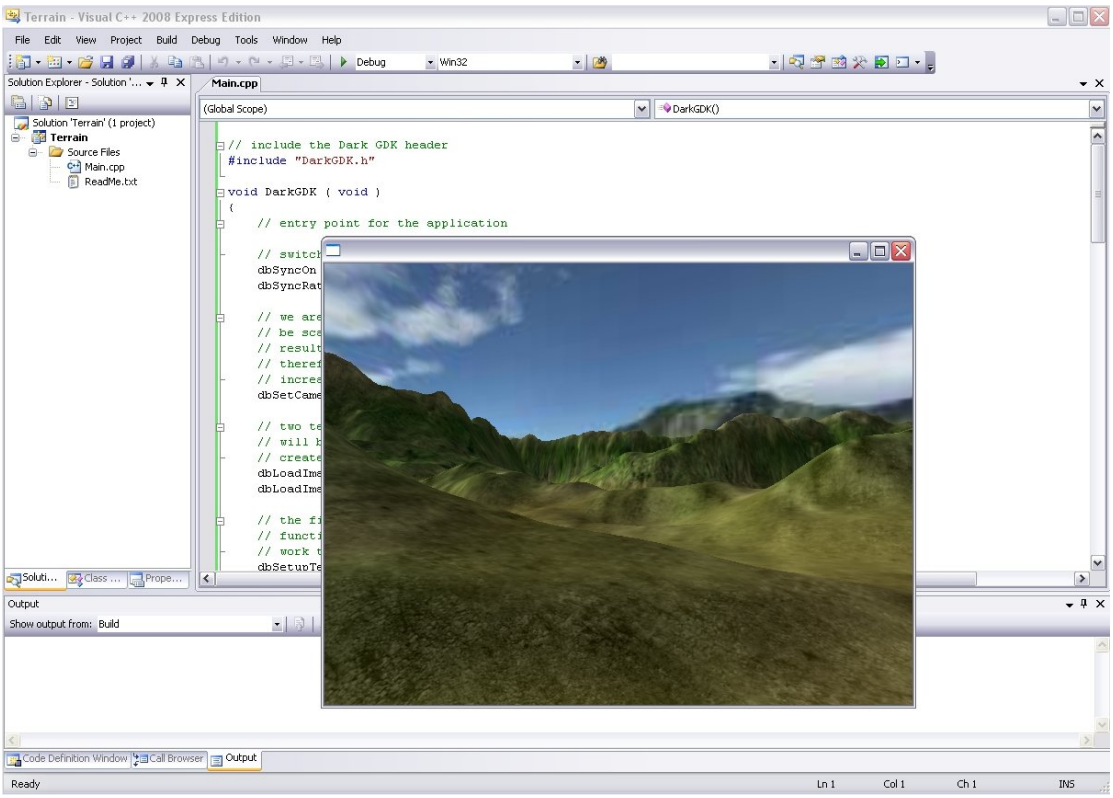
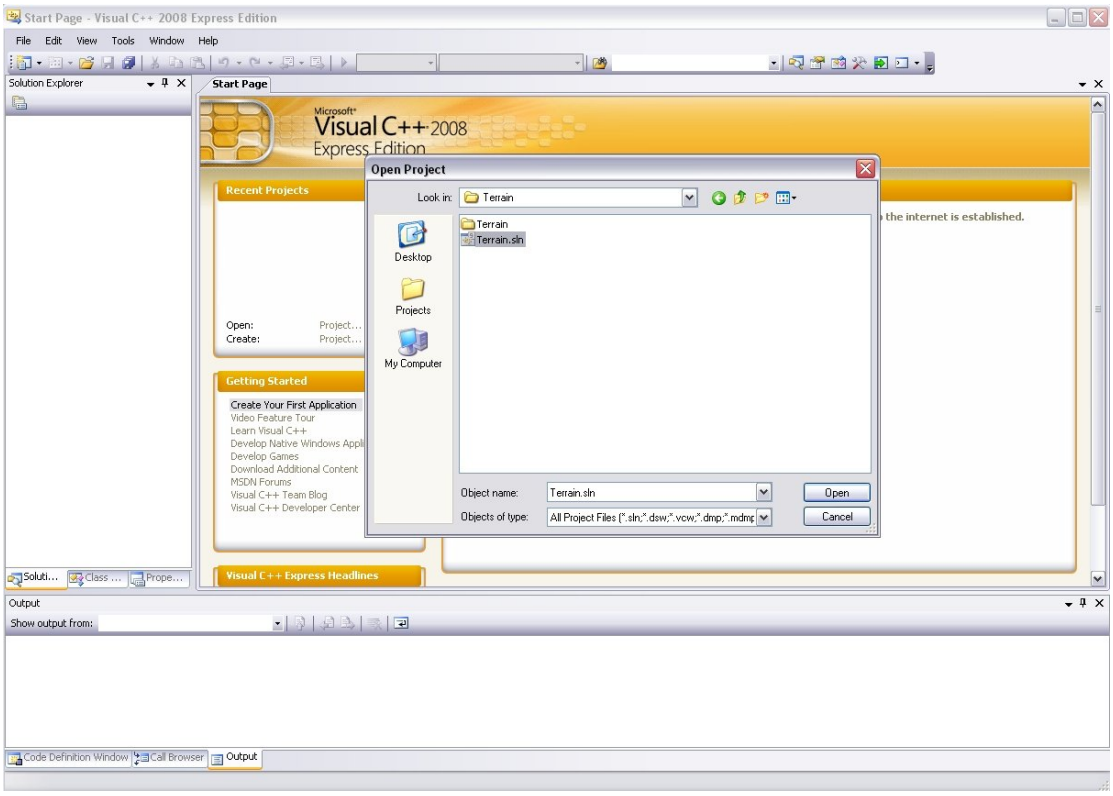
Game Terrains

Introduction
In this tutorial we're going to look at the terrain functionality included in Dark GDK.



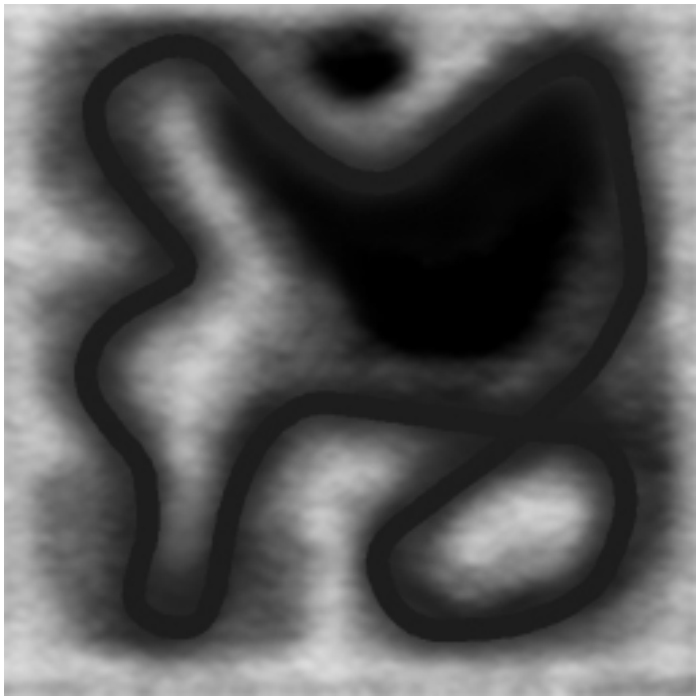
Opening the project
Launch Microsoft Visual C++ 2008 and go to the File menu and select Open and then Project. From here navigate to the directory where Dark GDK is installed. By default this is C:\Program Files\The Game Creators\Dark GDK. Now go into the Tutorials folder and then the Terrain folder. Finally open the solution file for Terrain.



The project contains a source file named Main.cpp. This file contains all the code to load a terrain, skybox and move around it using the camera.

Creating a terrain

In order to create a terrain we need to supply a height map image. A height map is used as a way of representing our terrain in a 2D image. This is then imported into our world and transformed into a 3D object. Here's the height map we're going to use in our demo:



This image is processed in such a way that light areas will create height and darker areas will create depth. It may not be so clear at the moment but when you see the program run it should make sense.

Dark GDK has a collection of functions that allow us to take a height map and generate a terrain. Here are the key functions we will be using:

```
void dbSetupTerrain ( void );

void dbMakeObjectTerrain ( int iID );

void dbSetTerrainHeightMap ( int iID, char* szFile );

void dbSetTerrainScale ( int iID, float fX, float fY, float fZ );

void dbSetTerrainLight ( int iID, float fX, float fY, float fZ, float
fRed, float fGreen, float fBlue, float fScale );

void dbSetTerrainTexture ( int iID, int iDiffuse, int iDetail );

void dbBuildTerrain ( int iID );
```

dbSetupTerrain
This function is used as a method of setting up some internal settings. It is necessary to call this function before making any terrains.

dbMakeObjectTerrain
The initial creation of a terrain object starts with a call to this function. It takes an ID number and this ID number is shared with those from the Basic3D function set. For listings of what functions are included in each set please view the function help. For example, if you have previously loaded an object into ID 1 you cannot then use an ID of 1 when making the terrain, unless you first of all delete the current object using this ID. Calling this function will simply create a place holder object. Nothing will be visible directly after calling this function as there is still more work to be done. The next few stages involve setting properties of the terrain.

dbSetTerrainHeightMap
This function and others that set properties of a terrain can be called after the terrain has been made using dbMakeObjectTerrain. This particular function takes an ID number and a pointer to a string. The ID number is used as a way of deciding which terrain you want to set a property for. The second parameter is used to determine the file that will be used as the height map.

dbSetTerrainScale
When a terrain has been created its initial scale may not conform to the results you desire. This can be easily controlled by calling dbSetTerrainScale. This function takes three parameters that allow you to control how the terrain is scaled on the X, Y and Z axis. In most cases it will be necessary to scale the terrain out on the X and Z axis

and perhaps reduce its size on the Y axis. We'll see how this works later on.

dbSetTerrainLight
When the terrain is built, lighting data will be generated for it. By using this function you can control the light properties. This function takes several parameters. The first is an ID number for the terrain. The next three are X, Y and Z values that define the direction of the light. As an example, you could set the light direction to 1, 0, 0 meaning the light is pointing from left to right on the X axis. The next three parameters are used to define the red, green and blue colours of the light. These are specified as floating point values with 0.0 being no colour and 1.0 being full colour. For example, if you specified these parameters are 1.0, 0.0, 0.0 then the light would be bright red. The last parameter controls the scaling or intensity of the light. A low value is used to reduce the power of the light e.g. 0.0 will result in a very dark terrain whereas a high value such as 1.0 will create a very brightly lit terrain.

dbSetTerrainTexture
This function is used to control which textures are applied to the terrain. It takes three parameters. The first is an ID number for the terrain while the second is an ID number for a diffuse image and the third is an ID number for a detail image. The way this works is that you can load an image using dbLoadImage and then pass the ID number from this image into dbSetTerrainTexture. For example, you may load an image into ID 10 that will be used for your diffuse texture and an image into ID 20 that will be used as your detail texture. Now say you had made a terrain using ID 1 you could call this function as dbSetTerrainTexture (1, 10, 20); The diffuse image is used as a way of colouring the terrain. Say you had created an image and filled it with blue and used this as your diffuse texture. When your terrain would get built it would be coloured blue. It would also be affected by the light settings. The detail image is an image that gets tiled over the whole terrain. It can be used as a way of applying detail onto the terrain. This will become clearer when we check out the demo.

dbBuildTerrain
Once all properties of a terrain have been set you can build it. This is the stage when everything will come together and be inserted into the world.

The key here is getting the order of calls correct. You must first ensure the terrain functions are ready to be used by calling dbSetupTerrain, then proceed to making your terrain, setting any properties and finally building the terrain.

Examining the code
The project we have opened already contains all the source code to handle the creation of a terrain, the loading of a skybox and movement with the camera. In the next few pages we're going to break down the source code and see what each part does.

Initial Setup
As with most Dark GDK programs the initial few lines handle some global properties. In this case the sync rate is turned on and the maximum refresh rate set to 60 fps.

```
dbSyncOn ( );
dbSyncRate ( 60 );
```

The next line adjusts the camera range. By default the camera will have a range of 1.0 – 5000.0. This means that anything past a range of 5000.0 will not get drawn. In our example we're going to create a skybox and scale it so that its larger than the terrain. This will result in the skybox being past the default camera range so in this line we make the range a lot larger:

```
dbSetCameraRange ( 1.0f, 30000.0f );
```

Images
In the next two lines images are loaded. These images are going to be used for our diffuse and detail textures:

```
dbLoadImage ( "texture.jpg", 1 );
dbLoadImage ( "detail.jpg", 2 );
```

Before we continue let's check out the file "texture.jpg":

This image is going to be stretched over the terrain we created and will provide the colour information.

Here we can see “detail.jpg”:

This image is going to be tiled over the terrain. By repeating this texture over the terrain we can create the illusion of detail.

Making the terrain
With our images loaded and in place we can make a start with loading the terrain. As mentioned earlier we must first make sure the terrain functions are ready to use, we can then make our terrain, set properties and build it. In this block of code our first line is a call to dbSetupTerrain:

```
dbSetupTerrain ( );
```

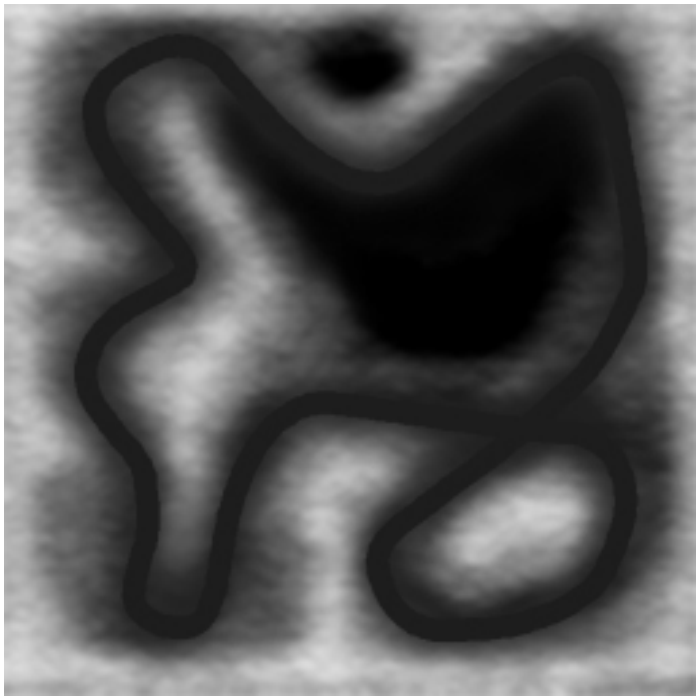
The next line makes the terrain and gives it an ID number of 1:

```
dbMakeObjectTerrain ( 1 );
```

Moving on from this we set the height map:

```
dbSetTerrainHeightMap ( 1, “map.bmp” );
```

The file “map.bmp” contains our data that will be transformed into a 3D object:



In the next line we adjust the scale of the terrain. It is scaled out 3 times on the X axis, reduced by 0.6 on the Y axis and scaled out 3 times on the Z axis:

```
dbSetTerrainScale ( 1, 3.0f, 0.6f, 3.0f );
```

The next line of code adjusts the terrain lighting:

```
dbSetTerrainLight ( 1, 1.0f, -0.25f, 0.0f, 1.0f, 1.0f, 0.78f, 0.5f );
```

It sets the light so it is moving to the right and downwards slightly. The colours are set to create a yellow light. In the last parameter the scale is set to 0.5f so the lighting is not too influential.

Following on from this the terrain textures are set:

```
dbSetTerrainTexture ( 1, 1, 2 );
```

With all properties set we can now build the terrain:

```
dbBuildTerrain ( 1 );
```

Making a skybox
The next block of code deals with making a skybox:

```
dbLoadObject ( "skybox2.x", 2 );
dbSetObjectLight ( 2, 0 );
dbScaleObject ( 2, 30000, 30000, 30000 );
```

The three lines of code are fairly simple. A model is loaded. This model is a box with textures on to represent the sky. Lighting is then switched off for it as

we don't need the skybox to respond to light. In the final line the skybox is scaled up to a large size. This is done to make the skybox much larger than our terrain.

The camera
There is one final line of code before we enter our main loop and this line sets the position of the camera:

```
dbPositionCamera ( 385, 23, 100 );
```

This will position the camera close to one of the corners of our newly created terrain.

Main loop
Our main loop provides functionality that allows the user to move around the terrain and also makes some calls to update the terrain and screen:

```
while ( LoopGDK ( ) )
{
	dbControlCameraUsingArrowKeys ( 0, 2.0f, 2.0f );

	float fHeight = dbGetTerrainGroundHeight ( 1, dbCameraPositionX ( ),
dbCameraPositionZ ( ) );
	dbPositionCamera ( dbCameraPositionX ( ), fHeight + 10.0f,
dbCameraPositionZ ( ) );

	dbUpdateTerrain ( );
	dbSync ( );
}
```

The first line in the main loop is a function from the camera feature set. By calling this function it allows the user to move the camera with the arrow keys. It takes three parameters. The first is an ID number for the camera. By default the initial camera is 0 so we pass that in. The second parameter defines how fast we can move around the world using the up and down keys. This gets set to a value of 2.0. The last parameter controls how quickly the camera turns when the left and right keys are pressed. This property is set to 2.0.

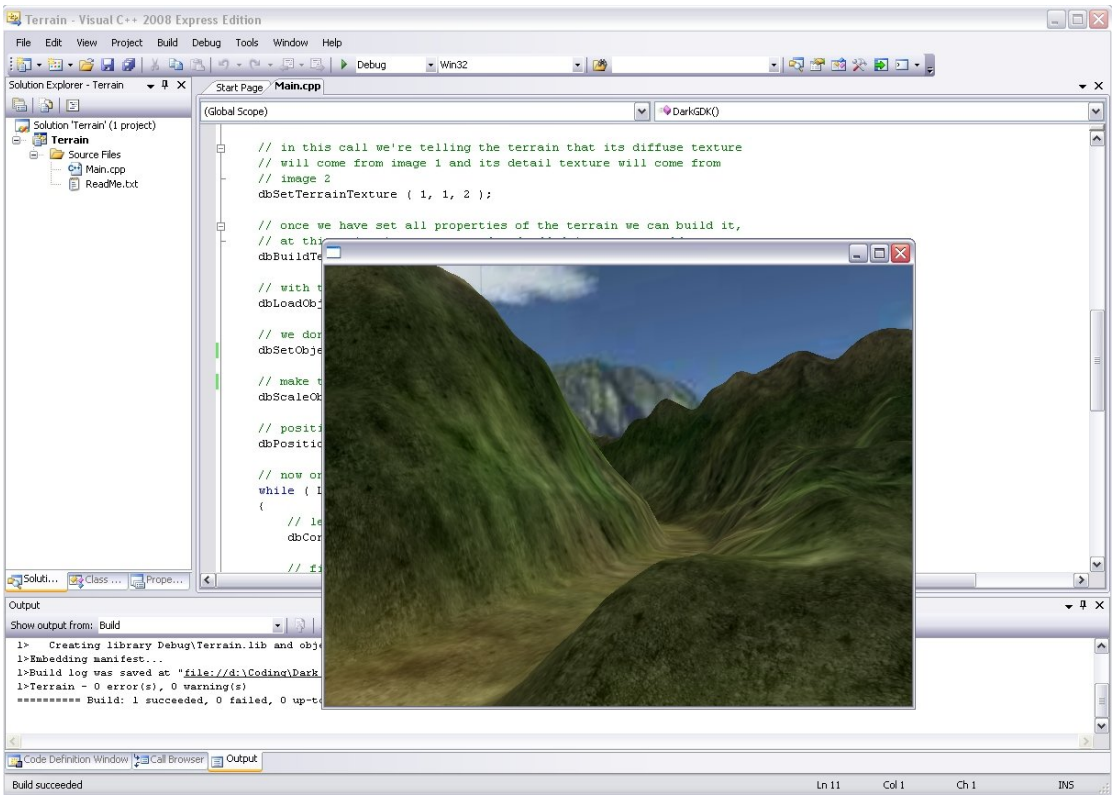
The second line is a call to a function in the terrain feature set. This function allows us to determine the current terrain height at the given location. So we pass in the ID of the terrain and the current camera position on the X and Z axis. The value assigned to fHeight will be the height at this point.

Moving on from this we position the camera. This function takes three parameters, X, Y and Z coordinates. We only want to adjust the Y position so we set the X position to whatever it is currently by calling dbCameraPositionX. For the Y position we set it to fHeight + 10.0f. The variable fHeight contains the height on the ground of the terrain. By adding 10.0f to it we move the camera off the ground and into the air.

Another important aspect of terrains is that once per main program loop you have to make a call to the function dbUpdateTerrain. This function does not have any parameters. It is responsible for going through all terrains and updating some internal information. An ideal place to make this call is directly before dbSync as in our code listing.

The final part of our main loop is a call to dbSync. This will update the screen.

Testing the program
With everything in place it's time to try out the code and see the results. Go to the Debug menu and select Start Debugging. While the program is running use the up and down keys to move forwards and backwards and the left and right keys to turn the camera left and right.



Conclusion
In this tutorial we've gone through the steps involved in creating a terrain and moving around it by controlling the camera.

It is well worth spending some time trying to understand the correlation between the height map and the terrain created. Try opening "map.bmp" in an editor such as Paint. When you have opened the image try drawing some white and black blotches. Now run the program again and see how this has affected the terrain. Also try changing the image to make it into a black rectangle – now notice how when you run the program the terrain is completely flat. Try adding a white area onto this and you will see how you can create a hilly area.

We would also recommend spending some time tweaking properties of the terrain. Try experimenting with the values passed into dbSetTerrainScale and see how you can make the terrain larger or smaller. As a test change the Y parameter in this function call from 0.6 to 1.0. Notice how much taller all of the high areas become. Now try a value of 0.3 and see how different this is from the original value used. Also try experimenting with the values passed into dbSetTerrainLight. Try adjusting the direction, colours and scale to see how this affects the terrain.