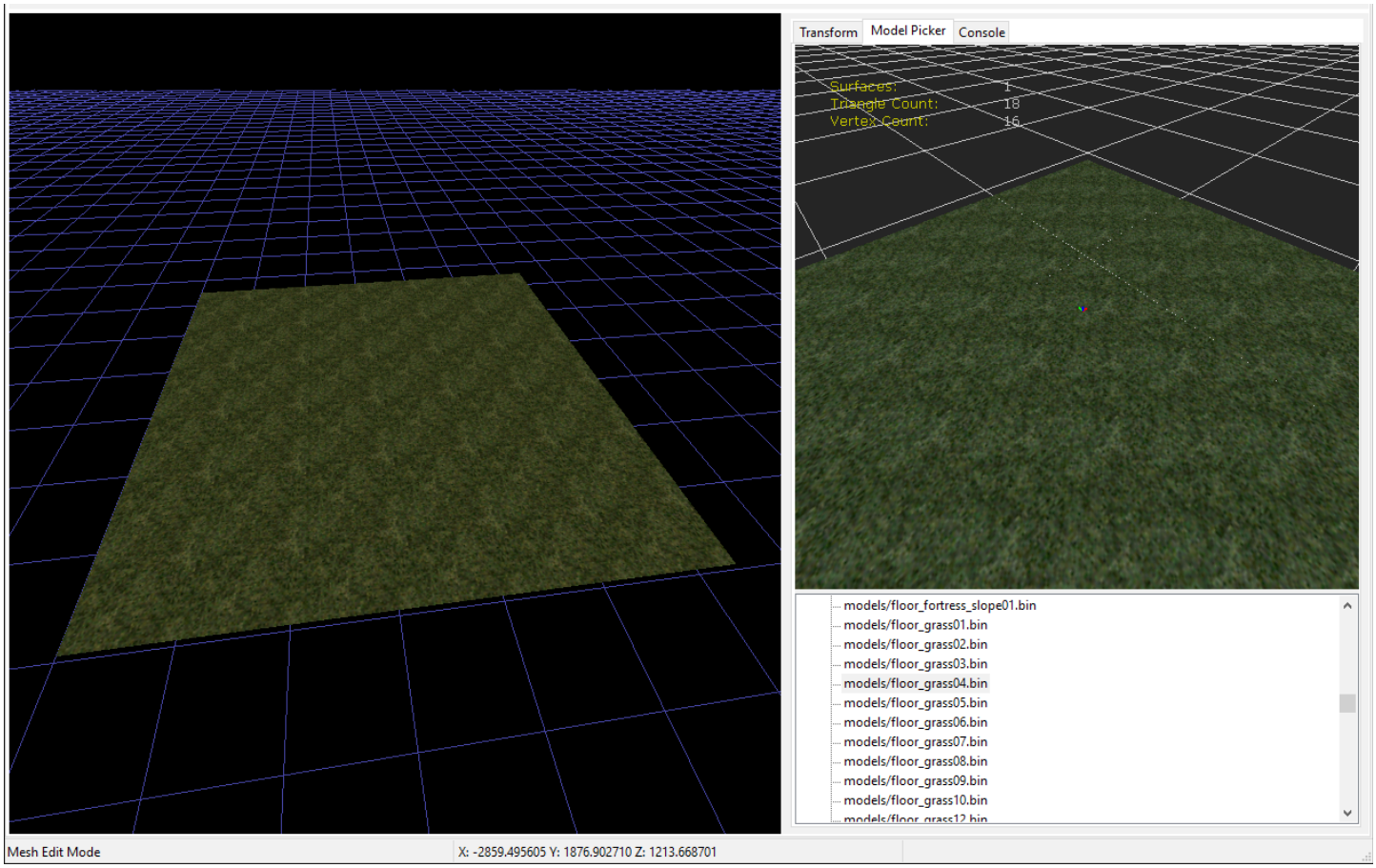
We have a fresh new editor session open. Let's start off by creating some floors for the player to walk around on.

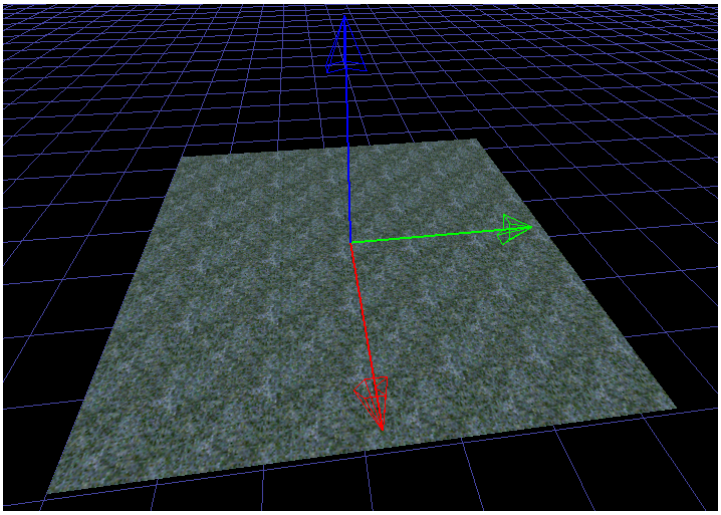
Floors in Turok (grass, stone, metal grating etc.) are composed of meshes. A floor will be built that is composed entirely of meshes, We won't do anything too advanced for our first floor so we can go ahead and make a nice grassy floor for our first area.

To set down a floor piece (I'll refer to them as “tiles” from now on) we need to select one from the Model Picker. The Model Picker is found on the set of tabs to the right of the Editor window. From here, we need to open the Models category and scroll all the way down until we get to the “Floor” tiles (names of floor files start with the word “floor” to help you find them faster)

Let's choose “**models/floor_grass04.bin**”. We can check what model we have selected by looking at the grey window above the list of models. This window has the same controls for movement and looking as the main editor window so be sure to move the camera up and the angle of it down so you can see what you have selected.

Once you have **floor_grass04.bin** selected, you can right-click on the editor view and click “Insert mesh here” and the tile will get placed into the world.





Now we have what will be our floor for starting on. Hold down Control and click on the floor tile we just made to select it.

When an item is selected in the editor window, it will change colour slightly and you will see some arrows appear in the centre of the selected objects. These arrows can be used to move the selected item around the editor by clicking on the corresponding arrow moving the mouse in that direction.

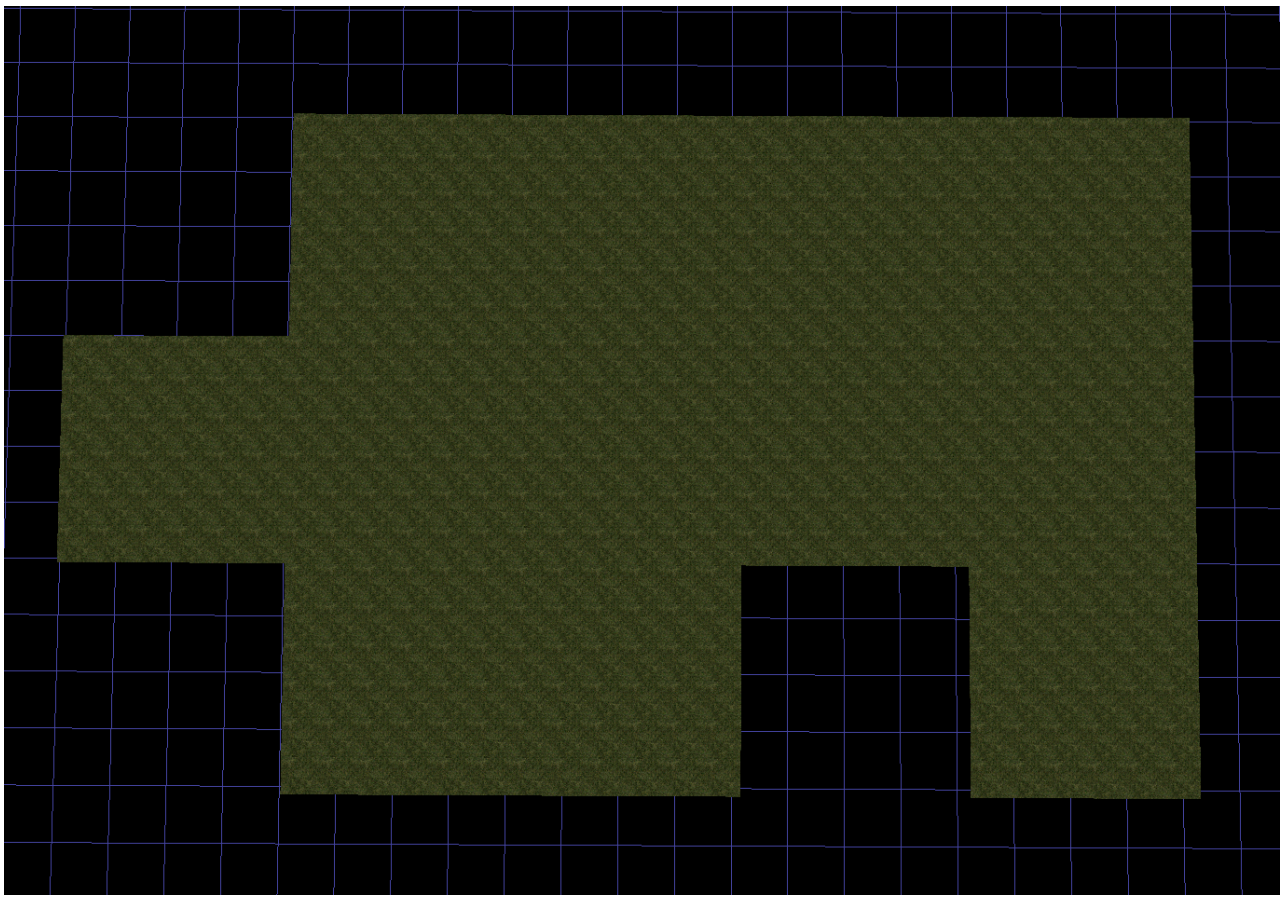
The Blue Arrow moves the item vertically up or down, Green and Red will move the item's in their pointed directions (back and forth)

This is also known as the “Gizmo Axis”

Now we have our first floor tile but we want to make another one quickly to put next to it. That can easily be done by right-clicking on the tile, clicking Copy and then immediately clicking Paste. At first nothing will have looked to have happened but by moving the tile using either the Red or Green Arrows, we can quickly see that the tile was duplicated at exactly the same X, Y and Z coordinates.

Let's move our new floor tile and place it next to the original. If you have “Snap to Grid” enabled, you can quickly and easily move the tile and position it right next to the original tile. If you do not have snap to grid enabled, you will have finer control over your tile placement but this can result in gaps in the floor, walls and other parts of the world.

Let's go ahead and create some more floor tiles by duplicating our original one, feel free to go back up to the **Controls** page to learn what keyboard controls do what. I went ahead and laid out some floor tiles in a pattern shown below.



Now we have a basic floor laid out but in order to put the player in a world that is somewhat believable, we're going to have to start putting some walls up around the player!

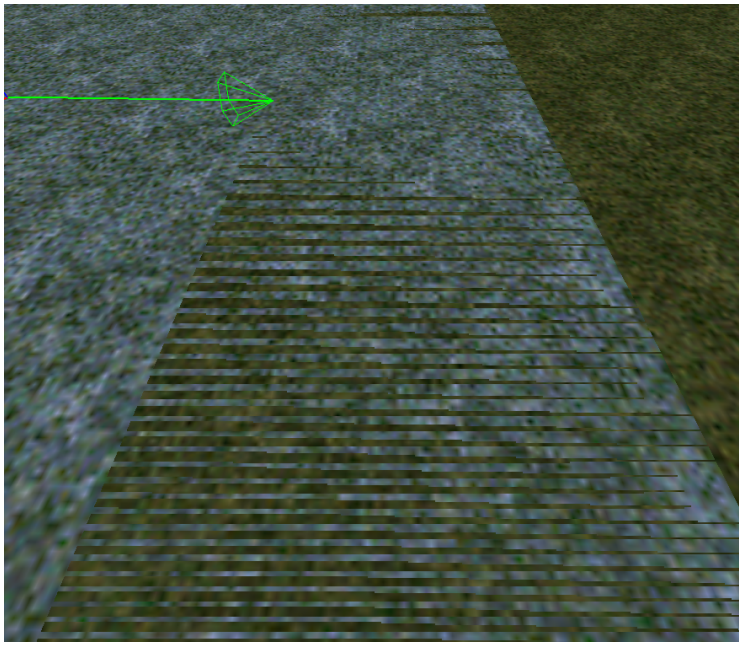
In the Model Picker window, we'll need a wall to match the environment. Canyon walls are good for grassy areas so let's put a few **"canyon_wall_corner_concave"** in there, put in one and then quickly copy and paste them to make 5 more, giving us a total of 6.

Use the on screen arrows and position a concave piece in one of the corners on the left, try to keep it flush to the ground (having **"Snap to Grid"** will REALLY help you with this) Once it is position, we'll need to put one in the other corner but we will need to turn it around so it is facing the right way first. This is where the Transform tab will come into play.

First however, we'll take a moment to discuss texture clipping and how you can avoid it from happening in your game play space.

Avoiding Texture Clipping

One thing you'll want to avoid doing from the beginning is causing a lot of Texture Clipping. Texture Clipping occurs when two surfaces occupy the same space at the same time. The graphics engine will try to render them both at the same time and will cause a graphical anomaly in the level editor. It can also appear in the game as well.



This is what it looks like.

The picture on the left demonstrates texture clipping when two floor tiles overlap. This can be easily avoided. Place your tiles and pieces of scenery sensibly. These errors will appear in game and can quickly make your work look bad.

Tips:

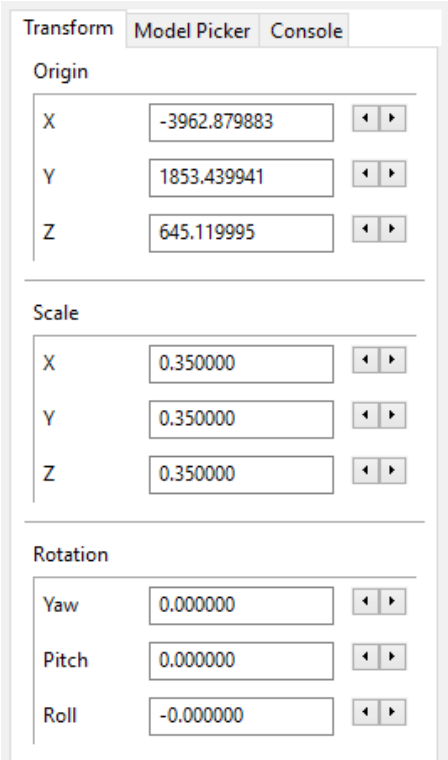
Grid snapping will solve this problem for you by having tiles snap closely to each other automatically. Turn on Grid snapping.

Move floor tiles so that parts of them are running outside of the play space. The player will not be interested in what bits of floor are outside of the map, hidden behind walls and cliffs. Move tiles and line them up so that no clipping occurs where the player can see it.

If you have no choice, try hiding the clipping textures under a nearby building. Reorganize your tiles so that the clipping will be hidden under other meshes. If all else fails, think about how to structure your level more efficiently and rebuild necessary parts of it.

This doesn't happen just with floor tiles, it can happen with walls, cliffs, ceilings and other meshes so think carefully about placement of your scenery.

The Transform Tab



The Transform tab will allow you to precisely place objects, scale them and rotate them with total control. You will spend a lot of time with this tab so it is important to become familiar with it early on.

“Origin” is the placement of the selected object in the world. You can alter it's X, Y or Z location very precisely with this group.

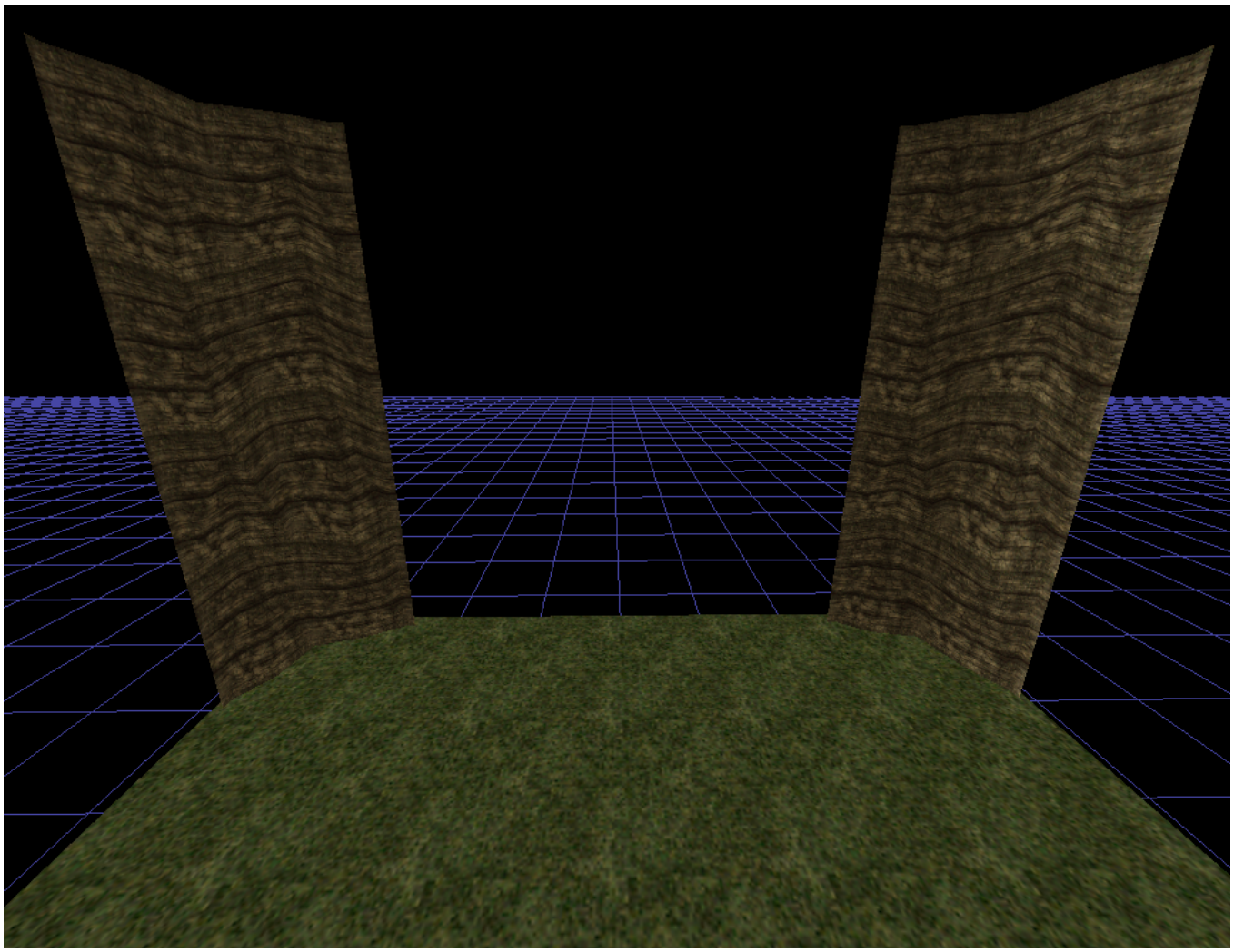
“Scale” is the size of the selected object, all objects placed in the world will have their default size given to them. It will be down to you if you want to change them or not. Scaling can be good for walls; sometimes a wall might not fully cover a gap in the scenery. Using the Scale transform, you can extend/retract the length of a wall.

“Rotation” is the way your object is facing in the world. Very important for pieces of walls, buildings, trees, rocks and other things.

Right now, we should have one of our concave pieces of wall selected and ready to be rotated so we need to change the “Yaw” value to something else.

I changed the Yaw of my concave wall to -90. Depending on how you have laid your floor pieces out, this number might have a different effect for you. Go ahead and mess around with the Yaw number to see which direction your concave wall piece rotates in.

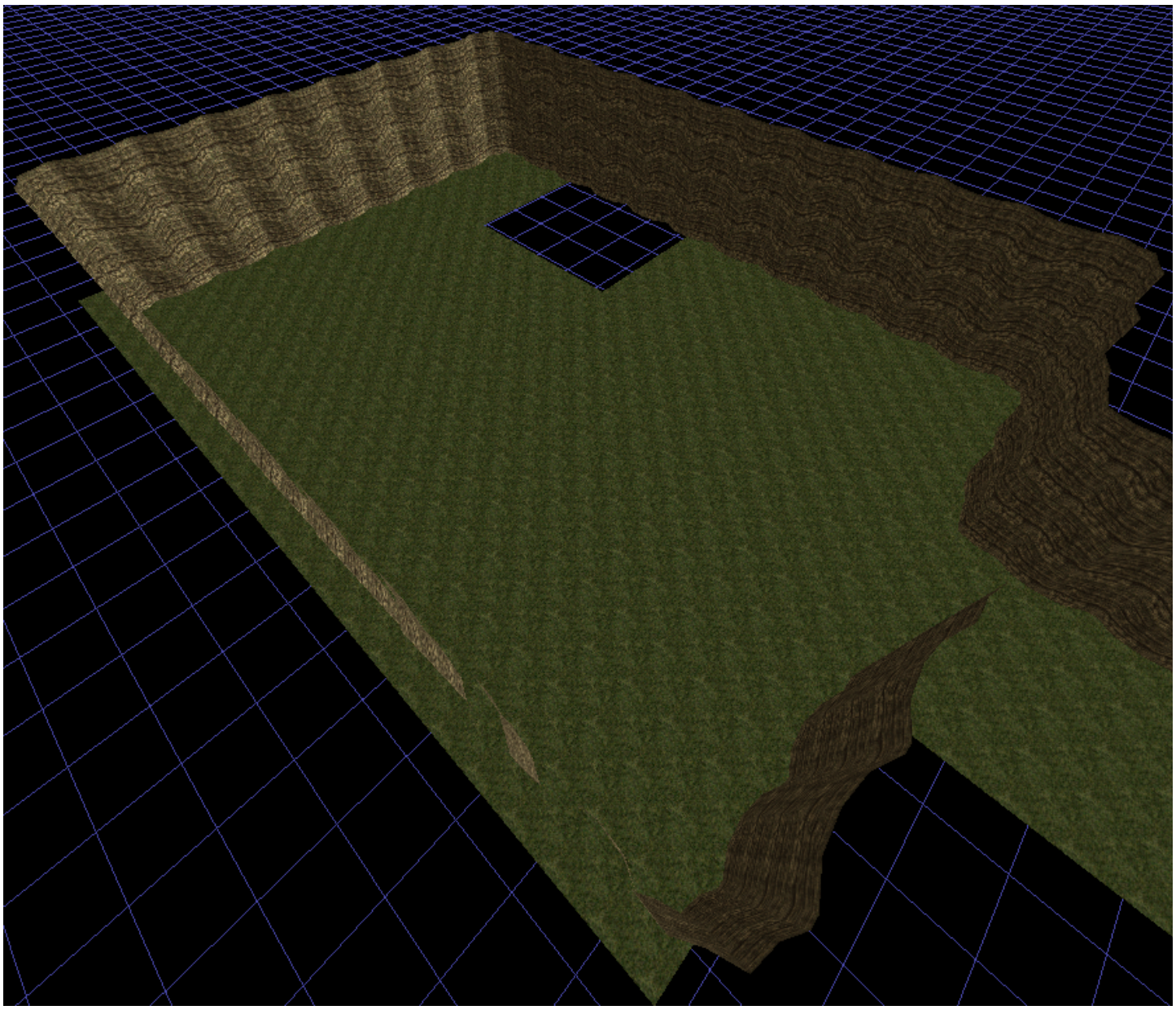
Hopefully by now, you should have something that looks like the screen shot below in your game world.



Another good piece of canyon wall to use to fill the gaps in would be “**canyon_wall_straight01**” so place one down in the world and fill the gaps in the walls here. Again, enabling “Snap to Grid” will be very useful for you with lining wall pieces up perfectly and quickly.

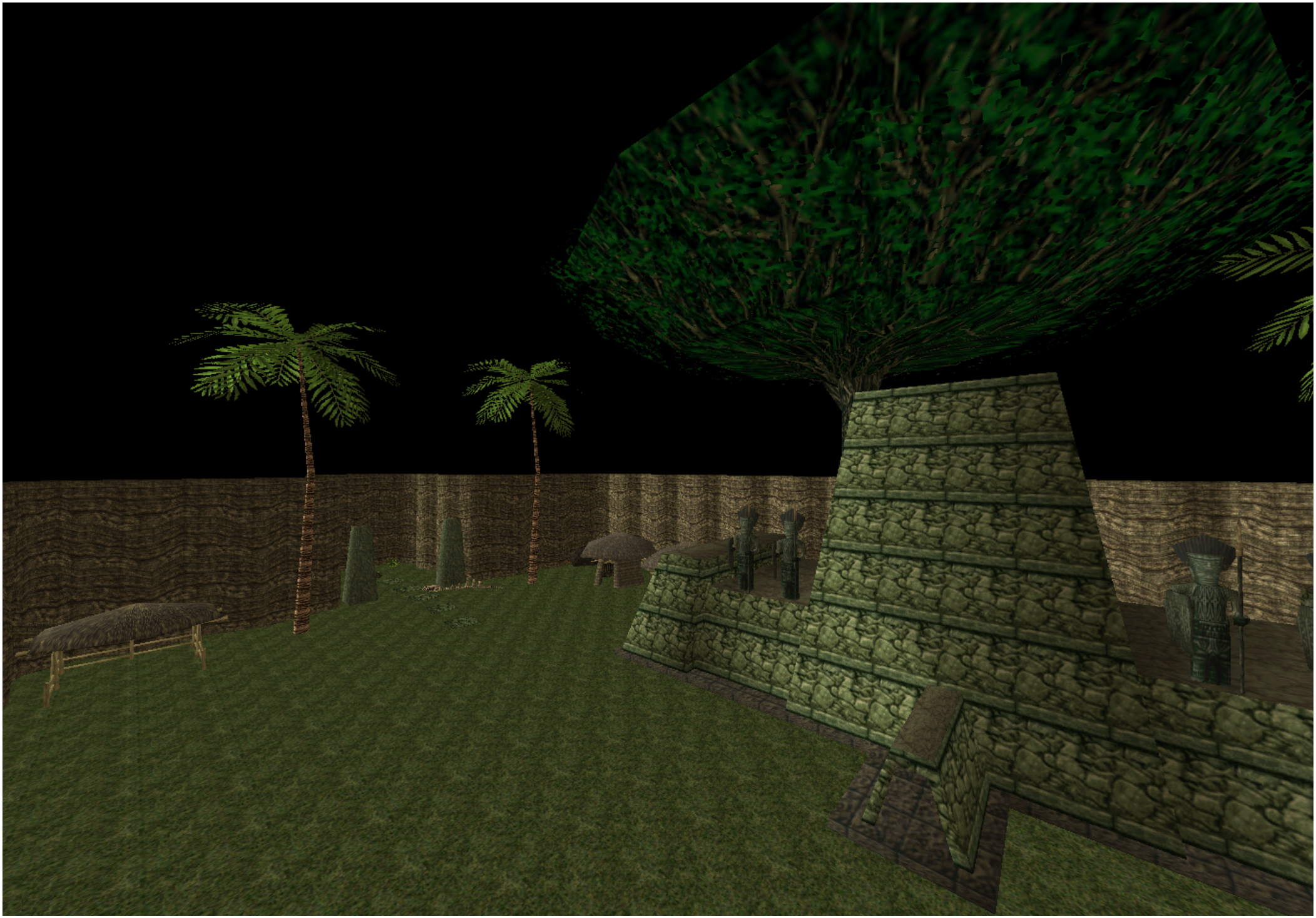
I've gone and filled these gaps in using the wall pieces “**canyon_wall_straight01.bin**”, I've also carried the wall along the sides and used the model “**canyon_wall_corner_convex**” to curve the corner around. It should all be pretty straight forwards. Once you get the hang of it, you'll be setting up walls like a pro in no time!

I have gone ahead and walled up the remaining area around the perimeter.



You may be wondering why I have left that gap in the floor in the corner over there. Don't worry about that, we'll come to it soon! But for now, this area is free to be decorated how you see fit. You can place trees, rocks, and plants down in our new environment, I'm going to put a temple down in the middle of the area but you can put whatever you want. Be creative here.

I went ahead and got creative myself; I put down a large temple in the corner which will contain some goodies for the player to pick up. I've put down some huts, some dinosaur bones, some trees and some statues on the temple walls itself as well as some sarcophagi inside the temple itself.

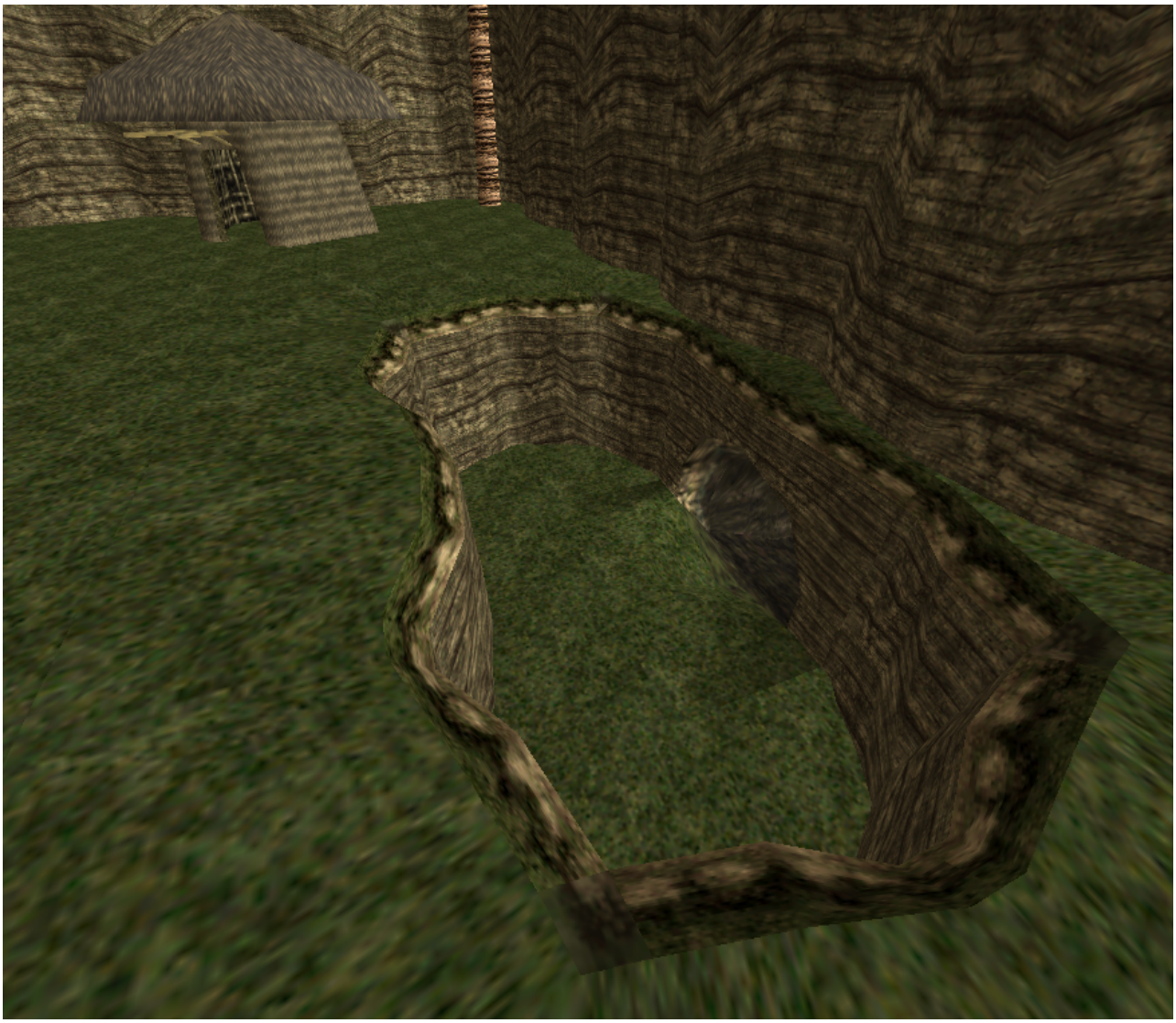


The models I have used to decorate my environment so far are:

cave_rock03
city_temple_large02
detail_bones02 – Dinosaur Bones!
detail_catacomb_coffin01 – Those are inside the temple
pillar_marble05 – I gave these a very small Z scale and have used them as paving stones at the beginning of the map.
village_treetop_bigtree01 – I made a few of these and have put them outside of the play space.
village_hut_single_01 – A few of these have been placed around the large temple.
village_stable01
and several assorted plants and things.

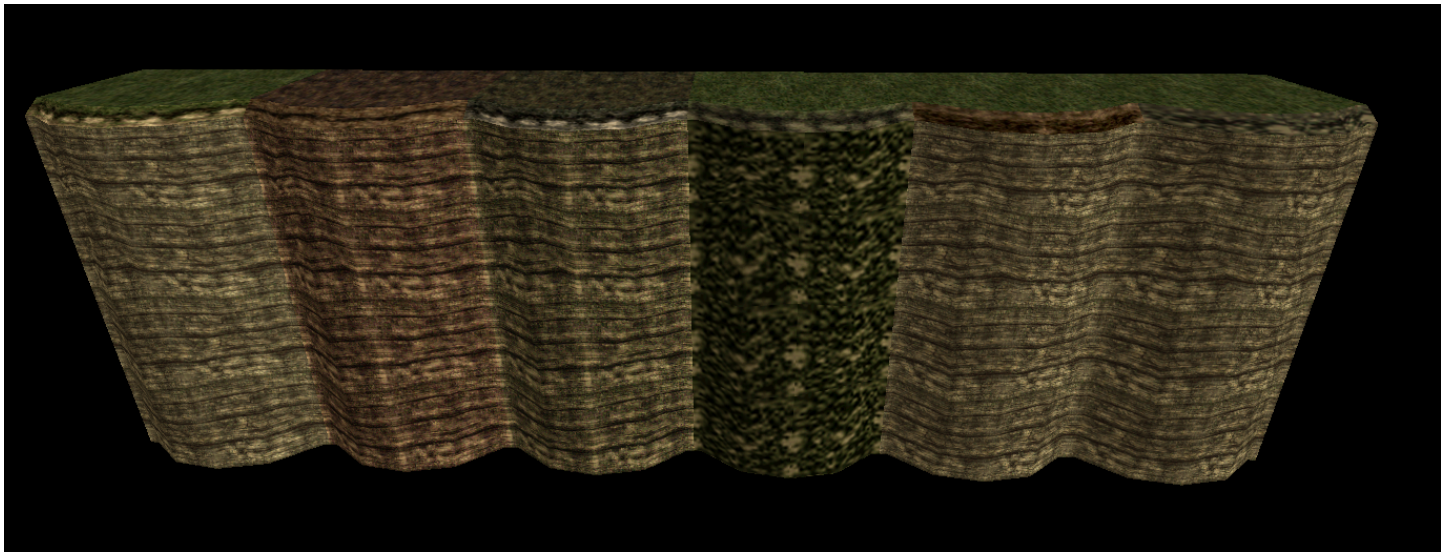
Now its time to do something about that left over hole; we need a way out of this place so it's time to put in a small cave system. I went ahead and placed some meshes in our hole we left in one of the corners of the map. I used the following meshes and adjusted the flat floor tiles accordingly to avoid clipping:

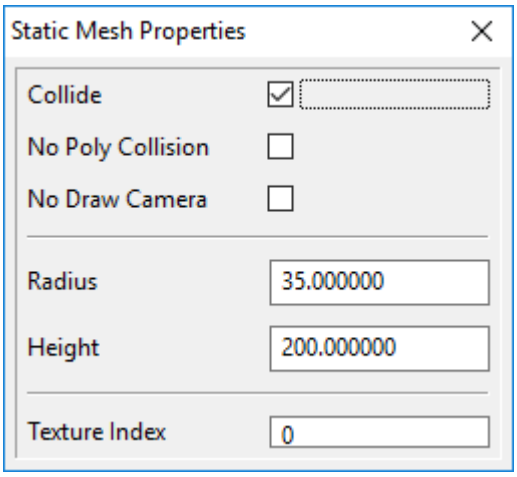
canyon_wall_flat_entrance_cave
canyon_wall_flat_concave
canyon_wall_flat_straight01



Mesh Textures and Collision

Each mesh can have multiple texture for it, instead of having a separate mesh for every texture. It can also have very basic collision. The collision is not suitable for all models and it's more suited to things like trees and boulders.





When you have a mesh highlighted, press F4 to bring up the Static Mesh Properties window. This window will allow you to set up collision.

To change the skin of a mesh, you can add a number to the Texture index, for the picture you see above, I placed the mesh down, duplicated it 5 times and went through the Texture index 0, 1, 2, 3, 4 and 5. It will change texture instantly!

As for collision, simply enable the collide flag, and in the Radius and Height fields, enter in a number. Press enter and you will see a light red wireframe pattern appear around the object. It's best used for trees and boulders but can be used for anything. A Radius of 35 and a Height of 200 for a tree is enough to stop Turok and enemies walking through it.

Something else that dictates where Turok and enemies go is “Sectoring”.

Now, it is time to discuss “Sector Editing” and getting Turok inside our play space so we can run around.

Sector Editing

What are Sectors?

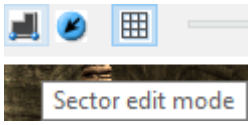
You may be familiar with something in gaming terms called a Navigational Mesh or a “Navmesh”

A Navmesh is an invisible layer on the floor that you can not normally see when you are playing the game but it provides information to the game about where the player and/or enemies can and cannot walk.

Turok uses a Navmesh system.

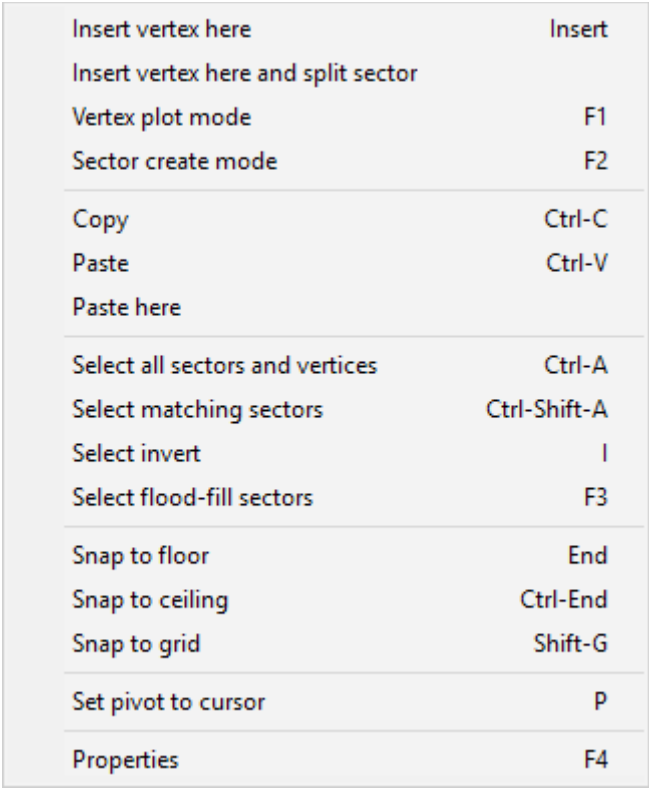
Currently, our game play space is not solid at all. Our game play space may look like an outside area with things inside it but the game doesn't know what it is, where anything starts or ends or even what objects can and cannot be walked into right now. We will have to go ahead and build our Navmesh ourselves by hand in order to define our game play space boundaries.

To start off we have to turn on “Sector Edit Mode” so we can begin our work.



We have a nice space already built so we can build our sectors right away.

If you right click in the Editor view, you will notice that our menu looks a little different now that we are in Sector Edit Mode.



The main options we need to remember here is “Vertex plot mode” and “Sector create mode”

Vertices, Sectors and applying them to your map

Vertices are simply points that join up to make a Sector. When they are placed in the world, they are very small blue-purple squares.

Sectors are Vertices that have been joined up together using the Sector edit mode. Each Sector must have a minimum of 3 Vertices but don't have a maximum limit. It is best not to create too many Vertices however as it can become confusing later down the road when you have to manage many Vertices and Sectors.

Vertex plot mode allows you to put down many Vertices in a short amount of time.

Let's try it out. Right-click and go into Vertex plot mode (or Press F1) and you'll see that when you mouse over a piece of floor, wall or mesh, it turns red.

From here, you can easily paint vertexes into the world by pressing the middle mouse button when the mouse is over a place you want to put a vertex. So lets go ahead and fill in just this beginning area. We want to keep vertexes away from the walls just enough so that we don't walk through them but not so far away from walls that it becomes unrealistic.

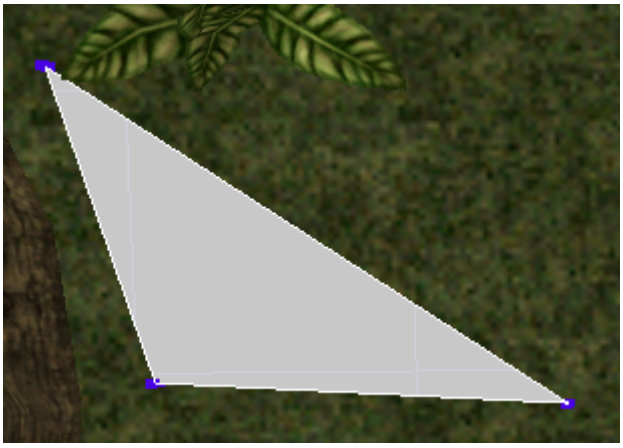
“Ortho” view can be very useful when laying out Vertexes. This layout should be good for what we need.

Now that our Vertexes are laid out for us. It's time to turn them into Sectors.

Very important: Because of the way the game engine works, always select the Vertices you want to turn into a Sector in CLOCKWISE order. Camera position doesn't really matter as long as you're looking down at the Vertices. Use Ortho view for the best experience when making and linking vertices.

Select your first Vertex using the Control key then, by holding Control and Shift, in clockwise order, select two more Vertexes nearby. Try to make a shape such as a Triangle. Vertices should turn brighter in colour when they have been selected.

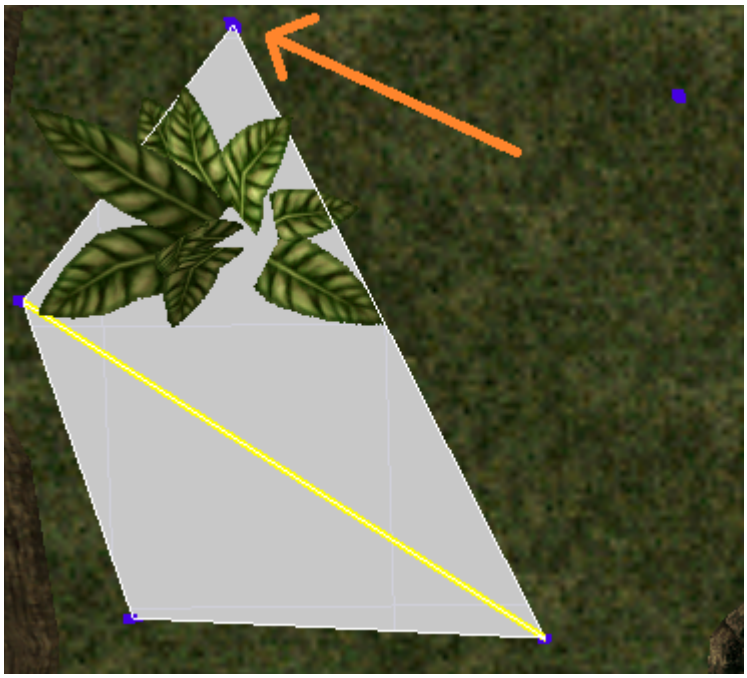
Once you have them all selected, press Backspace on the keyboard and a white shape should appear connected the selected Vertexes.



We have our first sector! Luckily we don't have to do this every single time we want to make a new sector, we can very quickly make new sectors now in “Sector create mode”.

Press F2 to enter Sector create mode. The instructions on screen will give you a good idea of what we will be doing now.

Hover your mouse over one of the edges of the new triangle we just made and it should turn yellow. When it turns yellow, press and hold the SHIFT Key to lock your selection in place. All you have to do then is click on a nearby Vertex to create a brand new sector!

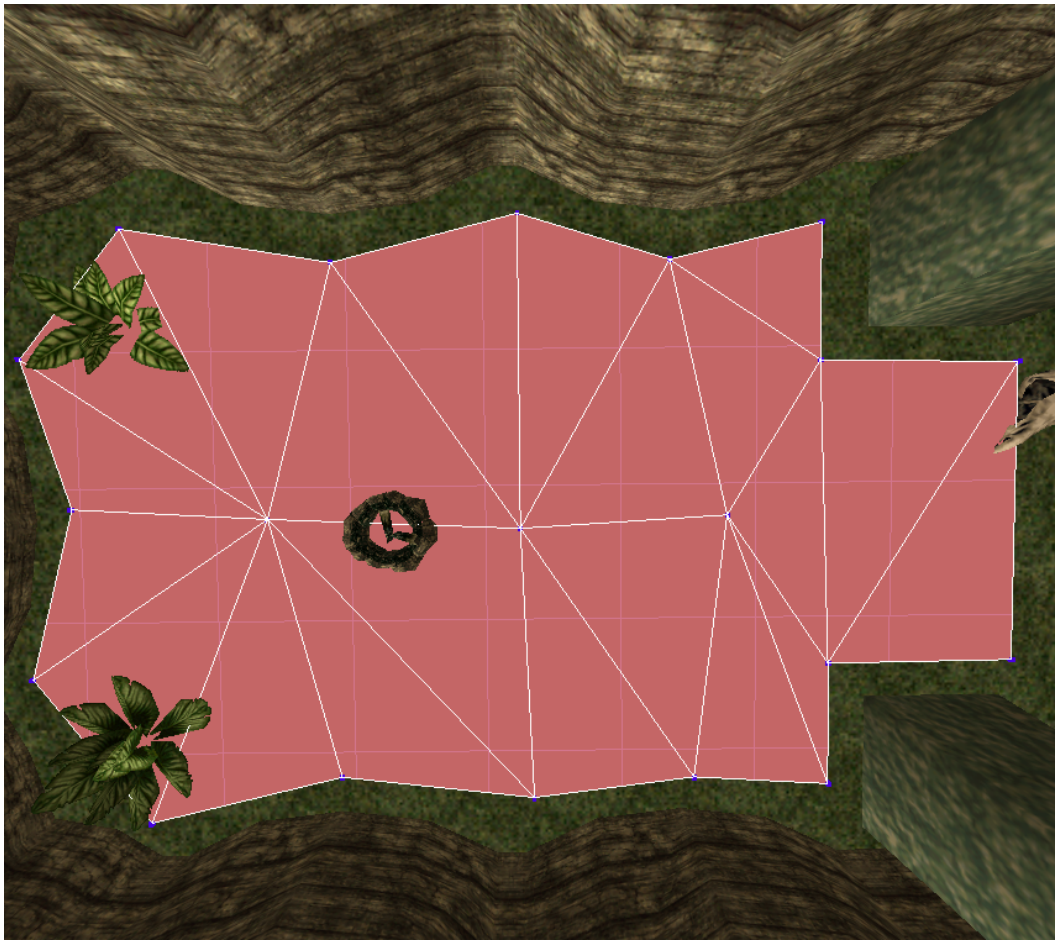


I clicked on the Vertex pointed out in orange in the above picture to create a new sector. From here you can create more and more sectors. By repeating the above method.

The triangles (or whatever shape you create) is white because it is currently selected in the editor. You have to have one Sector selected in order to enter Sector creation mode. You can select Sectors by holding CONTROL and clicking on a Sector. You can deselect all Sectors by pressing ESCAPE or one at a time by clicking on them again when you are holding down CONTROL+ALT.

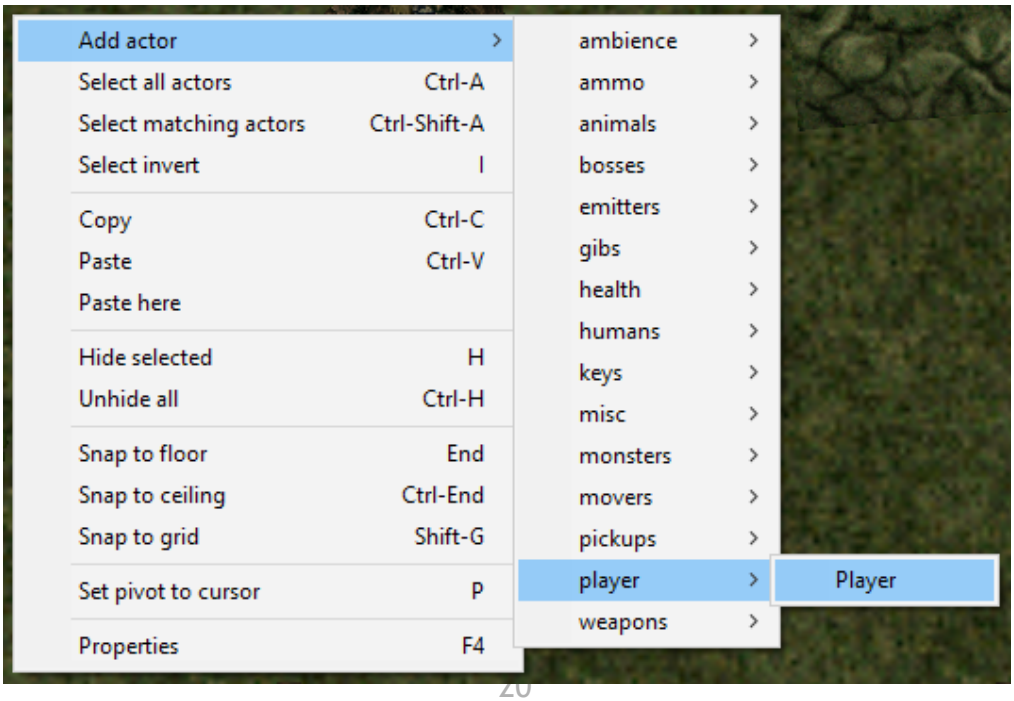
Deselected Sectors are, by default, a red-pink colour. They change colour based on what properties are given to them

(which will be discussed later on)

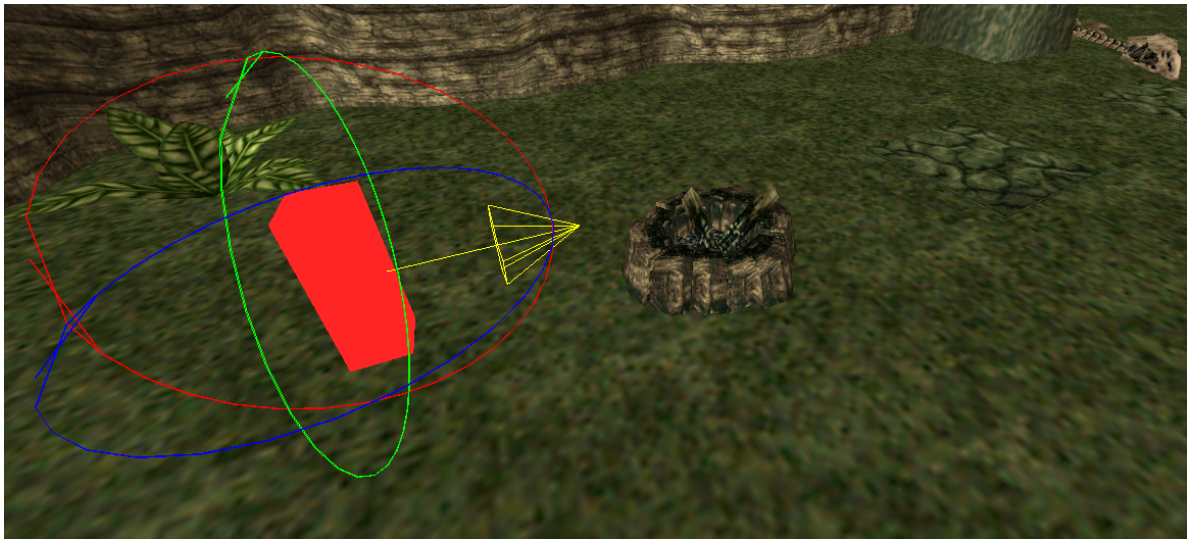


This is how I have filled out this beginning area.

Now that we have a small space that we can actually run around in, it's time to get ourselves in there. We can do this right now. Go into Actor Edit Mode (Your sectors will disappear from the screen when you do but they are safe), right click on an area of floor where you know there is a sector, Go to Add Actors > player > Player.



You should see that a red box has appeared in the play space (If not, try checking underneath the floor, raise it up and press the End key to ground the Actor). This is where Turok will spawn. The arrow on the front indicates what direction the player will be facing when they spawn into the world. You can rotate it by either using the Transform tab or the Gizmo Axis in Rotate Mode.



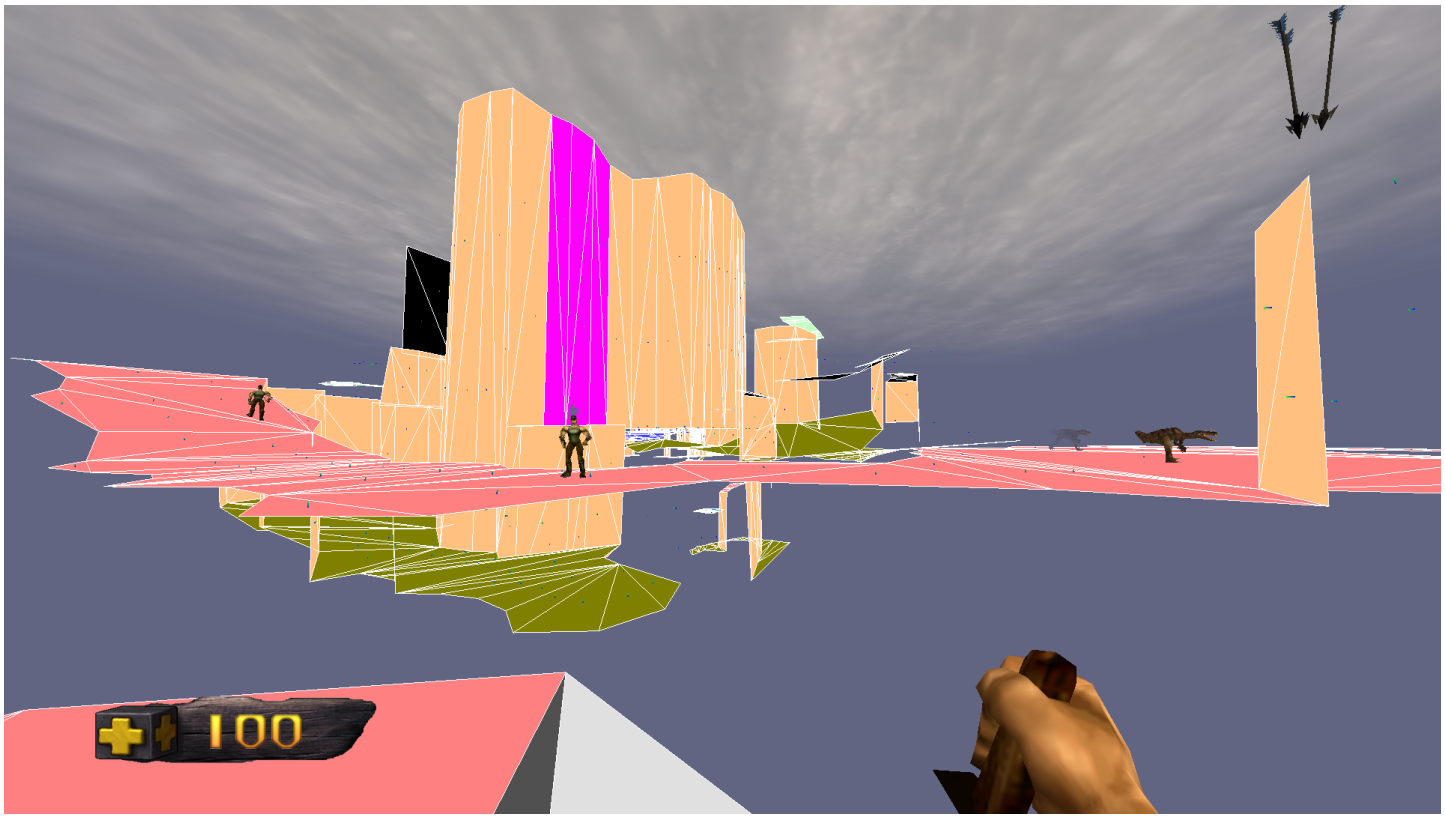
Finally we need to open it's properties (F4 when selected) and in the “Setup” tab, we need to give our actor the Model File “**models/dyn_turok.bin**” and the Anim File “**anims/dyn_turok_anim.bin**”. These settings will allow us to see Turok in in-game cutscenes when we are in the game. He might appear very big in the editor but his size in game doesn't matter. You can scale his model down to 0.2 on the Transform Tab to get him at his correct height.

Now we can finally go ahead and spawn into our world we have created! Save your map if you want and Press the Test level button, the game will launch and you should be taken straight into the map!

You should now be able to walk around too, depending on how much sectoring of your area you have done. You will not be able to walk into any area that is not sectored. It is all needed to be constructed by hand.
You can also check on what the collision looks like whilst you’re playing the game itself. When you are inside your level, bring down the console by pressing the Tilde key on the top left of your keyboard (` or ¬)

If you see a box with a flashing indicator appear on screen, you can type commands in this box to make the game perform certain actions. You can type the word ‘help’ in here to get a full list of commands but we only need one for now and that is ‘**showcollision**’. Type that, press enter and the game will remove all of the meshes from the world. It will then turn on viewing of the Navmesh so you can quickly troubleshoot areas. All sectors will retain their colours from the editor (which you’ll understand more of a bit later on).

Normal areas are light red, the current sector you are standing on is white. Any sectors that are joined up and next to or ‘neighbouring’ the sector you are currently stood on will be a dark grey.



This is a screenshot of level06.map with showcollision enabled. The different colours make it easy to separate what sectors have what properties. You’ll get to learn more of this as we go on.

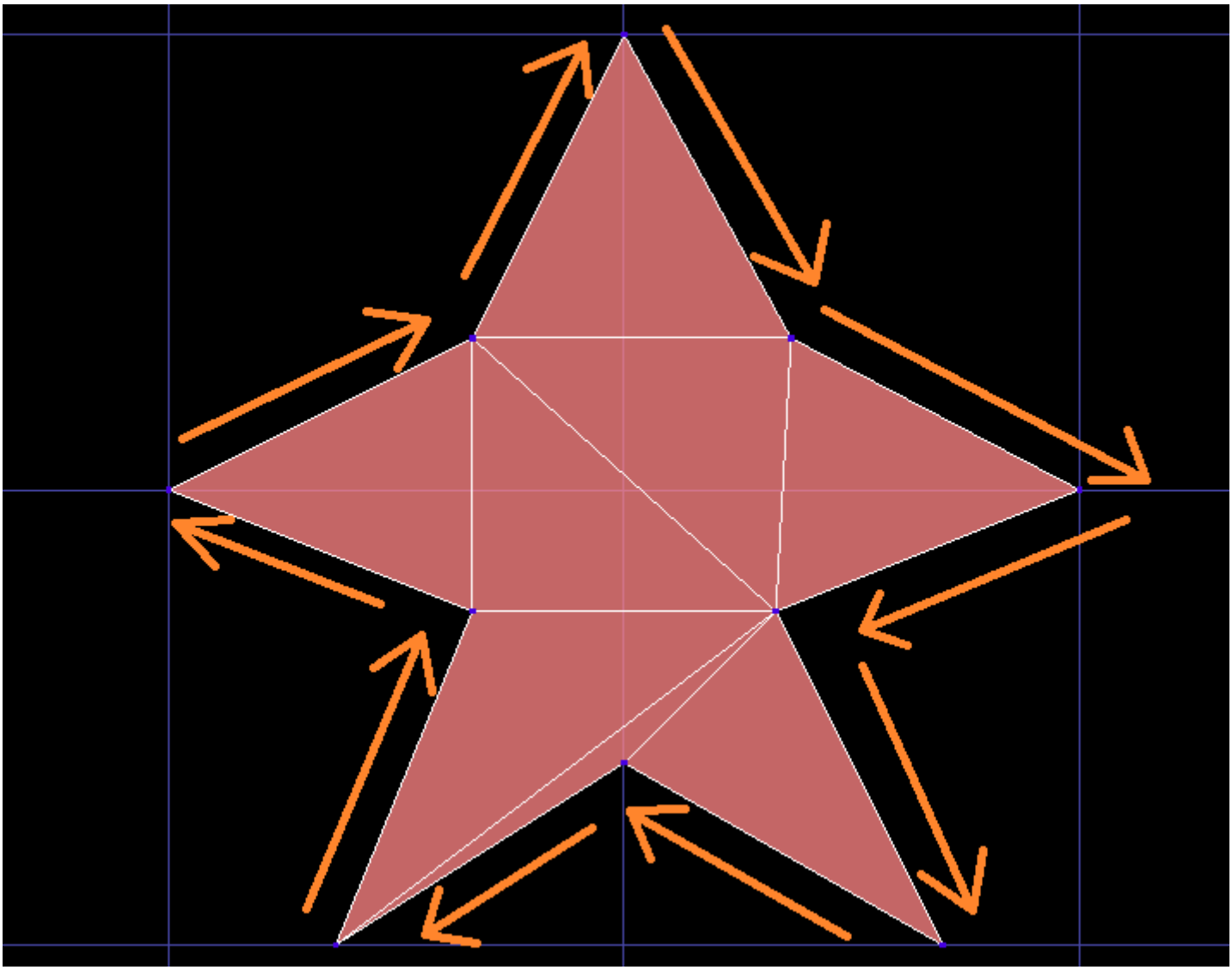
Draw Shape Mode

If you want a quicker way to draw sector shapes then the Draw Shape Mode could be what you're looking for. Be aware that the clockwise rule here still applies.

If you do not draw your shapes in a CLOCKWISE manner then it will only cause misery down the road and things will not work at all. If drawn correctly, your vertices will be filled by the light red colour of sectors, if you see that nothing happens after you have connected your shape then you have drawn it incorrectly and will need to remove those sectors completely before proceeding to redraw them, highlight the sectors AND vertices and hit the Delete key on your keyboard TWICE to 100% remove them from the game world. Pressing Delete once is not enough!

When in "Ortho" Mode, you can press CONTROL+D to enable Draw Shape Mode. This mode will allow you very quickly place down rough shapes. Follow the instructions on screen; pressing the middle mouse button will place a vertices down. Move the mouse and you will see a white line coming from the vertices you placed. Click again to place another vertices down. Keep going until the shape you want is completed and then middle-mouse click on the very first vertices you placed. This will complete the shape.

I drew the star shape you see below in a matter of seconds whereas, without Draw Shape mode, this could have taken several minutes. All I would then have to do is attached the star shape to my bigger Navmesh and everything would work as normal.



Now you have a basic idea of how sectoring works, let's go over some of the more advanced uses of sectoring.

Advanced Sectoring

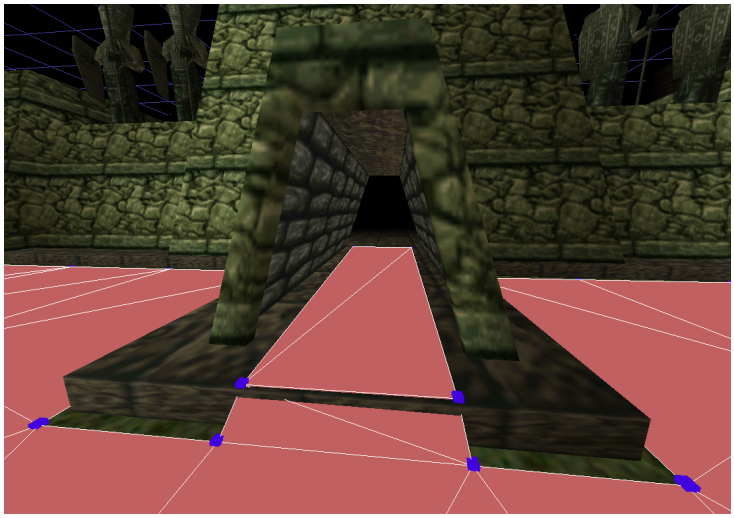
Now things are going to start getting a little more complicated. We'll be taking a look at sector height, restricting enemy access, cliffs, triggering, damage and modifying the sky based on where you are standing.

You may have realized by now that about 90% of your time in the editor will be laying out your sector system for your map. It's important to make sure when you build your environment that you build it and tweak it until you are 100% happy with it as any changes you make after you have put down your sectors will also have to be made to your sectors, especially if you move buildings around.

We're going to take a look as small steps first and then something bigger.

So we have a doorway here we want to be able to access. We have our sectors all around and ready to go but we need to create a small feeling that Turok has just taken a step up here to enter this building, you can create a simple step up by creating two vertices here on the mesh itself. Vertex plot mode will be able to pick up when your mouse is on a Mesh and when you set down vertices, it will have them at the right height for you automatically.

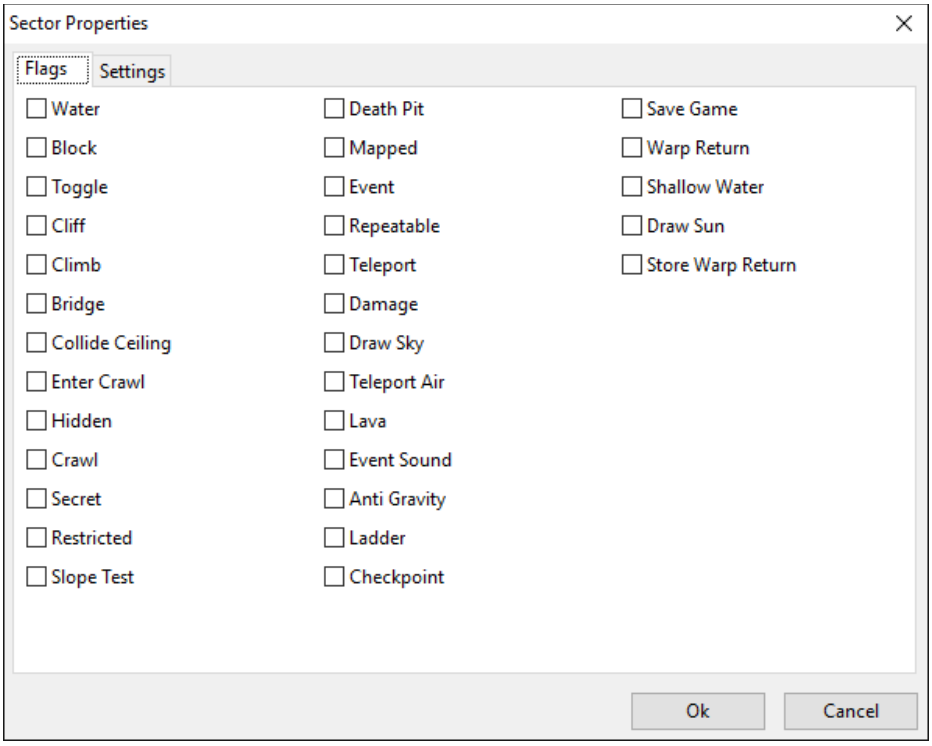
Creating two vertices just outside the door and then joining up with the two we just made is the recommended way of doing this. Take a look at the screen shot below to get an idea of how to set this up.

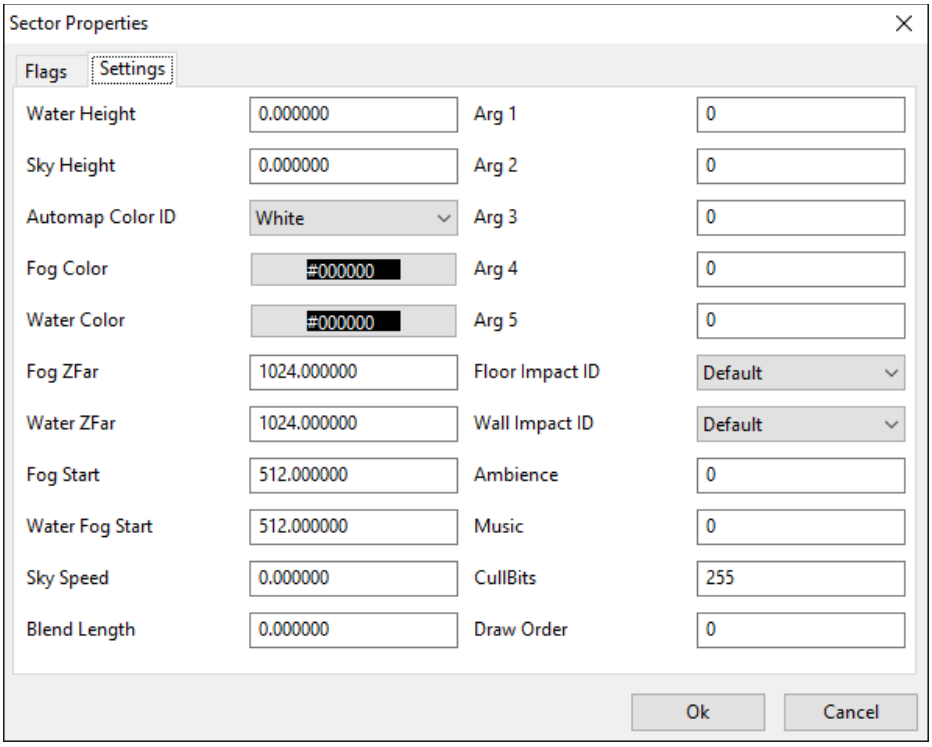


You can then continue the path into the building from here. Remember: Nothing in this world is solid. It’s all an illusion created by your sector system that dictates where the player can go.

Sector Properties

You can assign different properties to a sector. A sector can have multiple properties. It is very useful to familiarize yourself with all the properties so you can make your campaign the best it can be. Right-click on a sector or a number of sectors and click Properties (or press F4).





We can start off by learning about Fog, Cliffs and other subjects.

Fog

Fog is something you can't escape from in Turok. The engine and the enemies have their view distance tied to the fog system. You may have noticed that if you changed the “Fog Colour” of the water sectors when you made your pool of water that when you were jumping into the water, the sky around you quickly changed colour whilst you were over those sectors.

Fog can be manipulated from every single sector in your level. The stock colour that the game tends to use is Hex colour #495A7F, which is a combination of the following:

Hue: 147, Saturation: 65, Luminosity: 94. Red: 73, Green: 90, Blue: 127

We can make this the default colour across all our sectors right now. In Sector edit mode, press CTRL+A to select every single sector in your level (Hint: DON'T press Delete! You will make yourself very sad if you do)

“Draw Sky” & “Draw Sun”

In Turok, sectors need to be told to draw the sky and the sun. It is actually referring to where clouds are positioned.

In the Flags section for any sector, check the “Draw Sky” box and the “Draw Sun” box and you will have clouds and a sun straight away. You can dictate the height of the cloud layer by playing around with the value of the “Sky Height” value in the “Settings” tab for any sector.

Don't like where the sun is in the sky? You can move the sun's position in the sky to wherever you like it by placing the camera view where you would like the sun to come from, going to View > World Properties and pressing the “Generate Sunlight Direction From Camera”

Save your map and test out your level when you have given all the sectors Draw Sky and Draw Sun.

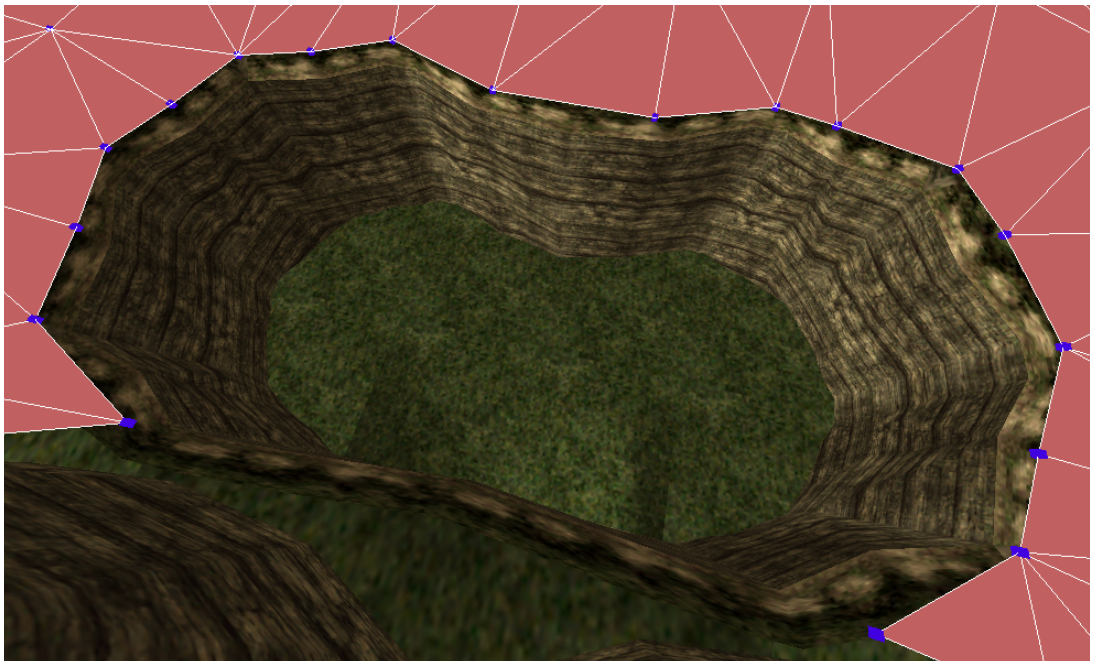
It is best to give all your sectors the Draw Sky and Draw Sun flags very early on. You don't want to have laid down multiple huge complex sector structures only to realize that they all don't have this flag attached to them.

Any sector that is created from another one in “Sector creation mode” (F2) will copy it's properties for itself. Good for when you want to have to do a cliff or a water area as you don't have to assign those flags back on the new sectors. Be aware of what flags your sector has before you branch off other sectors from it.

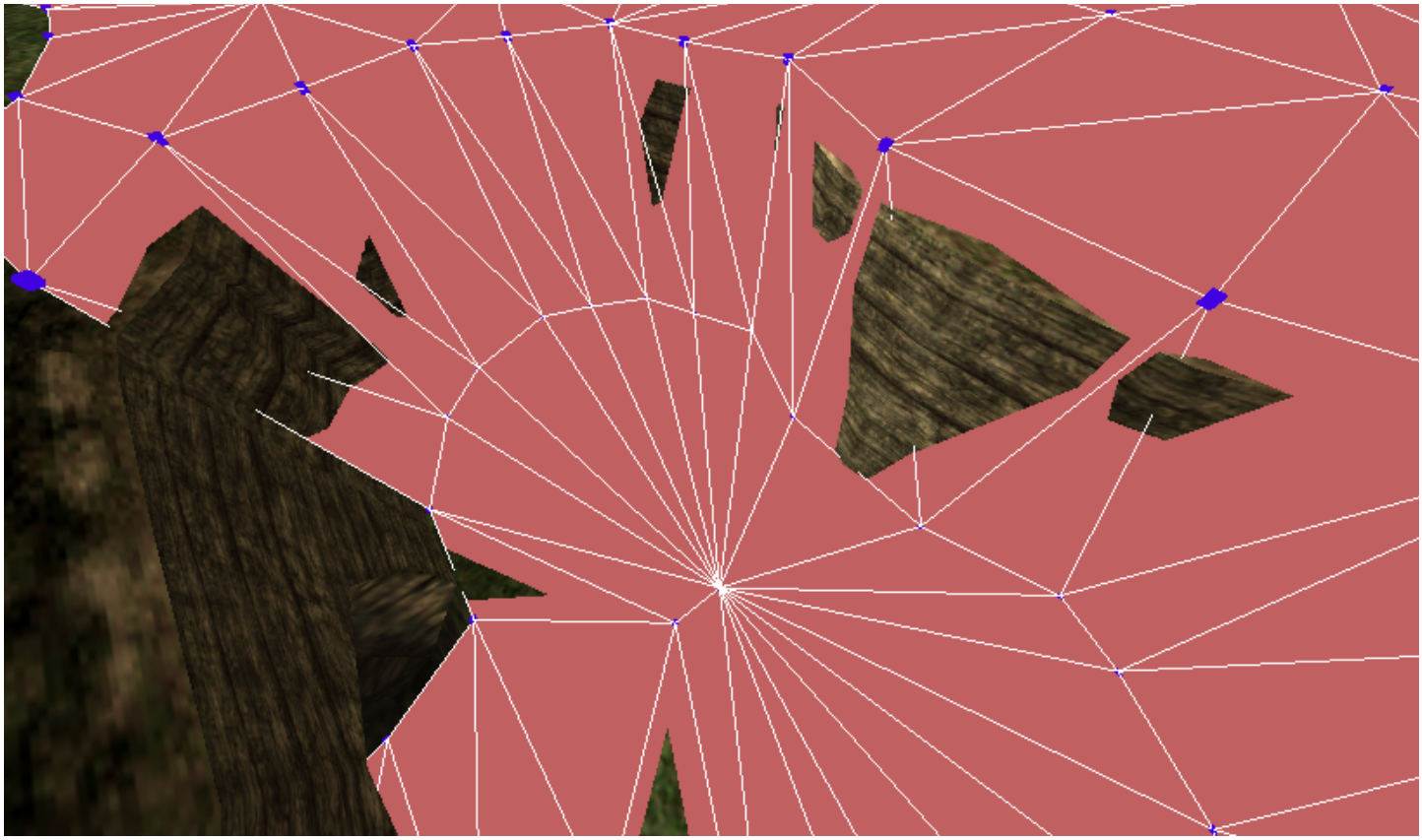
Cliffs

A sector set up with the cliff property will tell the game that that surface can't be walked upon. A player or NPC can fall past it but not get back up to sectors beyond it. Here is the drop that I have built in my level. It's pretty obvious that there is no jumping back up here once the player has fallen down so let's build our cliff sectors.

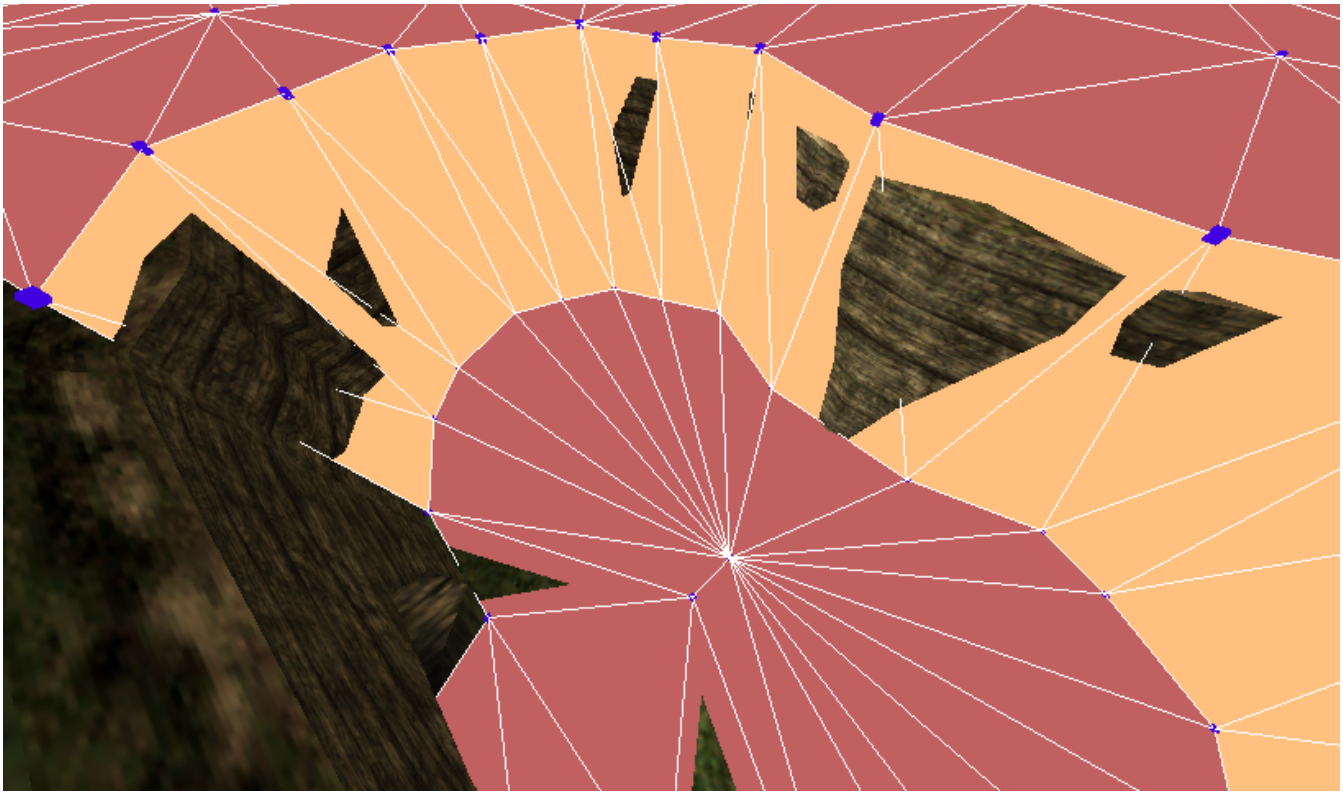
First, position your normal sectors right around the edge of the drop, try to be as precise with placement as you can; players may be upset if they fall off drops too early when they were just taking a look.



Once they are in place, you want to create a set of vertices at the bottom of the pit where Turok will land.

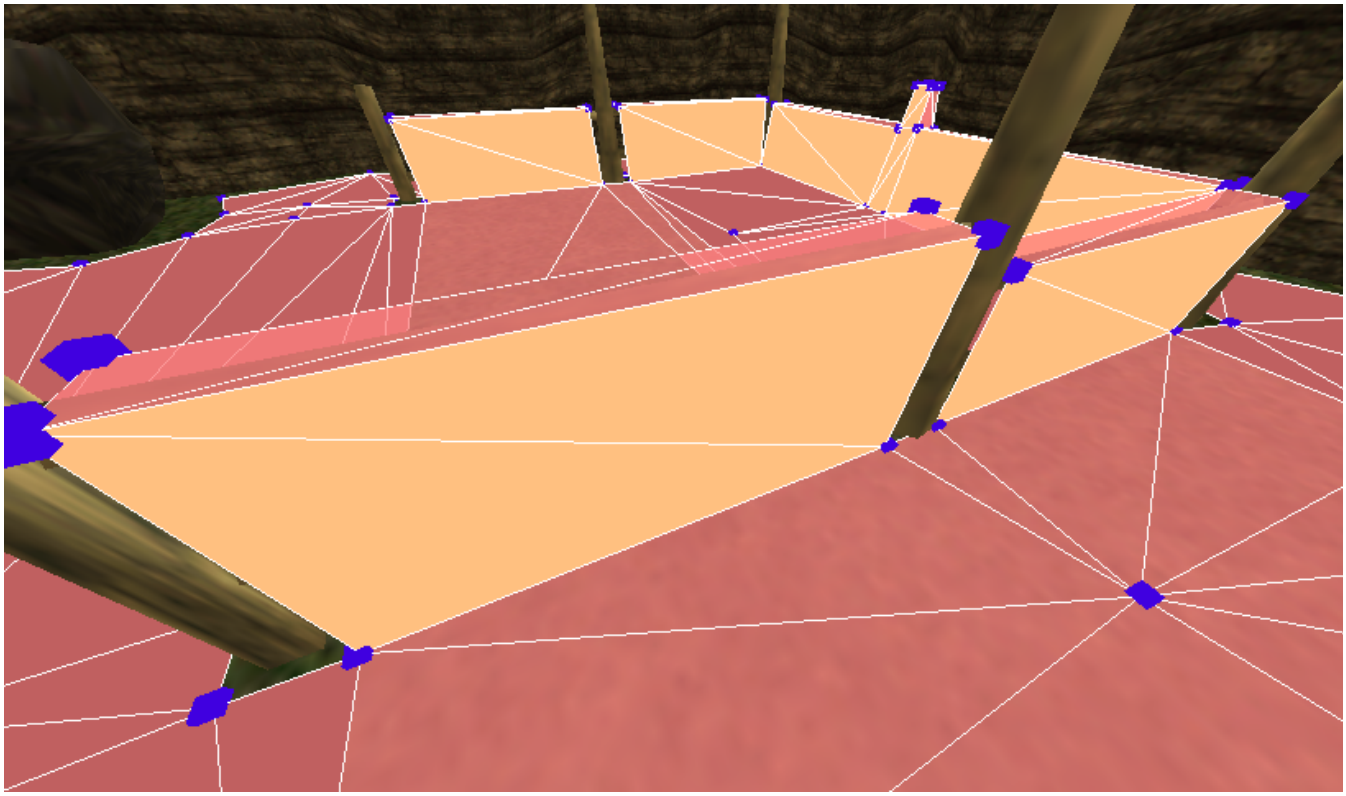


Once this is in place, these sectors downwards need to be set to have the cliff flag set. They will change colour when their properties have been changed.



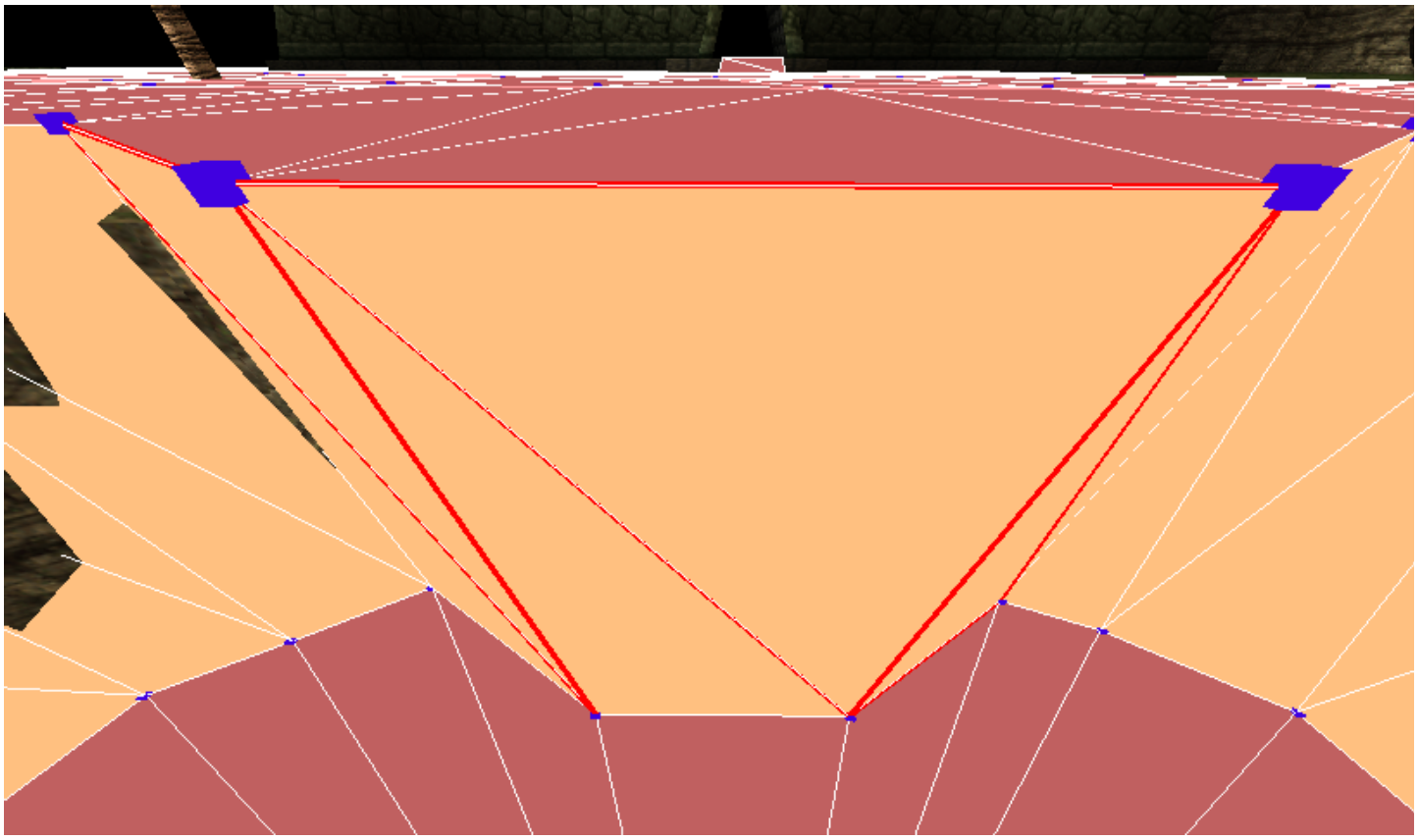
Now, when the player falls down this hole, they will be unable to climb back up!

This doesn't have to be used exclusively for cliffs. It can be used on other things such as fences as shown below.

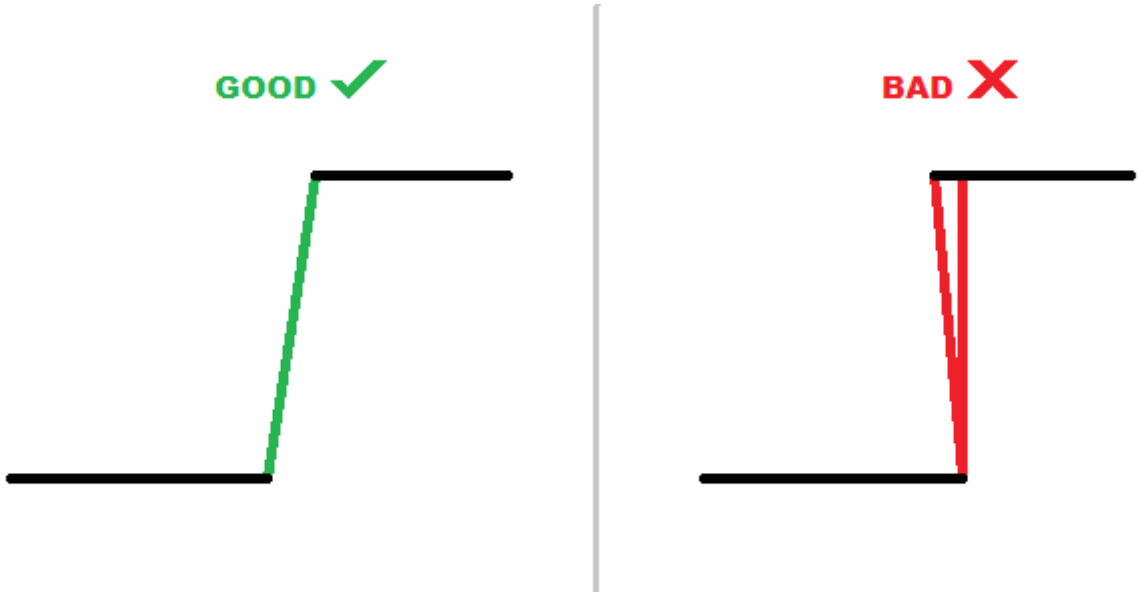


Bad Sectors

When you are positioning your sectors, you might see the outer lines of a sector turn red.



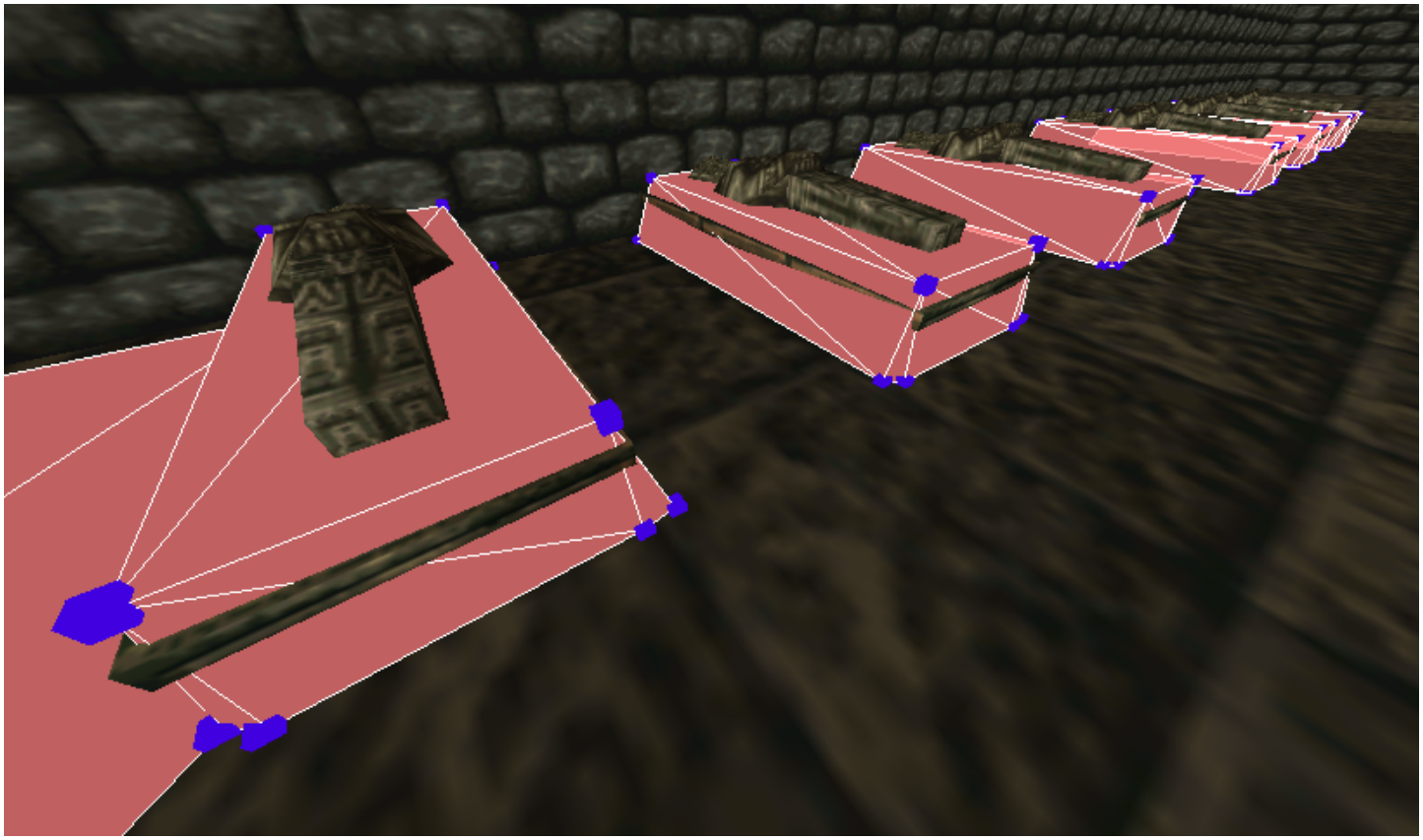
All sectors in the world MUST have their surfaces facing upwards. Because of the way the engine works, if the sectors do not have a part of them facing upwards, even at a complete 180 degrees, the sector will become bad. Your map will still function, save and can be played in but strange and undesirable things may begin to happen if the player or an enemy walks on one of these bad sectors.



Always try to place your sectors like this. As long as there is no red outline on a sector. It's OK.

Copy and Pasting sectors

Say you have a rather complex sector you've just created but you need to create this several more times for whatever reason. You can save yourself time by simple clicking on the sectors you want to copy (trying to Copy and Paste vertices will not work) and hitting CONTROL+C to copy and CONTROL+V to paste. The selected sectors will be placed in exactly the same location as the original ones but you can easily move them to your new desired location.

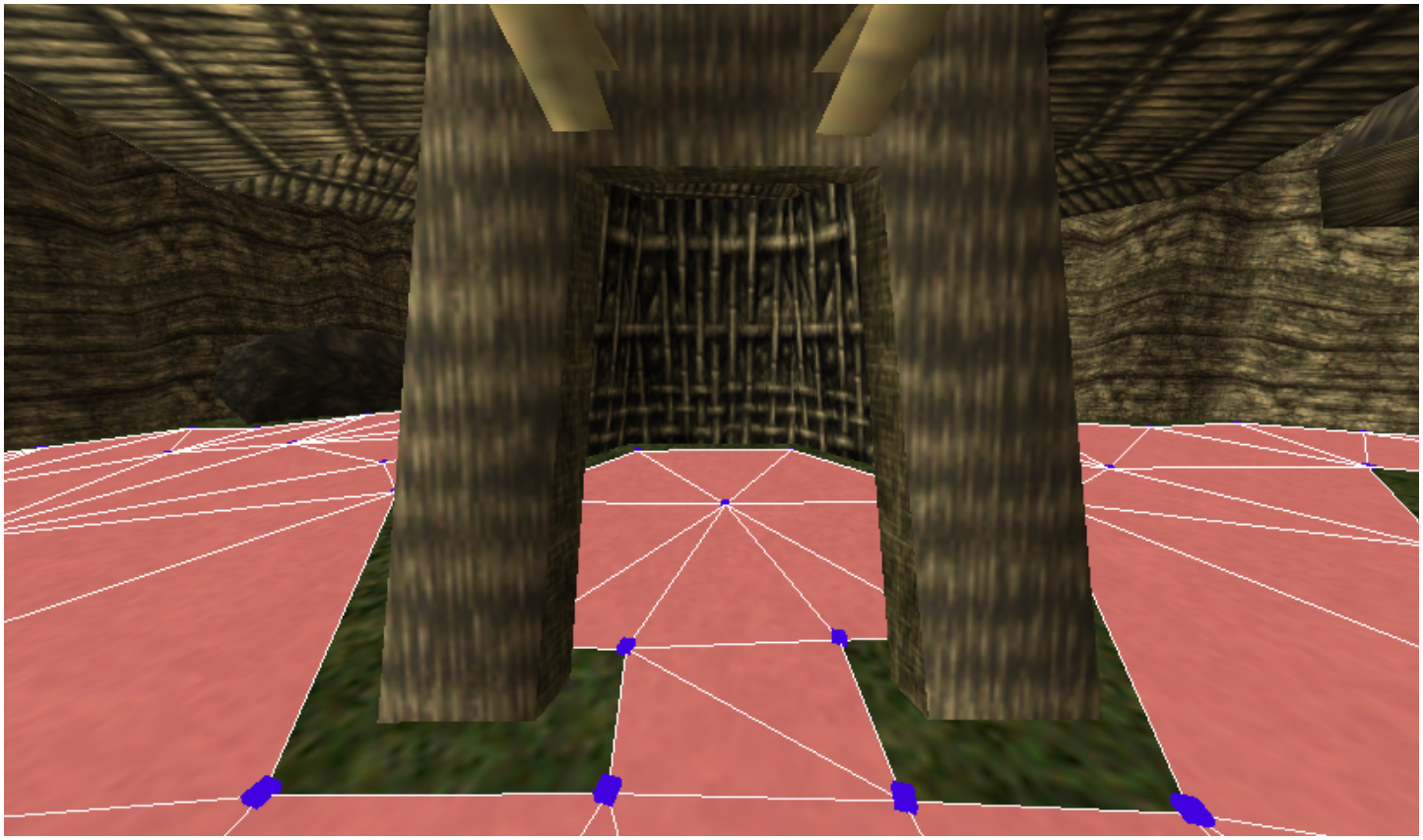


Remember to attach your copy-pasted sectors back to the main sector when you have created them. Otherwise, the player will not be able to access them.

Ceiling Sectors

Ceiling Sectors are exactly what they sound like. They give a ceiling to a sector. Ceilings must be considered for every building, cave and close quarters place in your game as they can prevent Turok from jumping too high and passing through meshes.

Currently, these huts do not have any ceiling information assigned to them. We will need to go ahead and do this now.



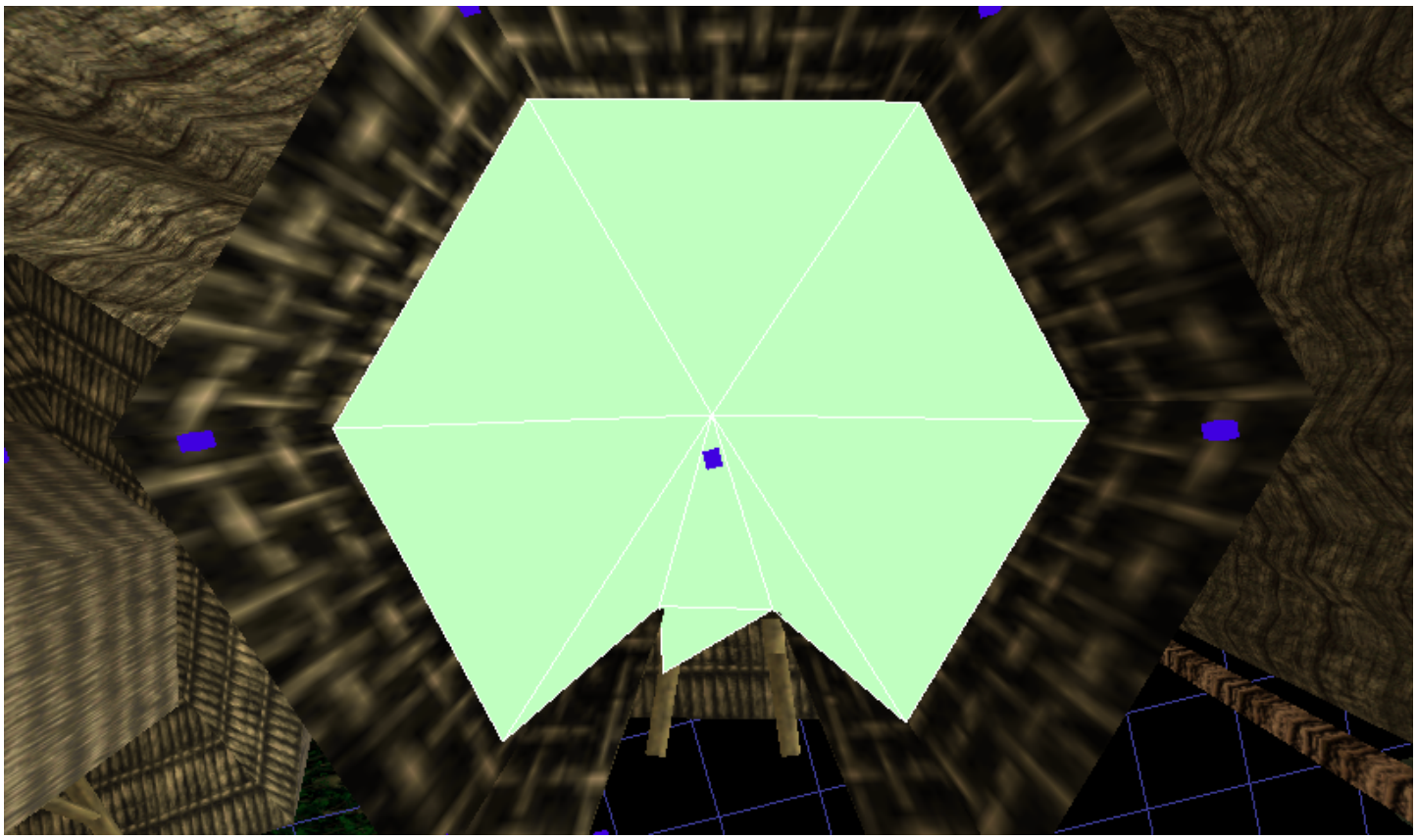
First, press ESCAPE to deselect anything you might have selected. Highlight all of the sectors that you want to give a ceiling to. Be sure to look carefully around your map and identify all areas and press F4 to bring up their properties. In the Flags tab, check the “Collide Ceiling” check-box and press OK.

Now if you move the camera back and bit and look up, you will see some light-green coloured sectors floating in the sky, a long way from where we actually want our ceiling.

We can move these ceiling tiles by selecting their ground vertex, holding down the SHIFT key and moving the Gizmo Axis arrow up and down. Instead of the ground vertex moving, the ceiling sector will be moving instead!

When you first move a ceiling sector, it will snap down to the ground instantly from its high up position. You may need to position the camera around several times as you configure these ceiling sectors. Eventually, you'll get the hang of it.

Position the ceiling sectors to where you want Turok to “hit” when he hits the ceiling and be sure to test out your map once you have done positioning them.



In some situations, you might find that assigning sectors with ceiling collision one by one is significantly faster as, if you move certain pieces before moving onto the next, those vertices will be in the correct place already. Experiment and see which works the best for you.

I've left one of the huts in my starting area with no ceiling sectors on purpose. See if you can make them for yourself.

Checkpoints

By now you should have built your cave system and are ready to learn something new: Checkpoints!

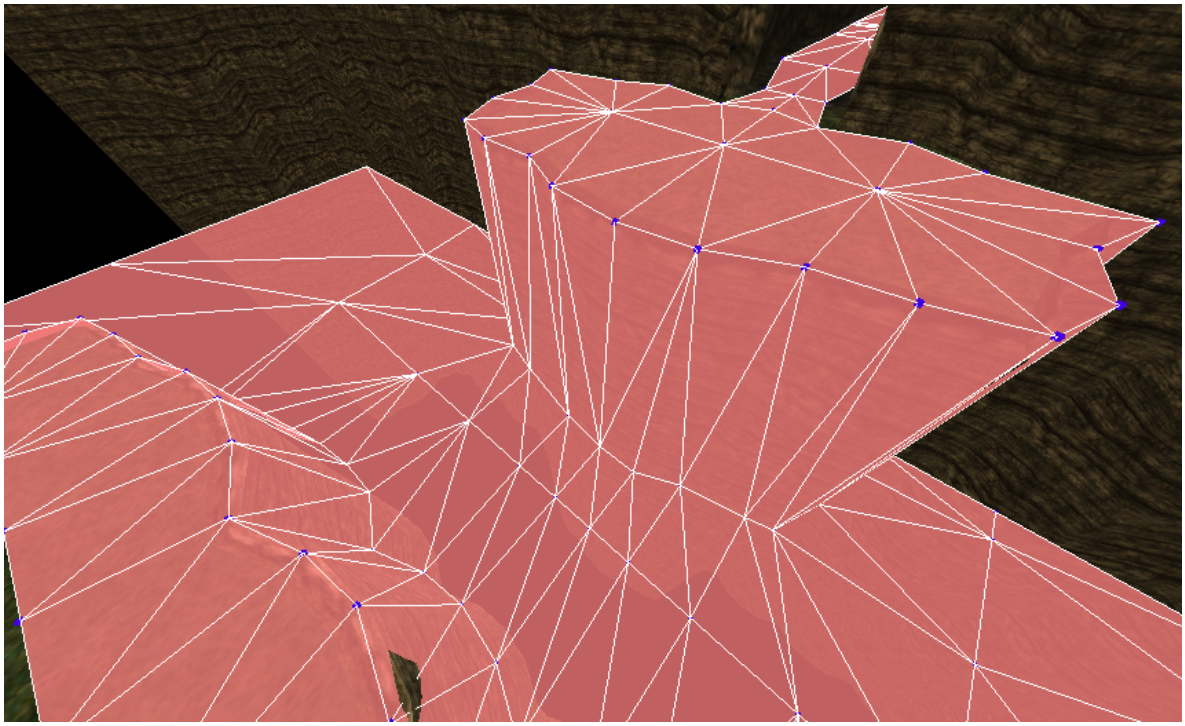
If Turok were to die at any point in our map right now, he would re-spawn in a different level altogether because there has not been any checkpoints set up for him to use. We need to set up a checkpoint here before we let Turok do anything dangerous.

Setting up Checkpoints is quite easy but they won't work straight away. We will need to set something up later in order to get them properly working but for now, simply get a small set of sectors close together, highlight them, press F4 to open their properties and tick the “Checkpoint” flag in the Flags tab. We also need to give these sectors their own ID Number. If we have multiple checkpoints in the level, we need to give them their own unique ID Number. For this specific task, we need to set the “Arg3” number to a unique value. I'll set mine to “1000” and press OK.

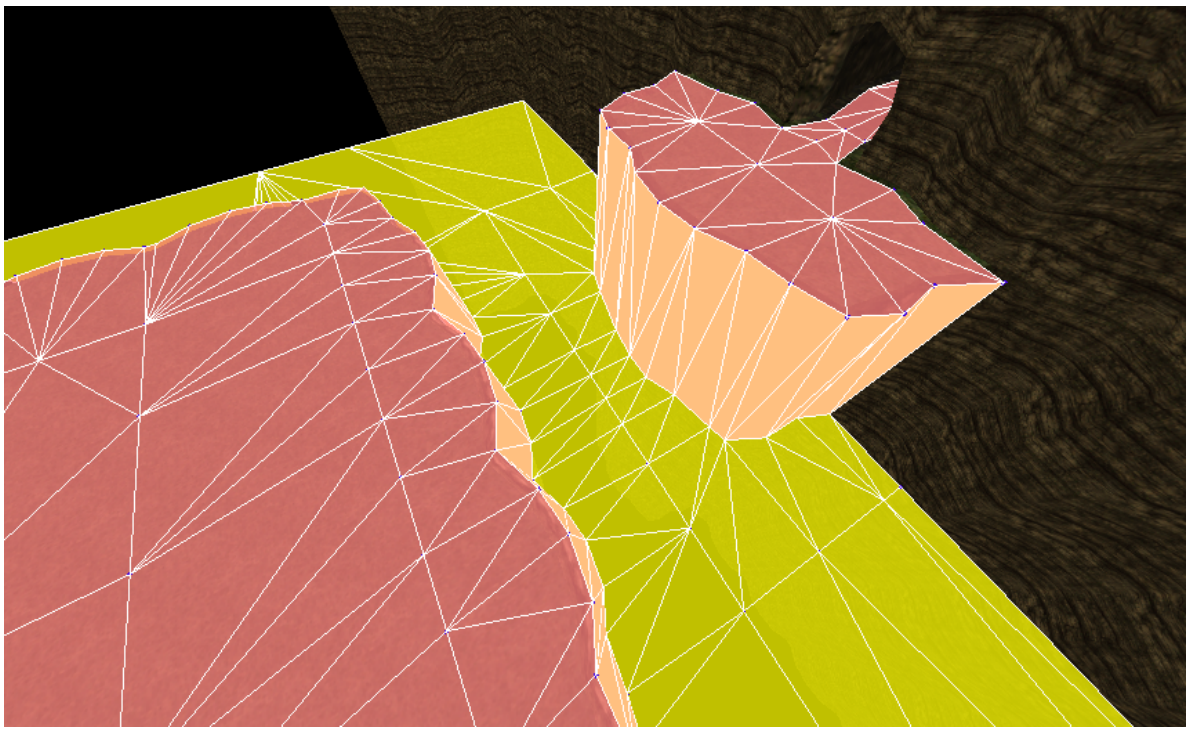
Now that is done, we need to add a WarpPoint that Turok will spawn from when he re-spawns. In Actor Edit Mode, go to “Add Actor > Misc > WarpPoint” and place one down on your checkpoint. Ground it by pressing End and press F4 to go into it's properties. In the General tab. Change the number in the TID field (not Target TID field) to the same number we put in the checkpoint Arg3 field (in this case, 1000) and press OK.

Death Pits

What would Turok be without platform sections and Death Pits? They work how you would expect them to; When Turok touches a sector that has been labelled as a Death Pit, he will die. To create sensible death pits, we have to first fill the area with sectors. Everywhere Turok can possibly go will need filling up.



Once we have that in place, we can go ahead and mark out our cliff faces and then, mark out our death pits on the “floor” of this area.

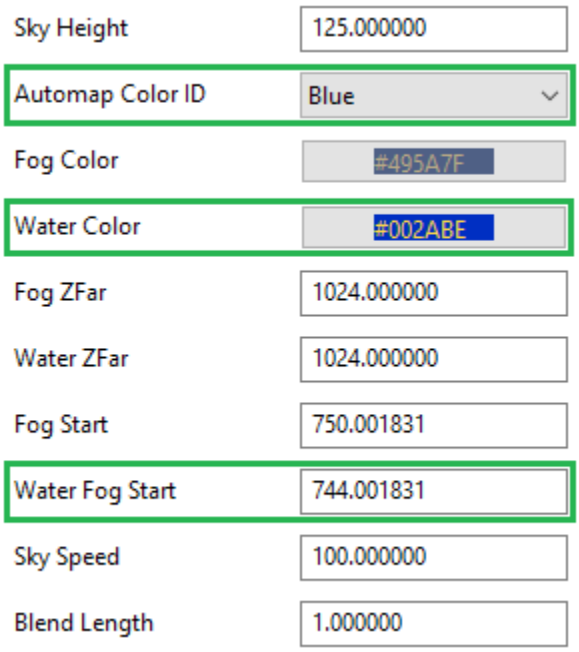


If Turok falls down here, he will die and re-spawn at the last checkpoint he walked across.

Water

Water is relatively simple but it will require a small amount of guess work in order to get it just right. Water requires you to, as always, lay down your sectors appropriately. Once you have done this, you will need to flag the sectors as water sectors (highlight them, press F4 to open their properties and flag the sectors as “Water”) and they will turn blue.

However you will also need to work out the “water height” of this sector. The water will then go from that sector up to the Z axis you input on the “Water Height” field in the Settings tab. Simply position your camera roughly where you would like the surface of the water to be and note the Z axis number which can be found at the bottom of the editor window. Put this number inside the Water Height field and that's it. The water will now work up to that height.



Other good values would be to changes the “Music” setting to 2, this will give you underwater music.

Water Colour #002ABE is a good water colour choice, one the game itself used. It is a combination of the following:

Hue: 151, Saturation: 240, Luminosity: 89. Red: 0, Green: 42, Blue: 190.

Once you have done all this, your water is ready to be splashed about in! Just mark the walls leading down to any water as cliffs so the user can't walk up the walls to climb out of the pool. Also add a floor_water mesh to the surface of your water to the player knows what to expect.

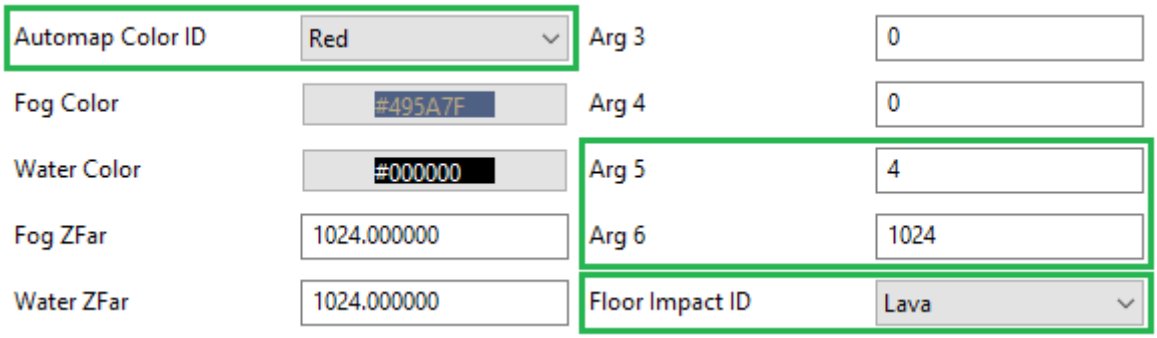
Lava

Another obstacle for Turok to avoid is Lava. Lava is very customizable; you can tell it how much damage you want it to do to Turok and how often along with all the regular things that come with sector customization.

The key settings for Lava are “Arg5” and “Arg6” or Argument 5 and Argument 6.

Arg5 for Lava is how much damage lava will do, based on the time set in Arg6.
Arg6 is the time between each damage “tick”.

A Good value to put inside Arg6 would be 1024. **1024 is exactly 1 second in real-time.** So if you input “4” into Arg5 and “1024” into Arg6. The lava Turok is standing in do 4 HP damage to him every second.



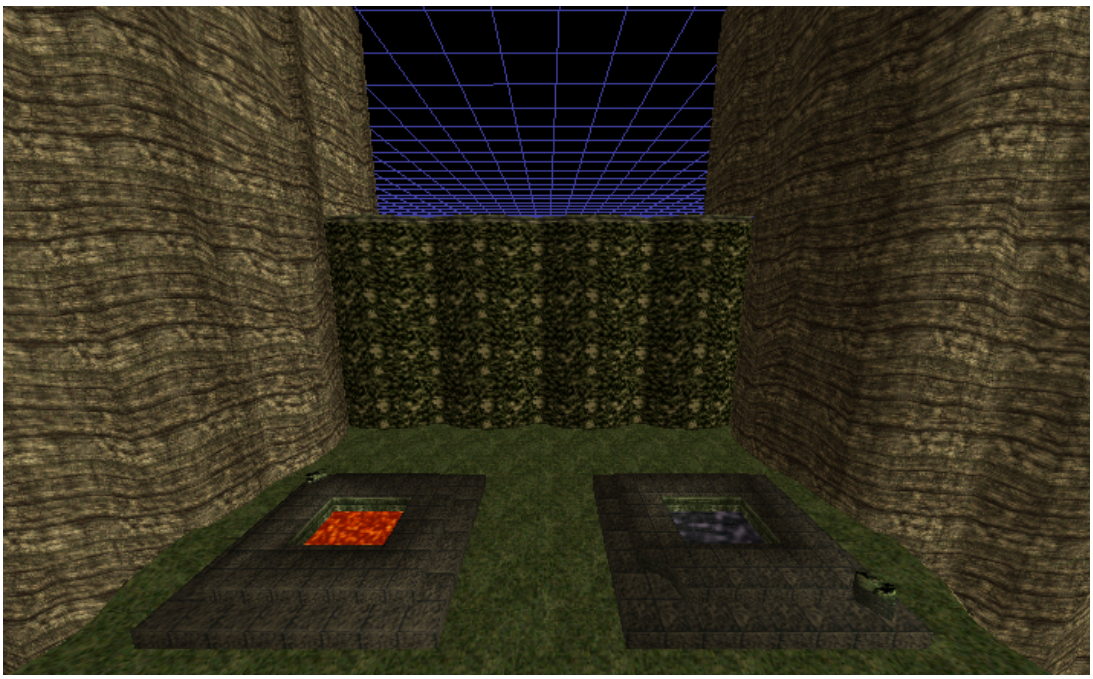
You can do simple multiplications of 2 for working out the time for damage but here is a run down of some simple values you could use in your project:

256 is ¼ of a second.
512 is ½ a second.
768 is ¾ of a second.
1024 is 1 second.
2048 is 2 seconds.
3072 is 3 seconds.
4096 is 4 seconds.
5120 is 5 seconds.

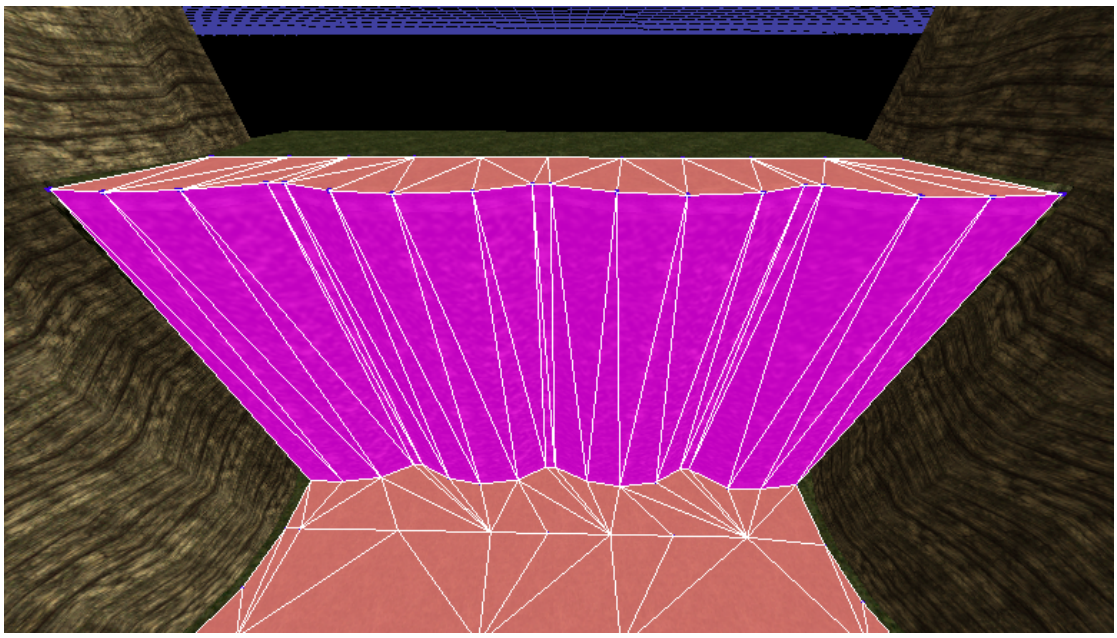
Climbable Surfaces (Cliffs and Ladders)

Turok can be made to climb certain walls. In reality, it's just another sector trick. You assign a sector as climbable and when Turok touches it, he will begin climbing until he reaches the top.

First we need to set up our mesh wall so that it looks a little different from the others to make it stand out to the player that this is a climbable surface (We could also put things going up the wall like Life Force pick-ups)



Once that this has been done to your wall or surface, you're going to have to do the sectors up the wall like your constructing a cliff. Once that has been done, simply select all the necessary surfaces, press F4 to open it's properties and Flag the surfaces as Cliff and Climb. Once you've done this, the sectors will turn bright pink.

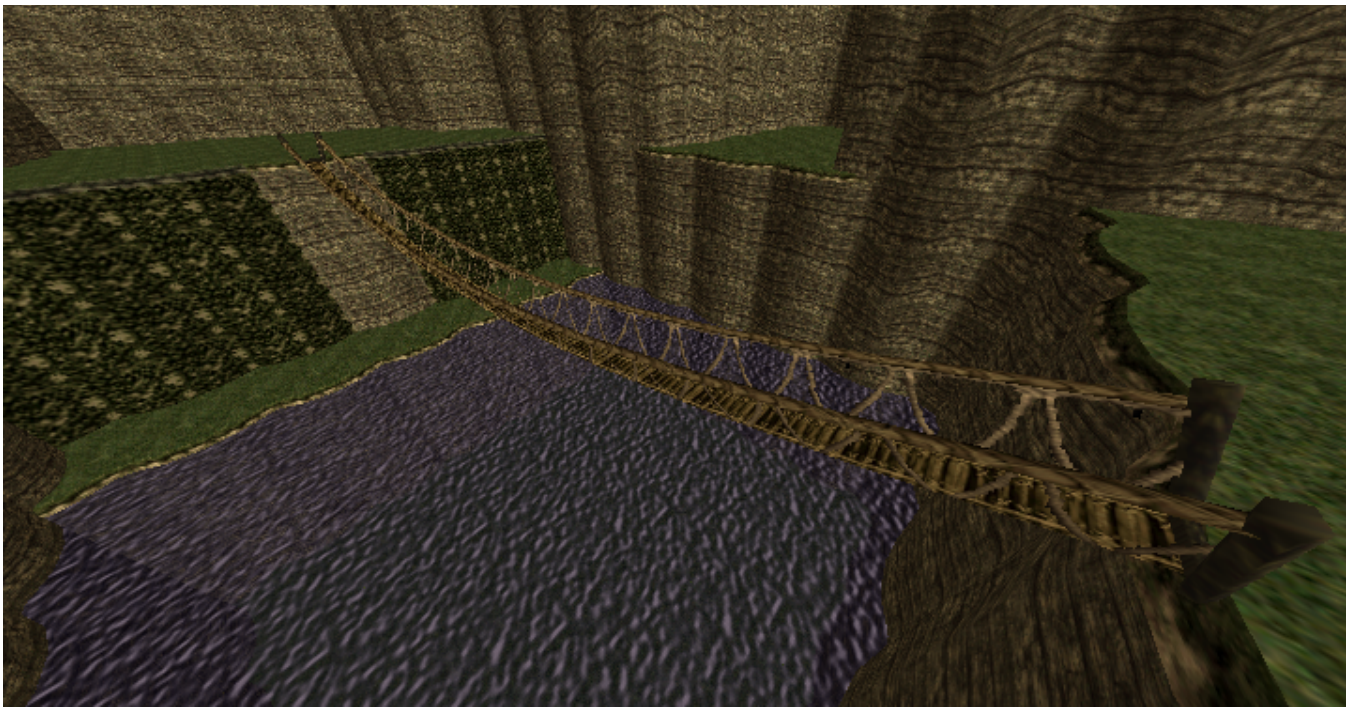


This also works for ladders but ladders make Turok climb slightly faster than climbing up surfaces. Set up a ladder exactly like you would for a climbable surface but you will need to flag the surface with the “Ladder” flag, the “Cliff” flag AND the “Climb” flag.

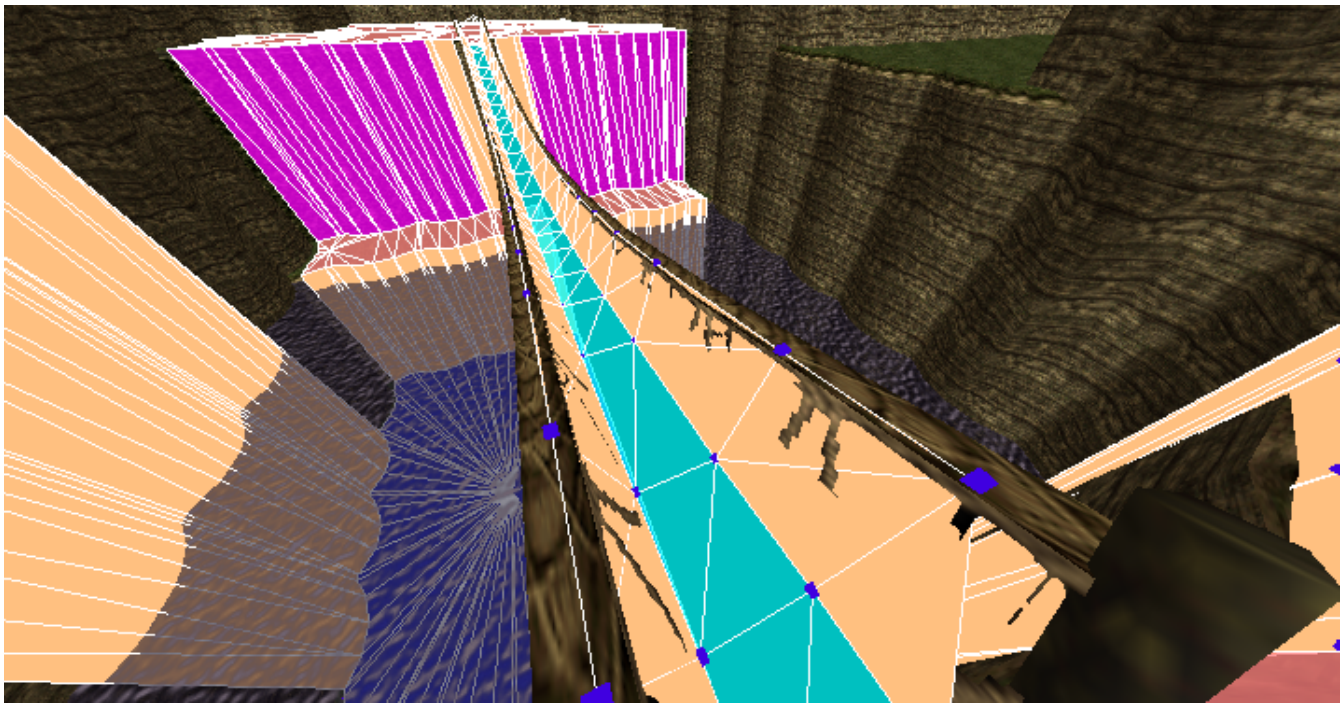
Bridges

Bridges are not that different from anything else we've set up so far but they can become quite complex. Bridges have to work so that things can walk on them as you would expect but also work so that things can walk or swim under the actual bridge sectors themselves.

For this I have constructed a canyon and laid down a bridge mesh spanning the canyon. I have also sectored the entire canyon except for the area immediately around the bridge. I have put some water down below to fill the canyon with something interesting.



We don't want to have ordinary ground sectors here and raise them up, that would mess up a lot of things. Instead we can simply create our own “bridge sectors” on the bridge itself going across, linking appropriately.

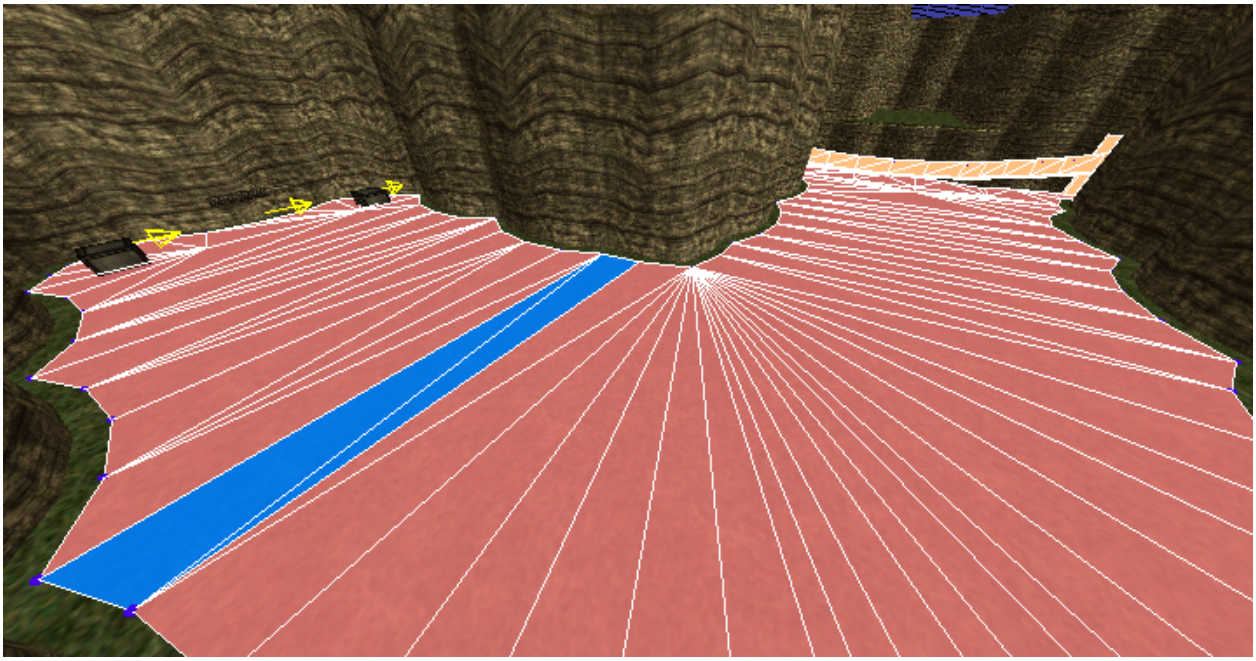


The light blue sectors you see have been given the “Bridge” Flag in their properties. The sets of sectors at either side also have Bridge as a flag but also have Cliff flagged as well to stop the player being able to run up them.

If bridges don’t have any joining sectors leading to a drop, you will see red arrows indicating which surrounding sectors those bridges are linked to when you jump out of the bridge

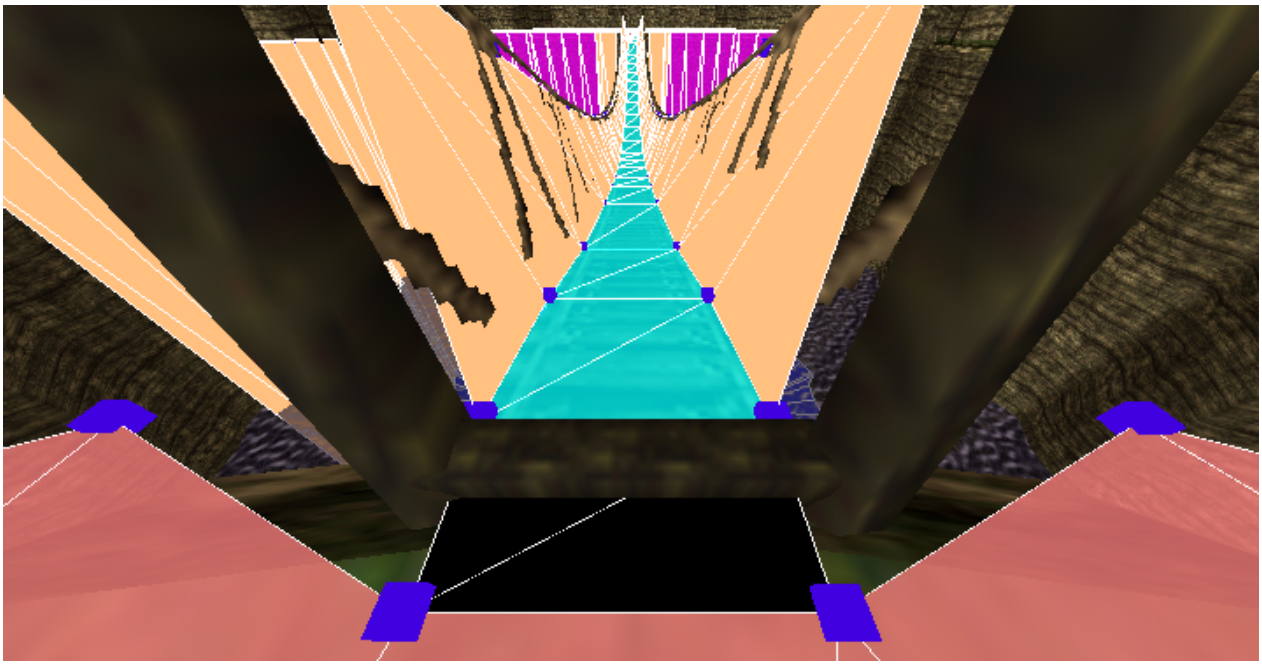
Secrets

Secrets are really easy to set up. Just create the appropriate sectors, select two adjoining sectors and give them the “Secret” flag and you're all set. Hide your secrets well. I've added one to my map in the bridge area. It's hard to reach but possible if you jump correctly.



Restricted

You may have a place you don't want the AI to go, the restricted flag will come in useful for places like this. Simply flag a sector as Restricted and it will turn black. No enemies will walk over them. Simple as that.



Event Sectors and Triggers

Event sectors are the real meat of any level or campaign. Event sectors do what you think they do; they trigger events in the world such as having birds flying overhead, monkeys climbing up trees (that you can see at the beginning of the game), they trigger buttons to open new pathways, enemy appearances from up high and many other things.

There are many kinds of events that we'll look at. We shall start off by doing something simple so you can get a good idea of how the system works and then we'll move onto something more complex.

We're going to have some birds fly past Turok as the map starts.

First we need to go back to our sector edit mode and head back to where Turok will spawn. On the sector directly under Turok, we need to set this sector's “Event” flag. Once that has been done it will turn green.

Now it has turned green, it is ready to receive an event to trigger.

Now we want to get our birds in the world. In Actor edit mode, we need to put down a few Actors > Animals > Animal_Generic (They may appear under the floor from where you clicked) in our scene, face them the correct way. And yes they are monkeys right now. We need to change their properties as well as a few other things.

All objects that do something need to have a unique TID number value. TID simply stands for “Tag ID” number. When Turok stands on an event sector, it will look up and activate the event it has to trigger based on the TID number. Let's set all this up now for our birds.

Highlight all your monkeys and press F4 to open the properties of all of them at once.

On the Set up tab:
Change the Model file to: models/dyn_animal_bird.bin
Change the Anim file to: anims/dyn_animal_bird_anim.bin

On the Spawn Flags 1 tab:
Un-flag the “Solid” check-box, Flag “Avoid Players”, “Melt on Death”, “Flying”, Teleport Avoid Water”, “Teleport Avoid Cliffs”, “Cannot Cause a Fight” and “No Wall Collision”.

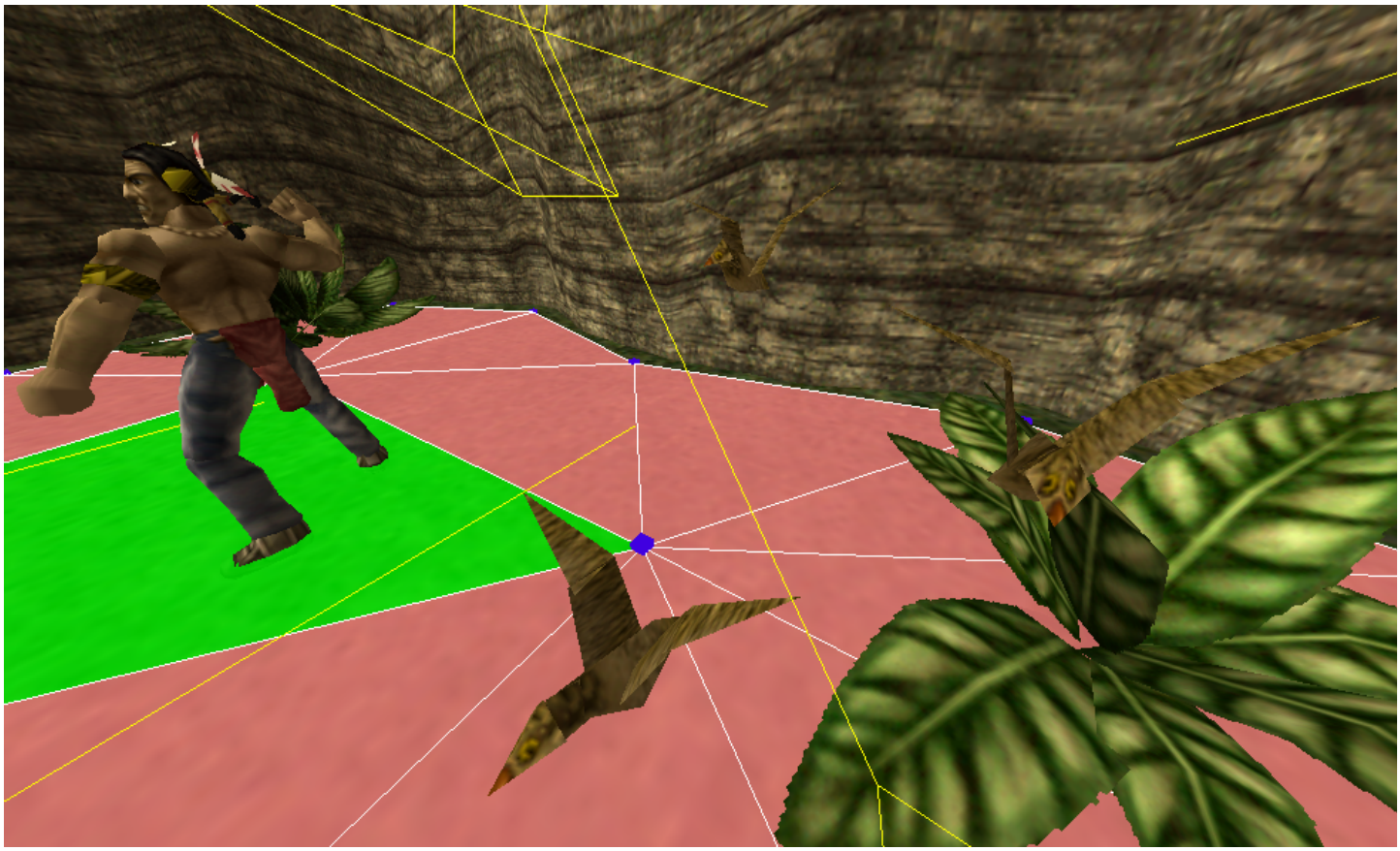
On the Spawn Flags 2 tab:
Flag “Remove on Completion”, “No Blood” and “Hold Trigger Animation”

On the Spawn Flags 3 tab:
Flag “Play Trigger Anim Once”

On the General tab:
Set the Health of our birds to 0,
Enter in a Unique TID number inside the TID field. I will go ahead and enter in the number 451,
Set the Target TID to 0,
Set the Trigger Anim to 1.

That should be it! When you press OK, they will automatically change into birds. Position them wherever you like in the scene behind Turok. Now we need to tell our sector to trigger the event “451” when Turok steps on this sector. Because I have put this directly below Turok's spawn position, it will trigger automatically when he enters the level.

Back into Sector edit mode now, go into the properties of our “Event” set sector and we need to put our unique TID number inside the “Arg4” field.



This is what my scene now looks like. I press “Test level”, the game loads my level and the birds will fly past as soon as the level begins!

But! All this was relatively easy stuff. Let's go on to having some enemies from down from the skies above when we get a ways into our map.

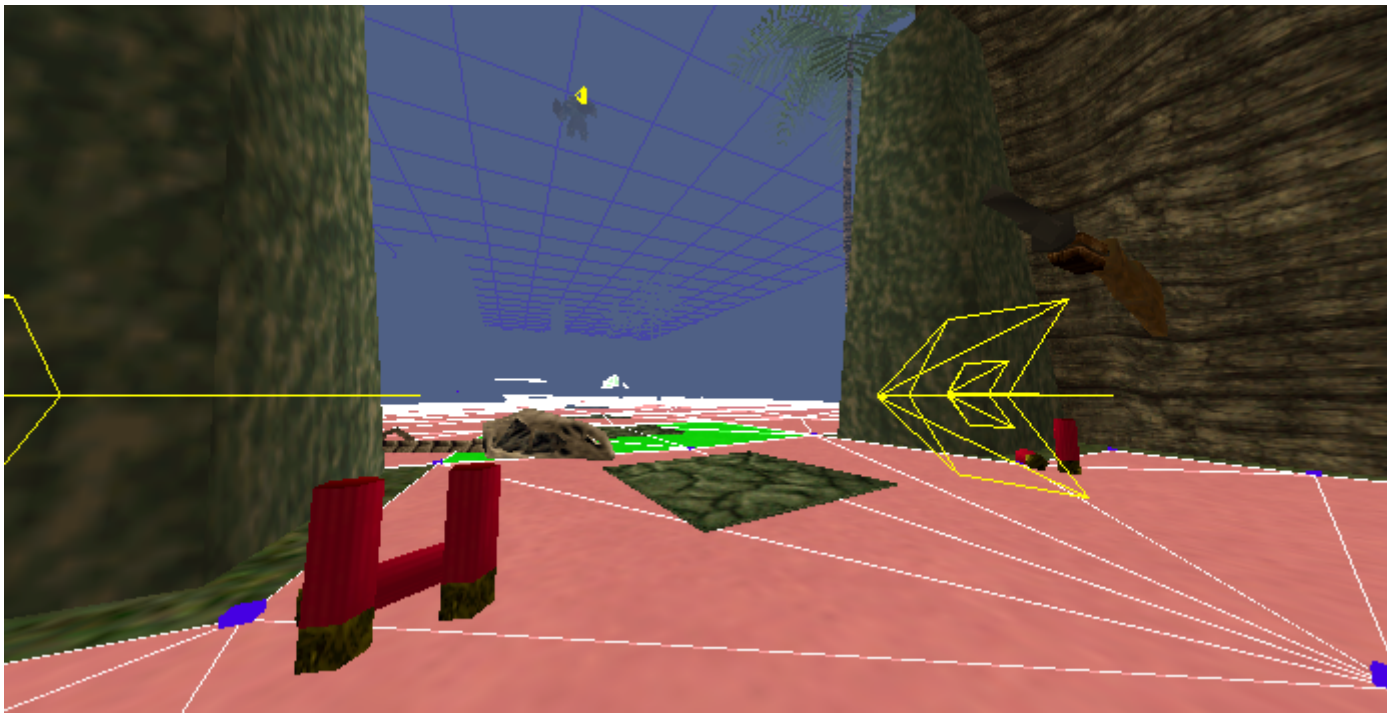
Regular enemies can be placed on the ground and they'll walk around, run around and will go for the player if the player is seen or makes enough noise for them to hear it. But we can have enemies drop down and attack.

Just further ahead, I've placed down a Shotgun and some explosive ammunition (Actor Edit Mode. Right-click, Add Actor > Weapons > **Wpn_Shotgun** and Add Actor > Ammo > **Ammo_ExpShells_Pickup**) and selected some sectors, flagging them with the Event flag and giving the sectors a unique TID number. I'm going to use the number **460**. The last thing we have to do is put a “Purlin” enemy in the world (Add Actor > Monster > Monster_Purlin)

I've raised the monster high in the air (The game will work out where the floor is for the enemy to land on when it drops down) and I've given it the following properties.

Spawn Flags 1:
Snap to Floor, Cast Shadow, Use Strong Attacks, Use Weak Attacks, Melt on Death, Avoid Water, Teleport Avoid Water and Teleport Avoid Cliffs

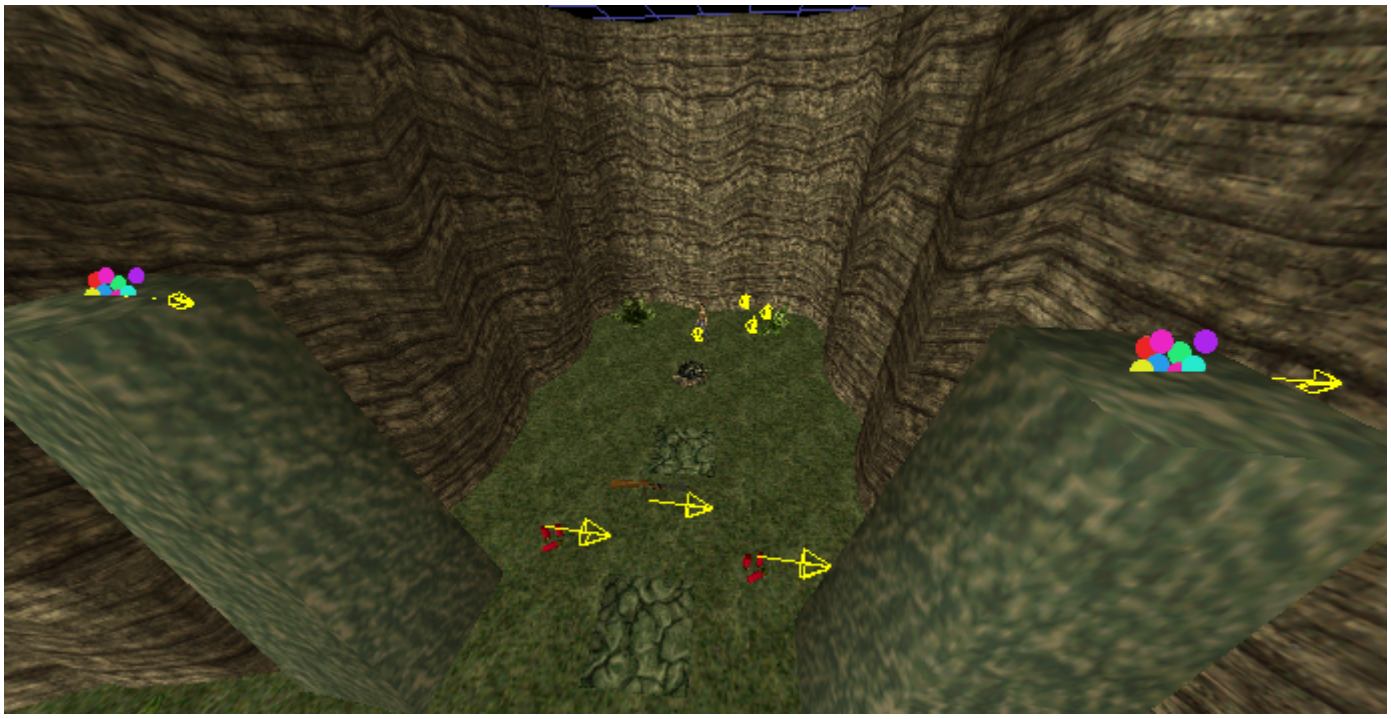
General:
TID: Just like before for the birds, we need to give a unique number in this field in order to tell the game that when the player steps on a sector with the same TID, trigger that TID event. I've used the number **460** here.
Target TID: 0
Trigger Anim: 6 (6 is the drop-down animation for the Purlin, it can be found in the Animation Editor under the number anim_27. To check this out in the Animation Editor by loading the animation file “anims/dyn_monster_purlin01_anim.bin” and the model file “models/dyn_monster_purlin01.bin” and going to the current animation “anim_27”)
And that's it! We just start our map. The birds fly past us, we can walk over and grab the shotgun with explosive ammo and soon after that, we get attacked by a Purlin as it drops from the sky.



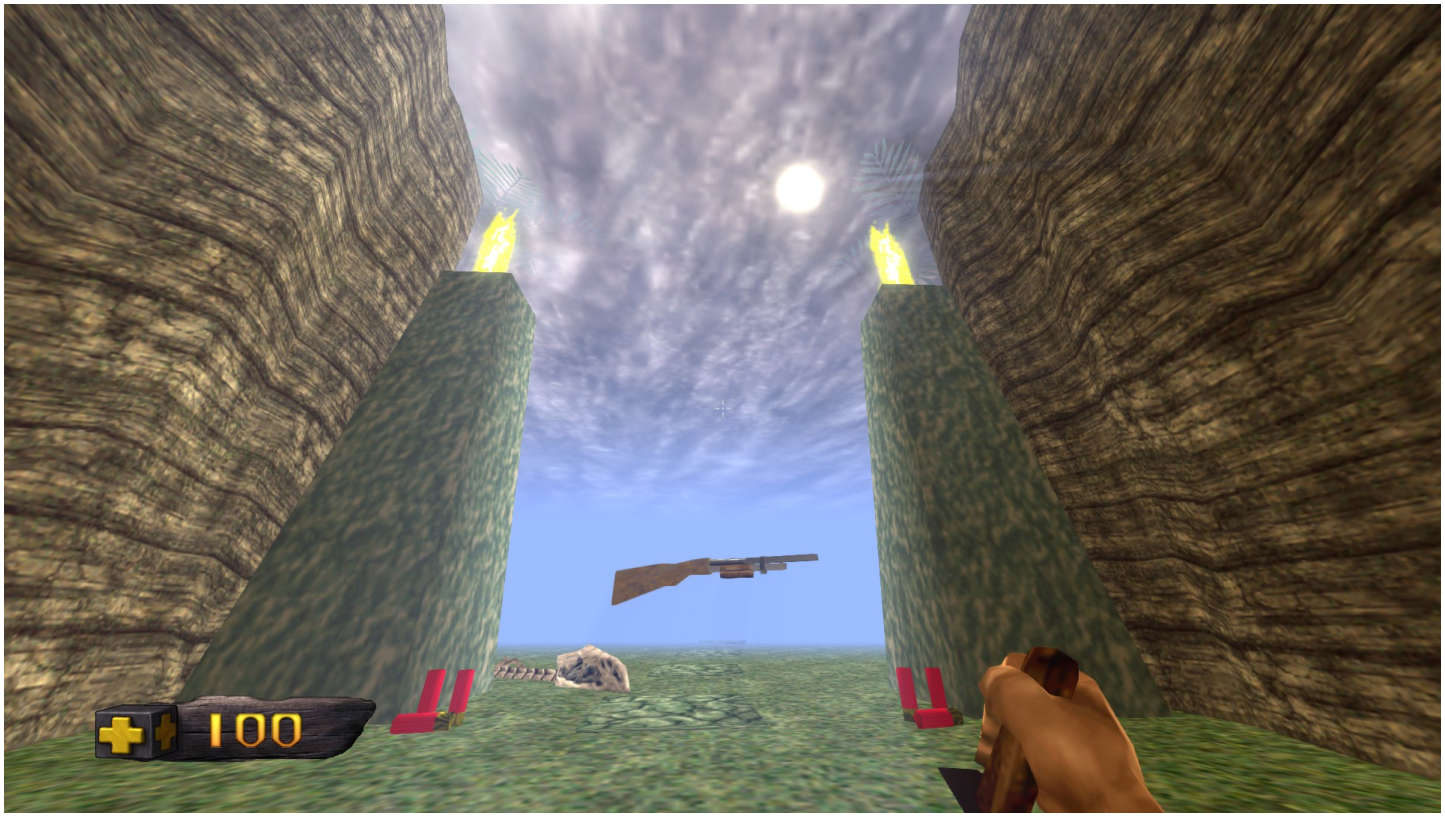
A quick important thing to mention is that some values cannot be too high in the editor. Set an Arg4 value or a TID value in-between the numbers -32768 to 32767. The editor will not accept any lower value than -32768 or any higher value than 32767. When you come to testing your level, things will not work as intended if you have numbers outside that spectrum.

Particle Effects

Turok comes with several particle effects. They come from “emitters” which you can place down in Actor edit mode. Right-click where you would like your emitter to go, go to “Add Actor > Emitter” and choose what you like! You can only see what is being emitted when in the game. In the editor, you will get a simple icon comprised of many multi-coloured balls which you can move around and reposition.



As an example, here are some fire emitters in the game on top of these posts!



Very Advanced Sectoring

The Mapinfo.txt file

OK, now it's time to get working with scripts! Turok's scripts are not the most complex thing in the world but it would REALLY help right now if you picked up a decent NotePad tool or replacement. You can still use NotePad but NotePad++ is highly recommended for this as it displays things much more efficiently. https://www.notepad-plus-plus.org

(Night Dive Studios, DreamWorks Interactive, Valve Corporation or any individual employee from those companies is NOT and CANNOT be held responsible for any software and/or hardware damage that could (but probably won't) happen as a result of installing 3rd party applications on your computer. By reading this, you agree to this.)

Now we need to alter our **Mapinfo.txt** file which can be found in our /defs/ folder in the Turok directory (You did extract the files from your game.kpf file right?). Go ahead and back this up first and then open it up in your text editor. Taking a look at the layout, we can see a familiar pattern in how the script works:

```
Map_Jungle      2
{
   title          "the jungle"
   map_1           "levels/level06.map"
   map_2           "levels/level07.map"
   mapid_1         6
   mapid_2         7
   maxDistance_1   1000
   maxDistance_2   2000
   warptid         11000
   keys           3
   script_1        "scripts/map/generic_script.txt"
   script_2        "scripts/map/generic_script.txt"
}
```

We first have our collective name for a level or a group of levels followed by a number that can be used for teleporting from one area to another. If you were to open level05 in the editor and look at the Jungle portal, it has a TID of 2, which is the number 2 you can see in the name.

Then the “title” which is used for when the player saves their progress.

Then the “maps” which is the most important value. You put the names of your map files in here with the “levels/” prefix.

Then the “map_id”, another important value. This will be used in Target TID fields for things like teleports and level changes.

The “maxDistance” value is where Extended Draw Distance comes into play. Turok ships with the ability to extend the draw distance and remove the vast amount of fog in the level. This can be altered here. When Draw Distance has been extended, the game will look for a value here when it loads the map. If it can't find one, it will default to how the level would draw fog in vanilla Turok. The number is in game units so you will have to do some tinkering, either in the editor or when playing your map in the game, to find the level that would best work for your map and what you are designing.

The WarpTID value is a unique value given to a single WarpPoint when Turok starts a map or a series of maps. The WarpTID is never used for any other teleport in the game.

The Keys value is the how many keys are collected in the entire series of levels. Keys are very complicated to set up and will not be covered in this basic guide.

And finally the script value is where you instruct the game to use any custom scripts that you may make in the future for your level.

You **will** need to distribute a customized Mapinfo.txt file when you distribute your Turok campaign. You can either wipe the entire file of its contents and have a completely fresh file or you can simply add to the bottom of the file the information you want to add. I'll just be adding to the file in this guide.

In any case, it's going to make life a lot easier going forwards if we get our level set up in this script file.

This is what I will add to the bottom of my Mapinfo file:

```
Map_MyCampaign      13
{
   title          "my campaign"
   map_1          "levels/tutoriallevel.map"
   mapid_1        101
   maxDistance_1   1000
   warptid         50000
   script_1       "scripts/map/generic_script.txt"
}
```

The brackets are important for giving the game clear information about what applies to what series of levels. Be sure not to forget to include them and put your text inside them.

With this new information, we will have to alter some things we have already set up in our map. Don't worry as it won't take long to remedy this. We also have to change the way we test our map.

Because the editor saves to a temporary file when pressing the Test level button, we will need to manually load up our level when the game. When in game, press the Tilde key (`) to bring up the console and type in:

maps levels/your_level_name.map

When you have done this, you may notice that, if you extended the draw distance, the draw distance has been changed and you can now see further away.

Warping and Teleports

You might not want to kill outright if they fall down a bottomless pit or maybe you want to surprise them by teleporting them somewhere. Teleport flags are how it's done. Once again, you will need to rely on the TID but also the Target TID will be coming into play. Let's start with warping the player to a particular place.

I have set up an environment for this in the tutoriallevel.map. Here I have made a gap that is too big for Turok to clear by jumping. I don't want to kill the player if they decide to take a running jump and try and clear the gap so the sectors in the pit I have, instead of making them Death Pits, assigned them as teleport sector with the “Teleport” flag in their properties. They will turn bright blue when you do this. All they need to be told now is where to teleport the player too when they touch this sector.

In the sector's “Settings”, the Arg 1 and Arg 2 fields are the important ones. We will need to fill these out and enter in the same information in the TID and Target TID field's of a WarpPoint.

Arg 1 goes to TID, Arg 2 goes to Target TID. Arg 1 is a unique number to tell the teleports where to warp Turok to. The Arg 2 and the Target TID is the map_id number we assigned out map in our mapinfo.txt script.

We need to give them unique numbers so, for my teleports, I have given them the unique Arg 1 value of 606 and the Arg 2 value of 101 respectively. Now all we need is to put a WarpPoint down in Actor edit mode (right-click. Add Actor > Misc > WarpPoint. Snap it to the floor with the End key) and assign the correct values to it in it's properties.

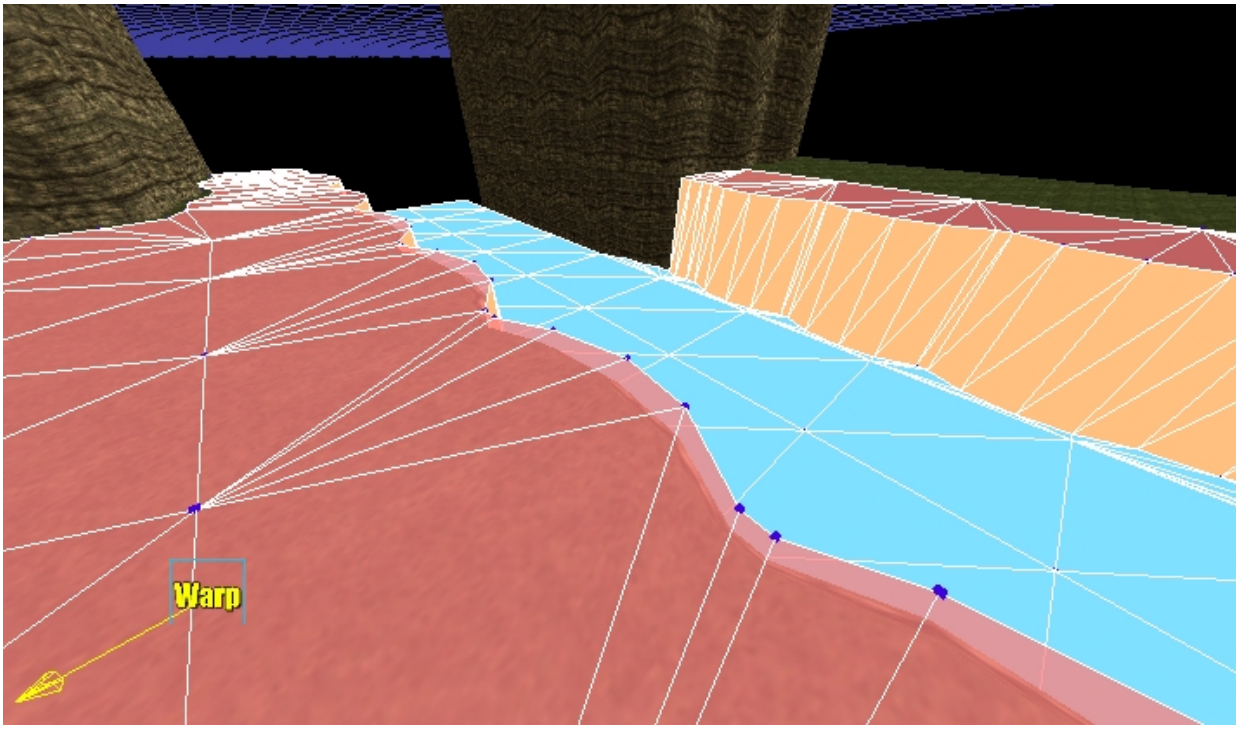
In the General tab of the WarpPoint, make the TID value 606 and the Target TID 101.

Quick Recap:

Sector Properties: Arg 1 > 606 Arg 2 > 101 (Same number as we assigned the mapid in mapinfo.txt)
WarpPoint Properties: TID > 606 Target TID > 101

Now, the map is saved, we load it up in the game (Tilde key to bring up the console, type 'map levels/your_level_name.map' into the console and press enter to load our map) and, if everything is set up correctly, when we fall down our new pit, we are instead warped back into the ledge we jumped from!

You can rotate the WarpPoint to change the direction Turok faces when he spawns. The arrow you see when the WarpPoint is selected is misleading as it does not point in the direction Turok will face but the complete opposite instead. Have your WarpPoint face exactly 180 degrees in the other direction.

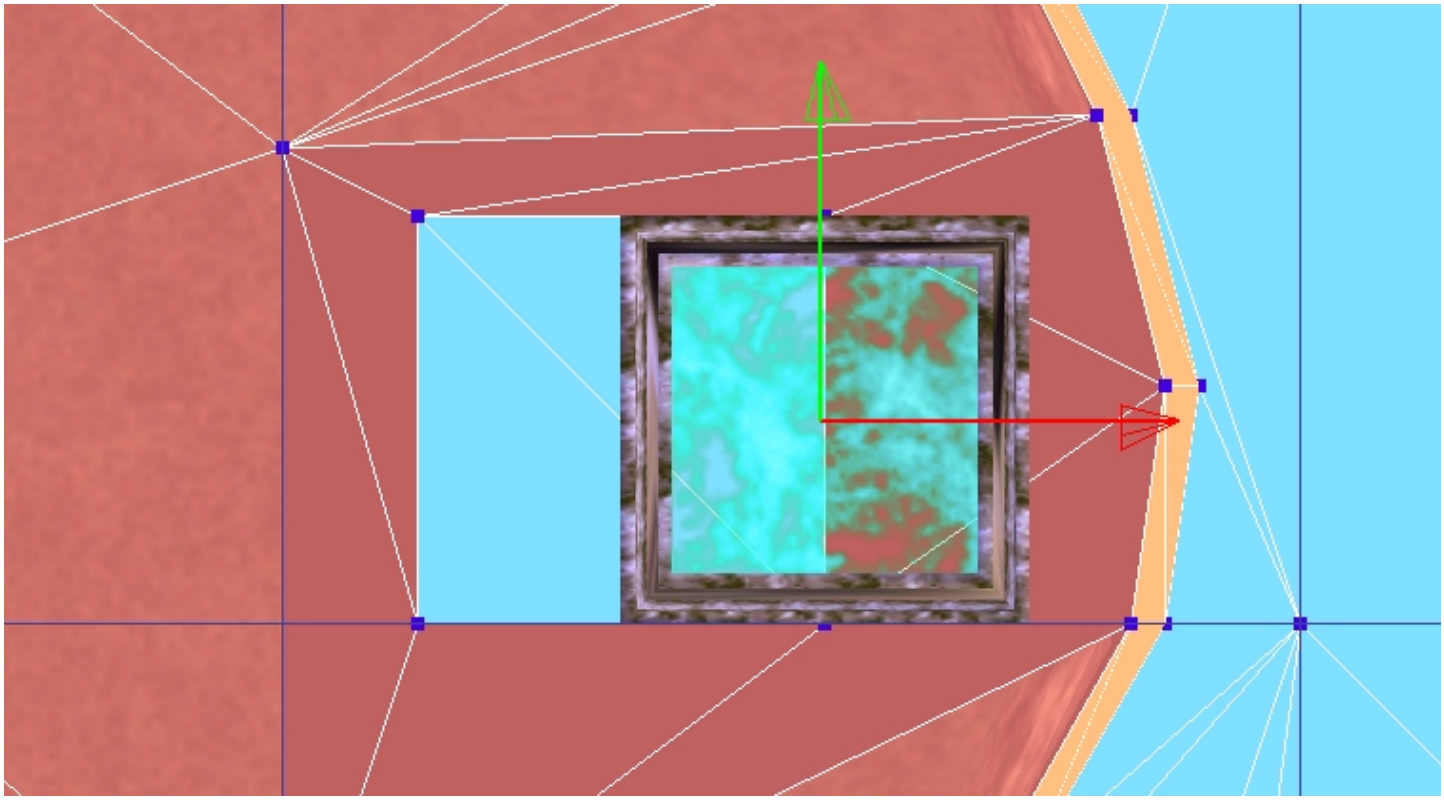


This is how all level transitioning works. As you may have guessed by now, you can warp players to other maps by having the Arg 2 number point to another level (e.g. 102) so when the player touches it, they get sent to a different map.

All this is well and good but right now though, this does not solve how we are going to get over this gap in the first place as we have no teleport. We need to construct something that looks like a teleport so that our player will know to take it. Not only do we need to do that but we also need to make the sectors underneath it teleports so that they will take the player to where we want them to go, in this case to the other side of the ravine that has been constructed.

You can use the basic teleport mesh “misc_portal_exit_common.bin” (there is no “entrance” mesh to go with it) or you could use the mesh “detail_teleport_frame01.bin” and something else if you want to create something different.

You can very easily set up a teleport like the one I have set up in the picture below. Snap to Grid will make construction of this very painless.



Now we can simply copy what we did before, using 101 as the Arg 2/Target TID number but now we have to use a new Arg 1/TID number to separate it from the previous portal we made. I have made this teleports number **610** and I have placed a WarpPortal on the other side of the pit match up with the sector's Arg 1 value and we're done!

Moving Floors

Setting up moving objects is relatively straight forwards. Unfortunately, it can require a bit of guesswork in order to get things working exactly how you want.

A little back-story:

Turok was originally designed to fit on a small capacity cartridge. The original developers ended up filling 7.99MB of an 8MB cartridge. They had to compress a lot of things and cut corners wherever they could to get the game to fit on a Nintendo 64 cart. Because of this, a lot of the following instructions are going to seem nonsensical. The developers made certain AI parameters have different effects on Movers. Just go with it and everything should be fine.

Here we go:

Moving floors doesn't require much in the ways of setting up. You can have a simple Mover actor placed inside the regions of a sector that are marked with the "Slope Test" flag, configure the Mover's properties and you're good to go. Let's take a look as this first before we move on.

I've constructed a cliff that the player will need to traverse to advance. It's not climbable this time. Turok will have to defeat four enemies in order to raise the pillars up one by one in order to progress. Let's put four enemies down here so they can be our "triggers" for the moving floors. I've given them the "Sniper" flag in their properties so they don't run around and attack the player when they get close (only certain types of enemies do well as Snipers)

In the tutoriallevel.map, I have put down four different enemies. Each has a unique TID number that will trigger our floors to move, in this case 1101, 1102, 1103 and 1104.

We now need to put our moving pillars in. In Actor edit mode, right-click and go Add Actor > Movers > Mover_FloorLowerOnce (The name is a bit of a trick as it doesn't actually mean just lower, it can be raised which is what we're going to do) and give it a better model than the one it has, let's go with "models/canyon_pillar07.bin"

Feel free to duplicate it however many times you need to. Open up the properties of the first pillar in the sequence and have the following set to it:

Setup Tab	
Model File	models/canyon_pillar07.bin
Anim File	Blank. It doesn't need an Anim File attached to it to work.
Actor Type	310

Spawn Flags 1:
Check the Solid Flag, nothing else needs to be selected unless you want the screen to shake as the pillar moves. If this is the case, check the "Screen Shake" flag in order to enable this effect. The effect of the pillar shake is relative to how close Turok is to it when it is moving.

AI Properties:	
Attack Chance	3
General:	
Health	0
TID	1101
Target TID	526
Misc:	
Params 1	1

Here is a run down of what all this means.

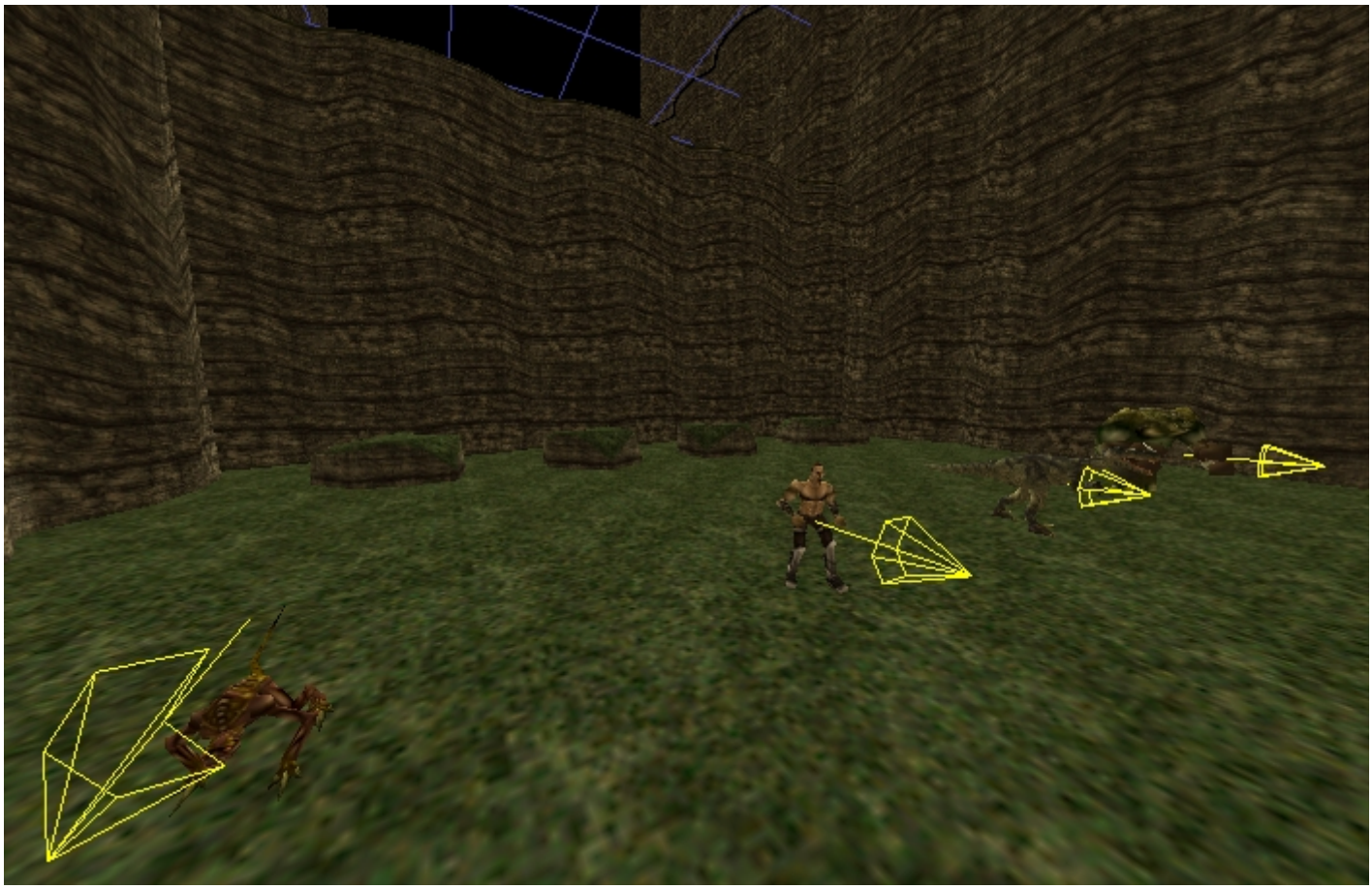
The “**Attack Chance**” value of the Mover is how fast it moves. Try experimenting until you find the value you want. Usually the value of 2 or 3 is adequate for most moving pillars like this.

The “**Health**” value of the Mover is the time that it takes in seconds for the Mover to reset to its original position. A value of 0 will cancel this effect so the pillar never resets to its original position.

As usual, the “**TID**” value is a unique number given to that object to distinguish it from all the rest. This number will be invoked when the enemy with the corresponding Target TID is killed. The trigger event just happens automatically and no input is needed on the creators end.

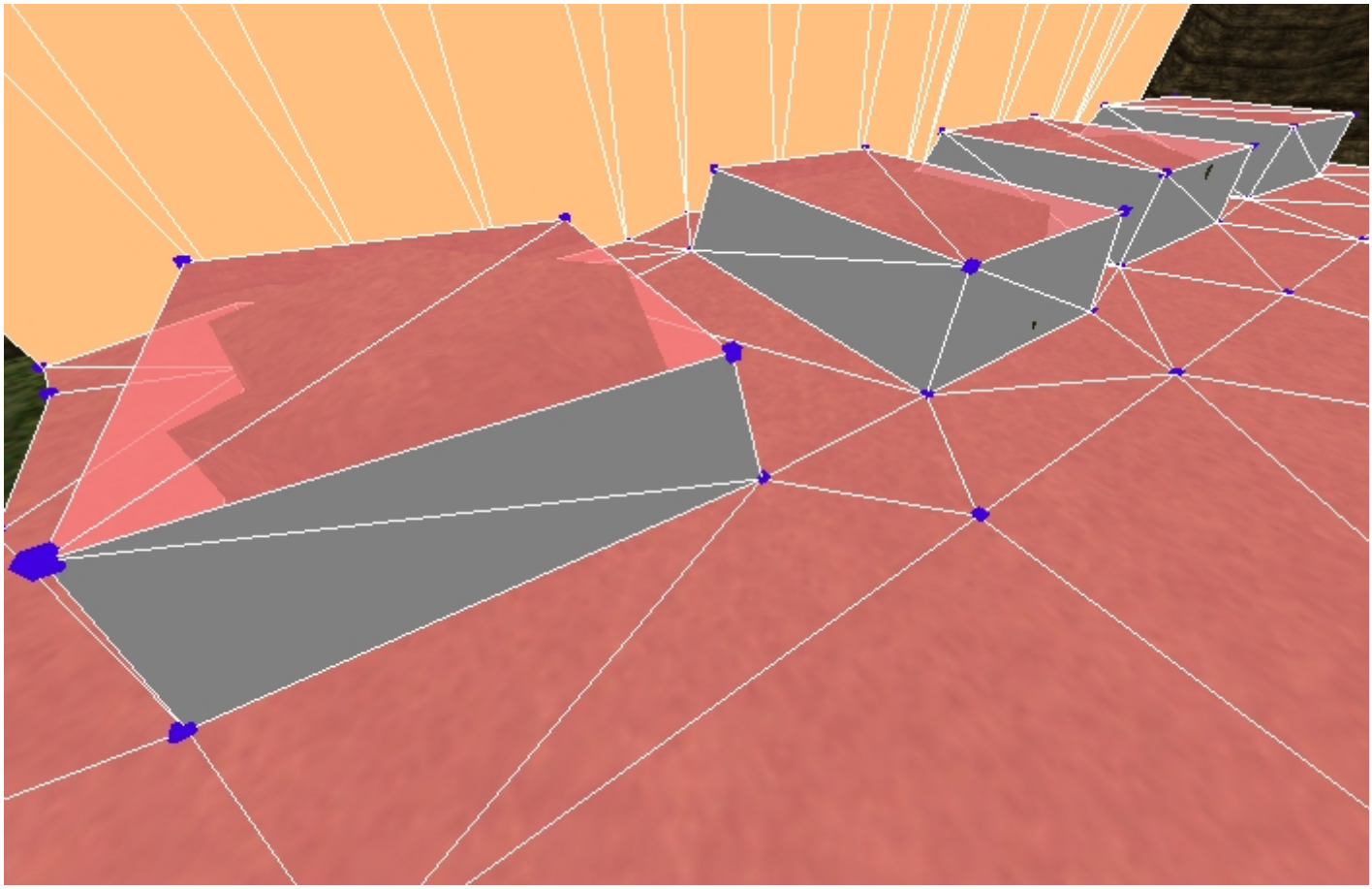
The “**Target TID**” value in this case is a sound ‘lookup’ number. You can go and open the file “/defs/FileLookup.txt” and see what number 526 matches up to. In this case, it matches up to “sounds/shaders/generic_141.ksnd” (Not very descriptive but it’s the one for the job). Sound 526 will play when the pillar moves.

The “**Params 1**” value is how far the pillar moves in game units. The game’s default grid size for Turok is 10.24 so the distance something will move is “Attack Chance” x “10.24. You can also use a minus value to have the Move move downwards.



Now for our sectors. We need to put the sectors around our object, fully encasing it inside. This will help the game to work out what is going to be moved. We also have to flag some sectors as “Cliff” and “Slope Test”.

Simply create something similar to what you see in the picture below.



You can see that our Movers are encased inside these sectors. The sides of which are grey because they have their “Cliff” and “Slope Test” flags checked.

When a mover is encased inside something like this, the game will automatically be able to work out what needs to move and when. If you do not encase your Movers properly, very strange things can happen to the surrounding sectors.

You do not need to assign any values at all to the grey sectors. They just know what to do!

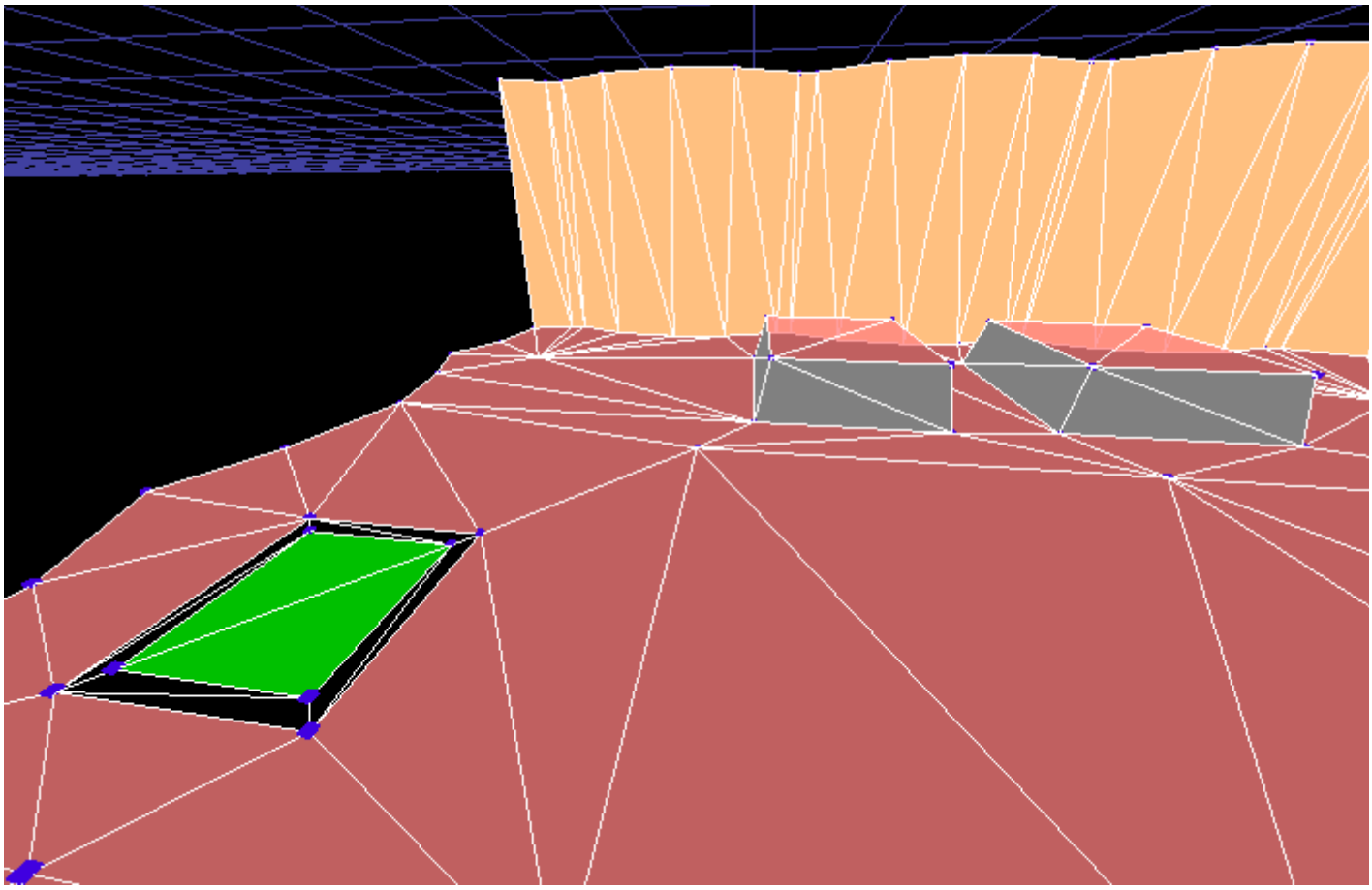
Once you have a set up like this, you can give your Movers their own unique “Attack Chance” and “Params 1” values. The “**Params 1**” values I have set to my pillars are 10, 3, 5 and 7. Feel free to check my set up out in tutoriallevel.map.

Of course, you don’t have to have to kill enemies in order to trigger any platform to move. You can set up a simple button to do the job for you. This time you only have to set up one TID for the pillars and have a button in the world for Turok to stand on.

Again though, the button is just a trick. When Turok stands on the Event sector, the trigger event will cause the button to animate as well as causing the pillars to move.

First we will need to put the button in the world. In Actor edit mode, right click on the floor > Add Actors > Movers > Mover_Door.

You will see that a button has been put on or near the floor. Place it wherever you like and set the Event sectors up underneath it so that when Turok walks on the Event sectors, the pillars will move. It’s exactly what we did before.



Above is how things look without the Actors and Meshes visible.

I have set up the **Arg 4** value for the Triggers to 1110. I have also set the **TIDs** of both pillars to 1110. The **Target TID** on the pillars is 526 (so the sound is played)

This time though, we want to set this up so that the pillars lower after a few seconds and the button trigger resets itself so it can be pushed again.

The only thing we have to do for this is give the pillars a "**Health**" value and our button an "**Attack Chance**" value.

I've given both the pillars a "**Health**" value of 5, after about 5 seconds when they've done fully extending to their resting position, they will sink back down into the ground and are ready to be triggered again.

Now for our button actor, we need to give it an "**Attack Chance**" value of 3. This is NOT 3 seconds. The game does some weird maths to determine the actual time that the value "3" is. It's the best timing fit I could find for this little sequence. Once this has been done, the following sequence occurs.

Turok stands on Event sector. Trigger call fires, lowering the button and raising the pillars. The pillars will be up for 5 and the button will reset back to its original position after "3"

Recap:

Mover_Door Actor.
AI Properties:

Attack Chance	3
TID	1110

Event Sectors.
Settings:

Arg 4	1110

Pillar 1.
AI Properties:

Attack Chance	3

General:

Health	5
TID	1110
Target TID	526

Misc:

Params 1	1

Pillar 2.
Same as pillar 1 but the "**Params 1**" value is set to 3.

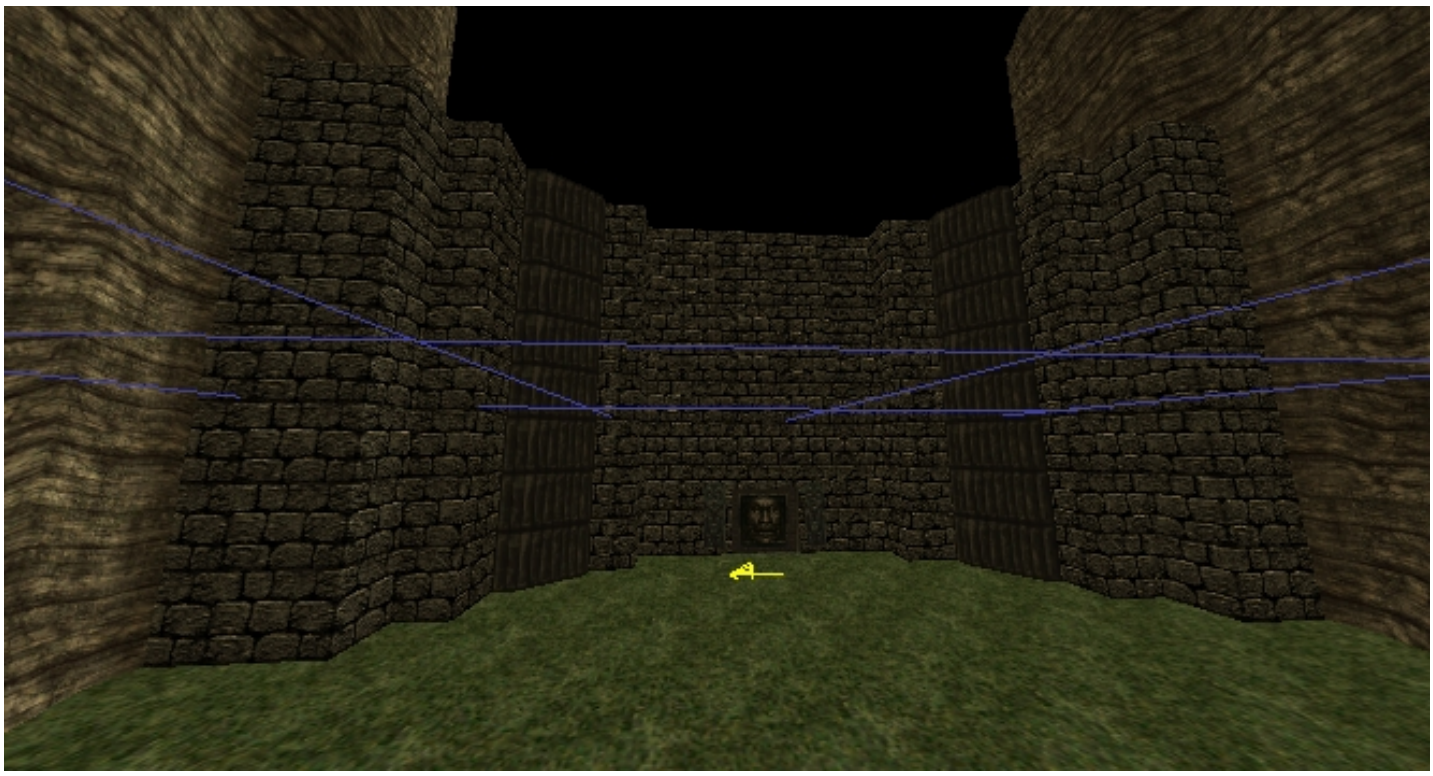
That's it! Go stand on your newly set up button and watch the magic happen!

Doors

For this guide, we should look at a simple door. Doors use a simple trigger event to open but it involves a few extra things. First we will need to put a brand new Mover_Door down so in Actor Edit Mode, right-click and go to Add Actors > Movers > Mover_Door.

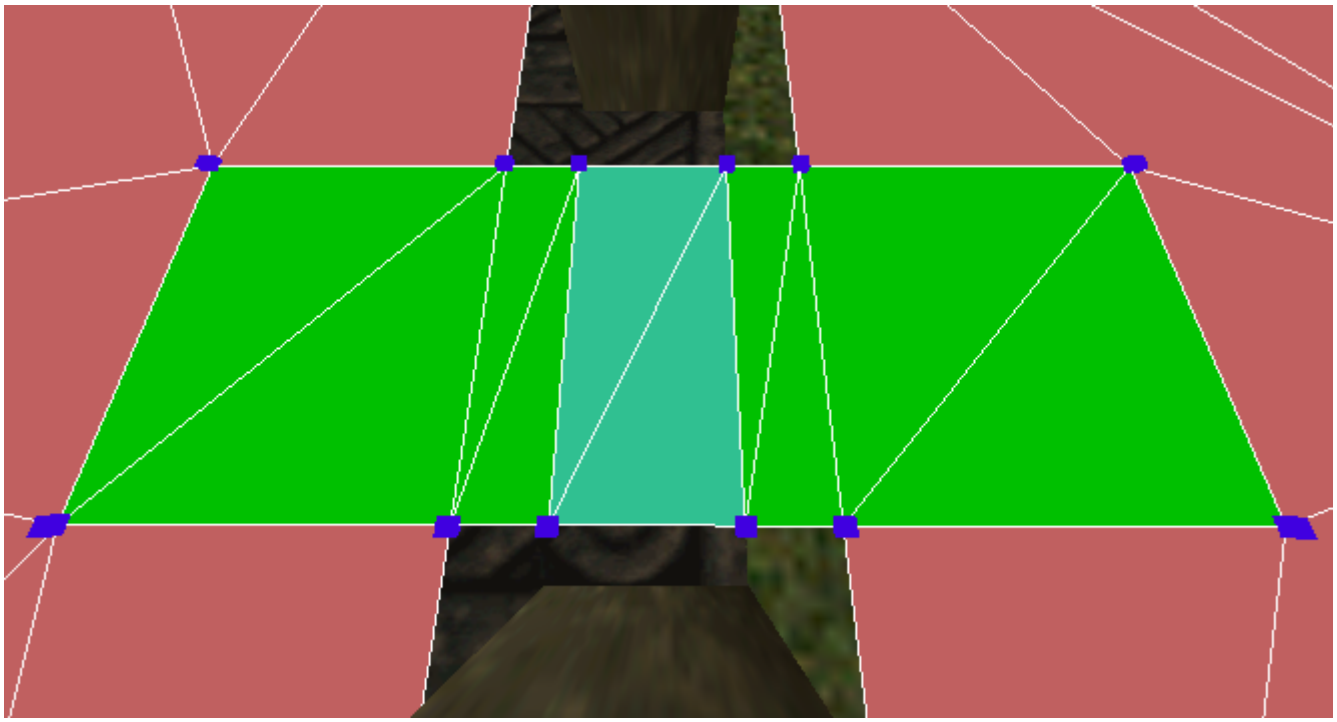
Once you have put that down, we need to give it a new model and a new animation file. Let's give it the model file "**models/dyn_door_catacomb_face.bin**" and animation file "**anims/dyn_door_catacomb_face_anim.bin**"

Now we have our door and it's ready for use. I've built a special place for the door to sit. A mysterious ancient temple!



The door is a simple trigger event. First we need to set up our trigger sectors so that when the player touches it, the door will open. These Event sectors need to extend out from the door so that when begin to open as Turok approaches them.

We also need to set up a blocking sector in the middle, where the door actually is, so that the player will not be able to immediately walk through the door before it is fully open.



The light blue sector in the middle is almost the same as the trigger sectors around it. It has the Event flag but it also has the Block flag enabled on it. The game is intelligent enough to work out when to disable the block when the door opens. Also, these sectors, unlike the Trigger sectors, do NOT need an Arg 4 value attached to them. They will just

work with no settings.

With that out of the way, you can go ahead and give your Event sectors a unique Arg 4 value (2440 in my case) and the Mover_Door should get a TID of 2440. When Turok now touches the event sectors, the event is triggered, the door will open and the Blocking flag will be disabled on the Block sector

The door will not close behind the player when they walk through this door for a very long time. In order to have it close behind them sooner, we need to simply alter the Mover_Door's "Attack Chance" value. Usually a value of 2 or 3 is enough. The door will close and the blocking sector will reactivate itself, waiting to be triggered again if the player decides to go back.

Oh and don't forget to give your doorway a Collide Ceiling. Or players will be able to clip through the doorway!

“Finishing” a level or campaign

All nice things have to come to an end sometimes and our level is no exception. Now that the temple is built, I’m going to have our map return the player to the Turok title screen when they walk on the portal that is present inside.

For this we are going to have to do several things. Create a brand new script file of our event, link our script to level through Mapinfo.txt and assign our Event sectors to the script to have the Event trigger when Turok walks over the sectors.

This is the room I have created for the finale. The teleport has Event sectors underneath. I’m going to give them the unique “**Arg 4**” value of 24601.

That’s all we need to do in-game and here’s why. When a sector has an “Arg 4” value but no corresponding TID or Target TID is found in the level, the game goes to the script system so see if it can find a matching value in a script. This is what we are going to do now.

In order to link a script to a map, we have to fill out the information in the Mapinfo.txt script first (we’ve done this before)

```
Map_MyCampaign  13
{
   title          "my campaign"
   map_1          "levels/tutoriallevel.map"
   mapid_1         101
   maxDistance_1   1000
   warptid         50000
         keys            0
   script_1       "scripts/map/MyScript.txt"
}
```

The script_1 has been changed to MyScript.txt which is the script we are going to create now. Be sure to give your script a much better name than mine! You cannot have more than one script for any one map, map_1 will be linked to script_1, map_2 will be linked to script_2 etc.

Now, go ahead and create a blank text file in your “scripts/map” folder and name it the same name you just put inside the mapinfo.txt file and add the following to it.

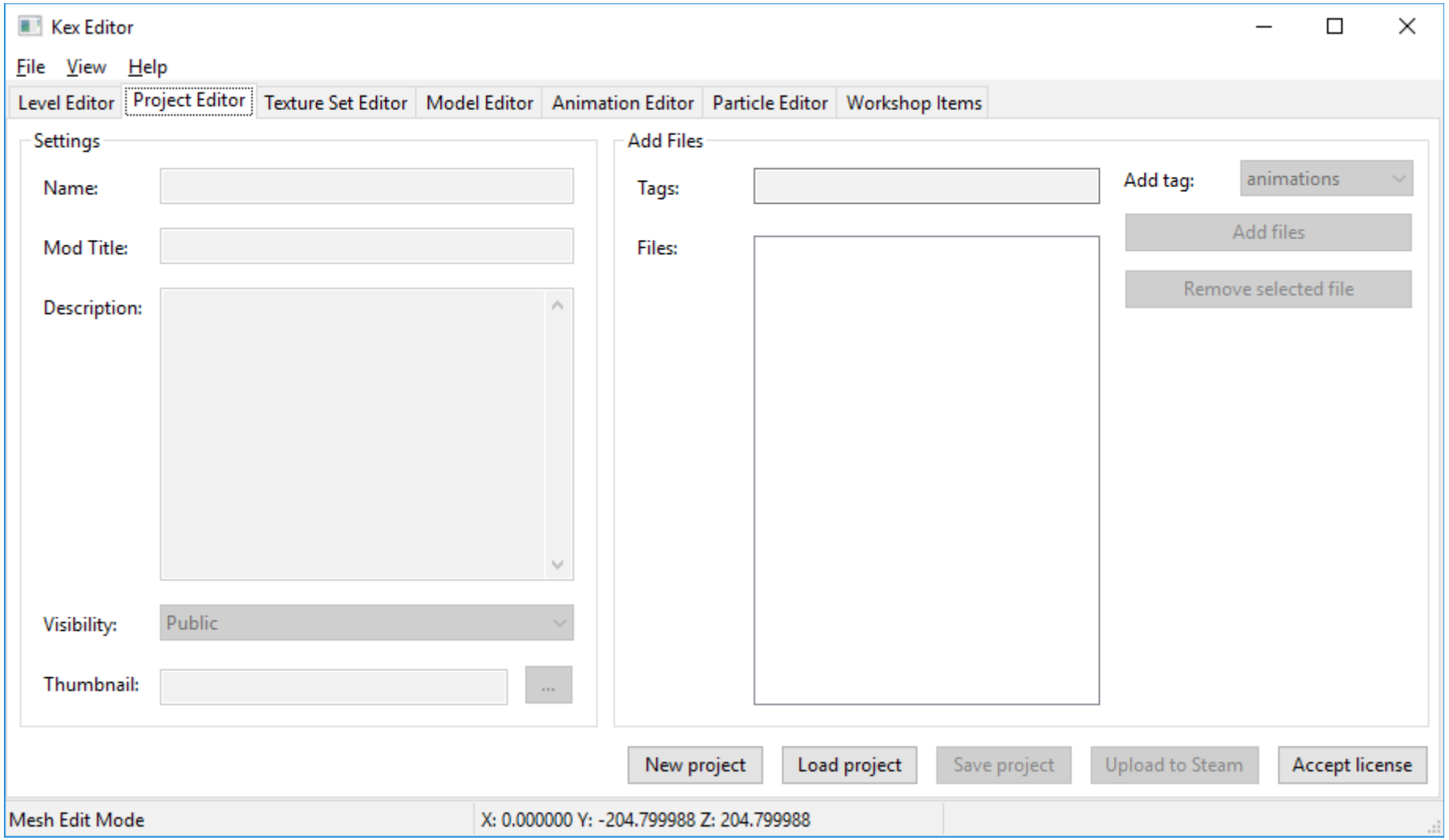
```
$script 24601
{
   if((instigator.Flags() & AF_ENTEREDAREAEVENT) != 0)
   {
                delay(-10.0f);
                Game.Restart();
                return;
   }
   $restart;
}
```

This script is a small script that will end the map and return the player to the title screen. It checks that if the sector in question is flagged with the AF_ENTEREDAREAEVENT (or Event Sector) flag, restart the game. The delay value is to start the restart of the game right away. The ‘F’ at the end of the number makes it a ‘floating point’ value and saves memory (only 4 bytes of data instead of 8!)

Congratulations, if you have been following along with this guide, then you (and I) have built our first Turok level from start to finish!

Preparing Your Mod For The Steam Workshop

The Project Editor



The Project Editor is where you will be finalizing your campaign. All the information you put in here gets put onto Steam for users to see when they go to download your mod or campaign. You will even use this tool to upload to Steam directly!

Now that we have our finished level or mod, we have to make it ready for people to download. Click on “New Project” to enable the Project Editor.

Here is a run-down of what you’ll need to fill in and where. If you don’t fill in any or one of the required fields, the editor will highlight the required field and tell you to fill it in.

Name: The Name of Your Mod. This is stored locally and is not displayed on Steam. You cannot have any special characters or spaces in this field.

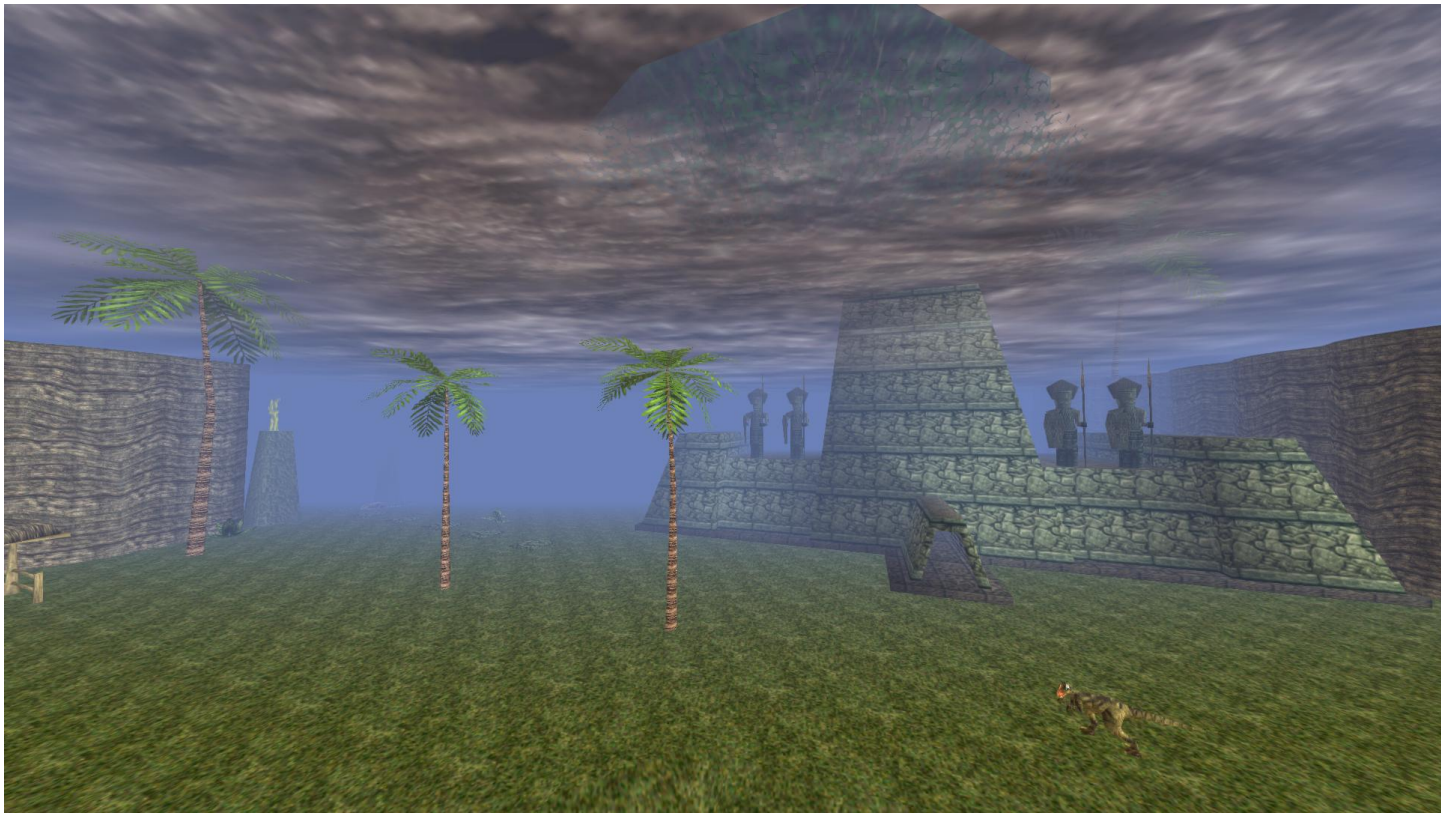
Mod Title: The Title of your Mod. This is what will be seen when people find your mod and click on it in Steam.

Description: Describe your mod/level/campaign to others and tell them why they should play it over everyone else’s.

Visibility: Three settings here. **Public**, **Friends Only** and **Private**. Public will set the mod to be downloadable by everyone as soon as it is uploaded to Steam. Do this if you are 100% sure that everything is correct and good to go right away. Friends Only means only the people on your friends list can see and download the mod. Linking the mod to other people will not work unless they are on your friends list. Private means that only you can see the mod on Steam but you can link the mod to other people, once the link is out there in the wild, there is no stopping it from spreading. Visibility of the mod can be changed at any time later in the Steam Workshop so Private is recommended so you can make sure everything is how you want it to be before unleashing your mod out onto the public.

Thumbnail: Here is where you will let your artistic side shine! All mods can be given a thumbnail to make it stand out from all the other mods on the steam Workshop. It is the first image that the player of your mod will see. It can be a picture of the mod in question, a picture of your level or something custom that you make yourself. There is a file size limit of 1 Megabyte and should be 16:9 in dimension.

If you want a nice picture for your level, you can remove the Cross-hair and HUD (Heads Up Display) from the screen in the Gameplay > HUD Options menu. You can remove the gun model from the screen by scrolling with the mouse wheel to select another weapon. Be quick! You can use the F5 Key to take a picture of the screen.



Your screen shot does not have to be included inside the KPF itself.

Tags: Your mod, level or campaign will have to be listed certain categories so it can be found easily by people who are looking for what you have made. Tagging your creation with the appropriate tag will help with this. To remove a Tag, select it again from the drop-down list.

Files: This is where you will select your KPF file for uploading to Steam. The Add files and Remove selected file buttons do what you expect them to do.

As for the buttons at the bottom of the screen:

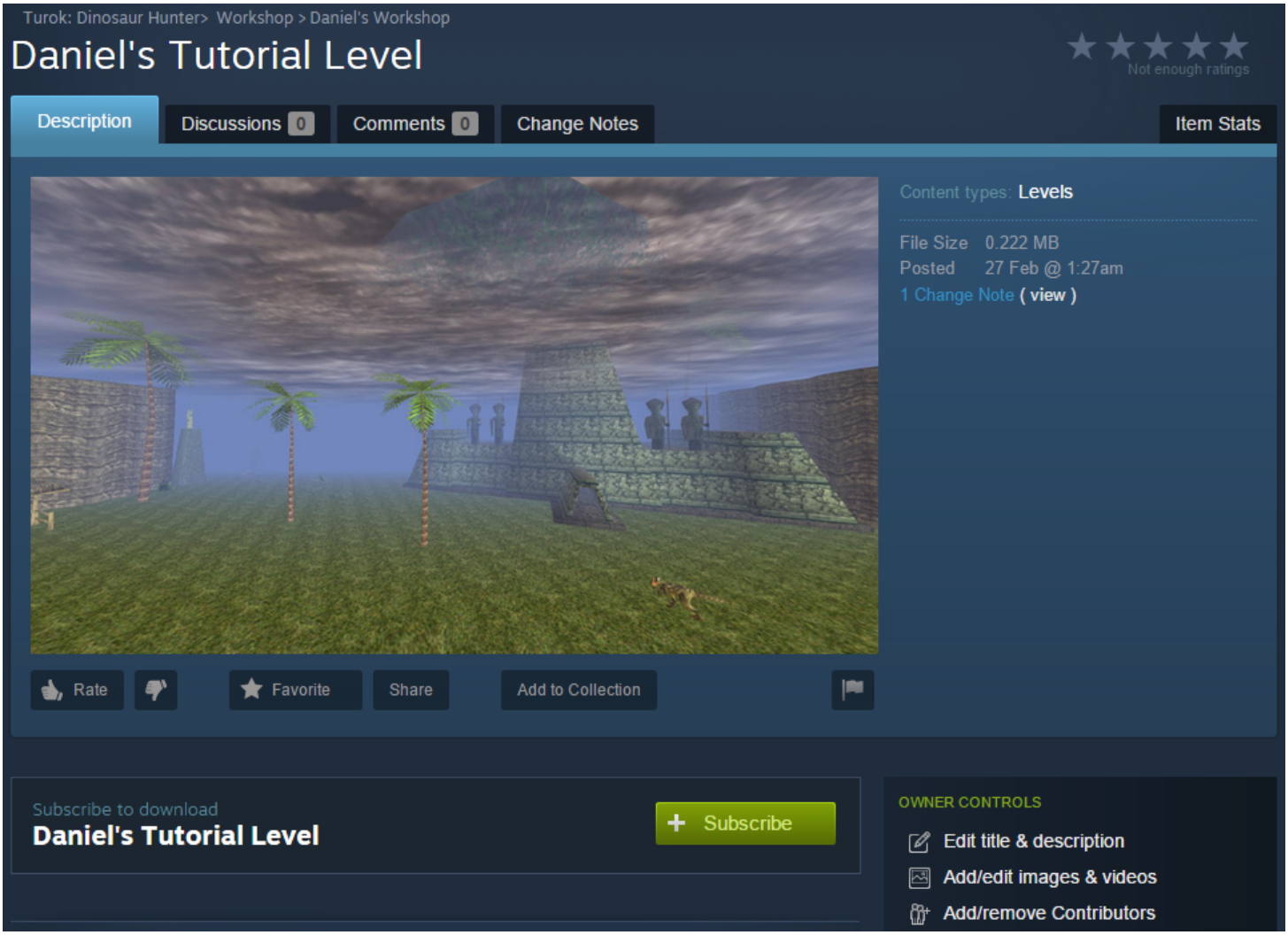
New Project: Starts a New Project.

Load Project: Will load a .kprj file that has been created by the 'Save project' button.

Save project: Will save all the information that has been entered into a small .kprj document that can be loaded from the Load Project button. The information is stored in plain text so you can easily open it with a Text editor and edit the file manually.

Upload to Steam: This is the magic button that will upload everything you've done to the Steam Workshop.

Accept License: This will launch a new window where, if you have not done so previously, you must read and accept the License Agreement to be able to upload things to the workshop.



How to create a KPF file

Creating a KPF file is extremely easy. All a KPF file is is an extension renamed ZIP file!

You may need to enable “File Name Extensions” in Windows so that you can see them, if you can see files with a dot and then a one, two or three letter suffix then you can already see file extensions.

All you have to do is gather up your scripts, your altered Mapinfo.txt if you have a campaign to upload) as well as the level files themselves, and put them in their own folder **KEEPING THE FOLDER PATHS** the same (if you do not do this, things will not work when you come to trying out your creation). Now, to make a zip file, you can either use the built-in functionality in Windows or you can use a third-party app such as 7-Zip to zip up your file. Just make sure you create a ZIP file and not a RAR file, a 7Z file or anything else like that.

This is also a good time to make sure you have given your script files unique enough names so that they don’t potentially clash with other downloaded mods. Having a script simply called MyScript.txt is not the greatest idea. Be sure to update your Mapinfo.txt with the new script names if you do decide to make things a little more unique.

Once you’re sure you have everything in place and ready, you can go ahead and upload your KPF file to the steam Workshop by selecting your compiled KPF, making sure all the correct fields are in place and then, hitting that upload button! The editor will then do all the work for you and upload your KPF.

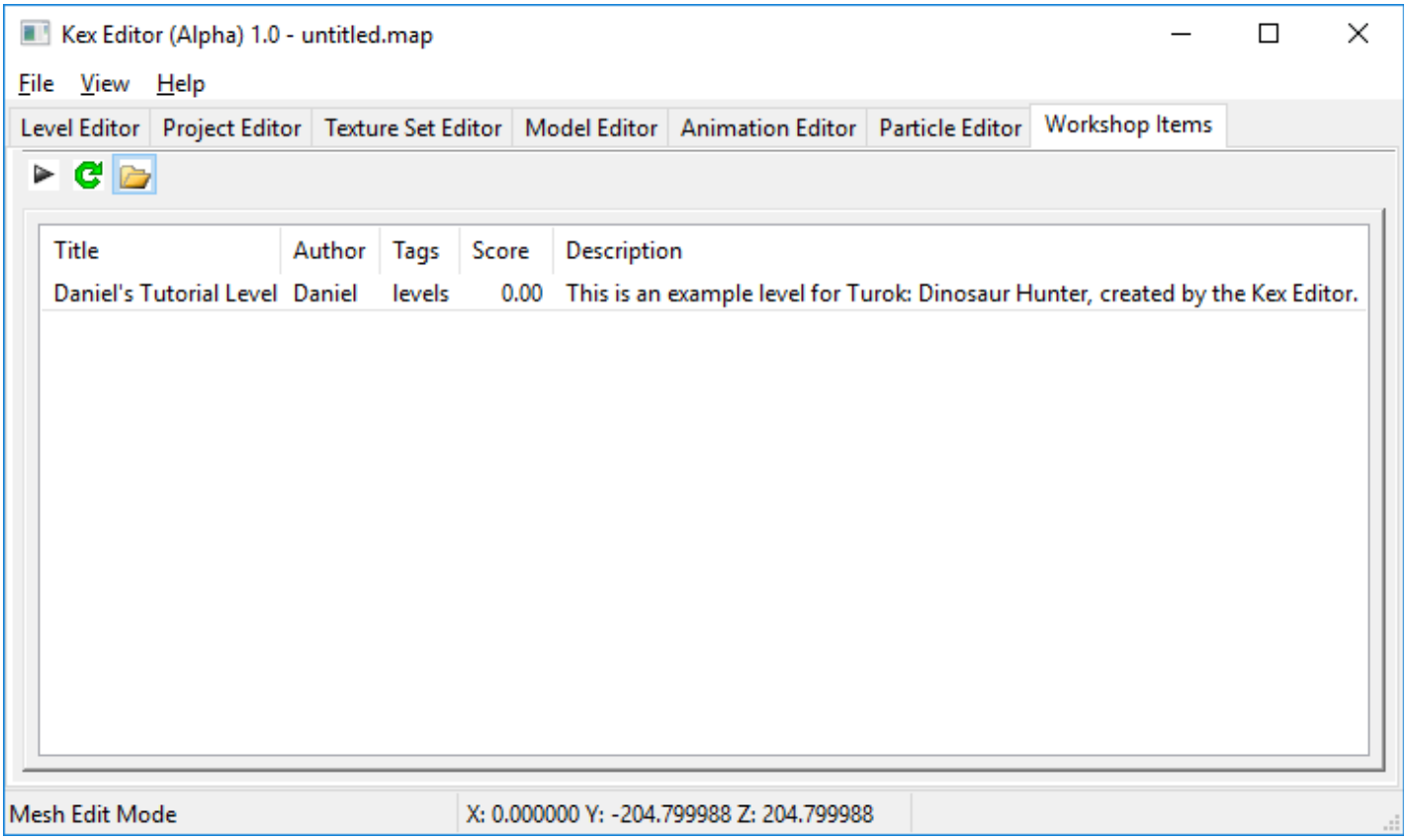
Once it has done uploading, you can immediately check out your upload by going to your Steam Profile in Steam and go to your workshop Items. You will see it listed in your workshop with the art thumbnail you selected.

You can then go and do many things with it from here. Things like: Changing the title or description, add and edit images, add and remove other people who may have contributed to the mod as well as other things. This guide will not cover the Steam Workshop in detail but go ahead and explore what you can do. If you set your mod to be private, you can make all the necessary changes here.

Playing Mods from the Steam Workshop

You may have downloaded some mods from the steam Workshop already and are wondering how to play them. The main Turok executable isn't used for loading the mods you have. Instead, you must use the "workshop Items" Tab of the editor.

The editor connect directly with your Steam account to work out what items you have subscribed too. Here I have subscribed to my own level for demonstration.



Everything here is pretty straight forwards, you can see the name, author, score and description. You highlight the mod you want to play and press the Play button at the top to play the mods you have selected. You can select more than one mod by holding down the CONTROL key and clicking on the mods you want to play. The game will then launch with the selected mods.

Common Editor Messages

Editor: “Vertex must be behind the edge”

This usually means that you need to reposition a vertex. It is probably exactly above another that is already connected to a sector you're trying to extend from. See the “Bad Sectors” section of this guide for more information on how to solve this problem.

Workshop: “Status: Failed to prepare upload”

This usually means that you’re not logged into Steam or have not started the editor through Steam. Re-log into Steam and/or run the editor through Steam and try again.

Handy Tips

Unsure if some sectors don't have the same properties as others? Highlight a sector you know has the properties you want and right-click it. Click “Select all matching sectors” and the editor will select all sectors that have the same exact flags as the one you highlighted. Useful if you think you might have missed some when trying to flag “Draw Sky” or something similar.

Remember to create your first sector, you MUST select the vertices in clockwise order or it will not be created properly. This also applies to the Draw Shape Mode, create everything as shown on the Draw Shape Mode page and everything will work properly.

If you make a mistake and need to delete vertices, always hit the Delete key on the keyboard TWICE to fully get rid of them. Data is left behind when they are deleted and pressing the Delete key twice removes the extra data. It’s just the way the engine works.