Tutorial on Creating a Simple 3D Navigable Terrain

1. What you Need

GIMP, download from:

http://downloads.sourceforge.net/gimp-win/gimp-2.6.8-i686-setup.exe

The installer file is:

gimp-2.6.3-i686-setup.exe

GIMP is a free software for graphics editing and creation. Similar to Adobe Photoshop.

2. 3D Navigable Terrain Design Methodology

1. Create the terrain using graphic tools, eg, Photoshop or Gimp.
2. Import the terrain graphics into your DarkGDK program.

3. Terrain Design Methodology

We will need to create 3 graphic files and download a sky picture:

a. Height map (call it myheightmap.bmp)

b. Texture (call it mydetail.jpg)

c. Detail map (call it mysky.jpg)

d. Sky (No need to create just google for one sky jpg).

3.1 Creating Height Map

Use GIMP to create this 256 x 256 *height map.* Save as BMP file “myheightmap.bmp”:

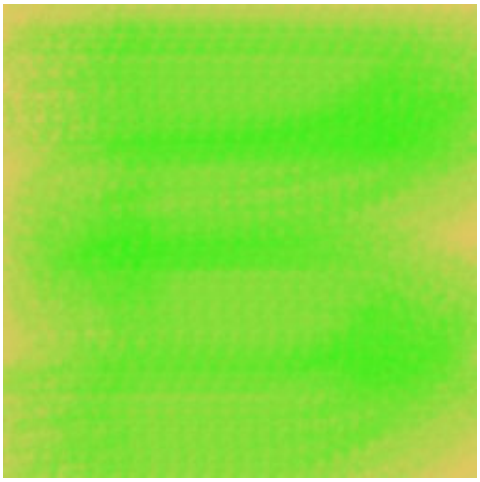
a. Bucket all black.
b. Lasso mountain peaks.
c. Select Feather
d. Bucket each Lassoed peak with White.

A *height map* (also known as *bump map*) is used to create mountains and hills. The black part means that it is ground level. The white parts are the hills and mountains. The whiter the color the higher the mountain.

Save all graphic files in your DarkGDK project folder.

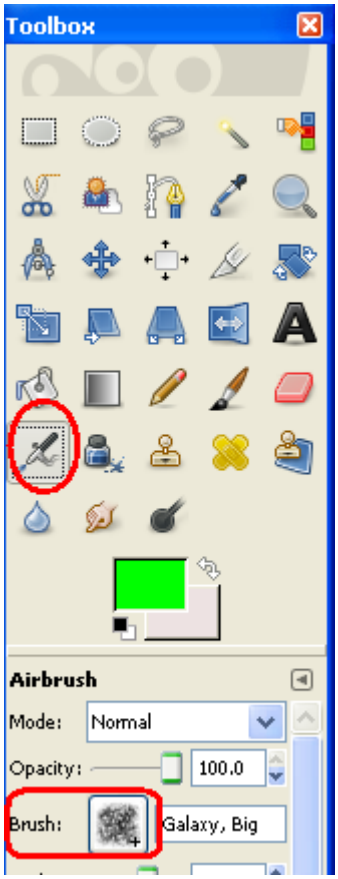
3.2 Creating Texture

Also use GIMP to create this 256 x 256 texture and save as JPG file. Save as JPG file "mytexture.jpg":



a. Use Bucket to bucket the background as light brown.

b. Then select green and use airbrush to spray the green grass color onto the ground:



3.3 Details Creation

Then use GIMP to create this 256 x 256 detail (which will be tiled over the texture above) and save it as JPG file :

Make sure you select Filters, Map, Make Seamless so that the tile can be tiled without noticeable edges.

3.4 Sky Sphere Creation:

Download any sky pics from Internet and rescale it to 512 x 512 and save as JPG file:

Source Code

```
#include "DarkGDK.h"

void DarkGDK ( void )
{

	dbSyncOn ( );
	dbSyncRate ( 60 );

	// we are going to use a skybox in this demo and it will
	// be scaled up to quite a large size, this will initially
	// result in it being outside of the cameras default range
	// therefore it will not be drawn, to adjust this we simply
	// increase the cameras range
	dbSetCameraRange ( 1.0f, 30000.0f );

	// two textures are going to be used for the terrain, the first
	// will be the diffuse part and the second will be used to
	// create extra detail on the terrain
	dbLoadImage ( "mytexture.jpg", 1 );
	dbLoadImage ( "mydetail.jpg",  2 );
	//dbLoadImage("mysky2.jpg", 3);


	//CREATE SKY SPHERE
	dbMakeObjectSphere(4,-3000);
	//dbTextureObject(4,3);
	dbSetObjectTexture(4,3,1);
	//dbScaleObject ( 4, 1, 1, 1 );
	dbPositionObject(4,0,0,0);

	//CREATE TERRAIN
```

```
	// the first step in creating a terrain is to call the
	// function dbSetupTerrain, this will perform some internal
	// work that allows us to get started
	dbSetupTerrain ( );

	// now we can get started on making the terrain object
	dbMakeObjectTerrain ( 1 );

	// here we pass in a heightmap that will be used to create the terrain
	dbSetTerrainHeightMap ( 1, "mymap.bmp" );

	// now we set the scale, this will have the effect of making
	// the terrain large on the X and Z axis but quite small on the Y
	dbSetTerrainScale ( 1, 3.0f, 0.6f, 3.0f );

	// adjust the lighting for the terrain, this function takes the ID
	// number, then a direction for the light, then 3 colours for the light
	// and finally the scale, by using this function we can adjust the
	// overall colour of the terrain
	dbSetTerrainLight ( 1, 1.0f, -0.25f, 0.0f, 1.0f, 1.0f, 0.78f, 0.5f );


	// in this call we're telling the terrain that its diffuse texture
	// will come from image 1 and its detail texture will come from
	// image 2
	dbSetTerrainTexture ( 1, 1, 2 );

	// once we have set all properties of the terrain we can build it,
	// at this point it gets created and added into your world
	dbBuildTerrain ( 1 );


	// position the camera
	dbPositionCamera ( 385, 23, 100 );

	// adjust texture properties of sky box
	//dbSetObjectTexture ( 2, 3, 1 );

	/*dbLoadObject( "short_knight.x", 200 );
	dbSetObjectLight       ( 200, 1 );
	dbPositionObject(200,0, 0, 0);*/

	// now onto our main loop
	float angle=0;
	while ( LoopGDK ( ) )
	{

		// let the user move the camera around with the arrow keys
		dbControlCameraUsingArrowKeys ( 0, 2.0f, 2.0f );
		//if(angle>=360) angle=0;
		//dbRotateCamera ( 0, 0, angle++, 0 );

		// find the ground height of the terrain
		float fHeight = dbGetTerrainGroundHeight ( 1, dbCameraPositionX ( ),
		dbCameraPositionZ ( ) );

		// reposition the camera so it is directly above the ground
		dbPositionCamera ( dbCameraPositionX ( ), fHeight + 10.0f, dbCameraPositionZ( ) );

		// update the terrain
		dbUpdateTerrain ( );

		// update the screen
		dbSync ( );
	}
}
```