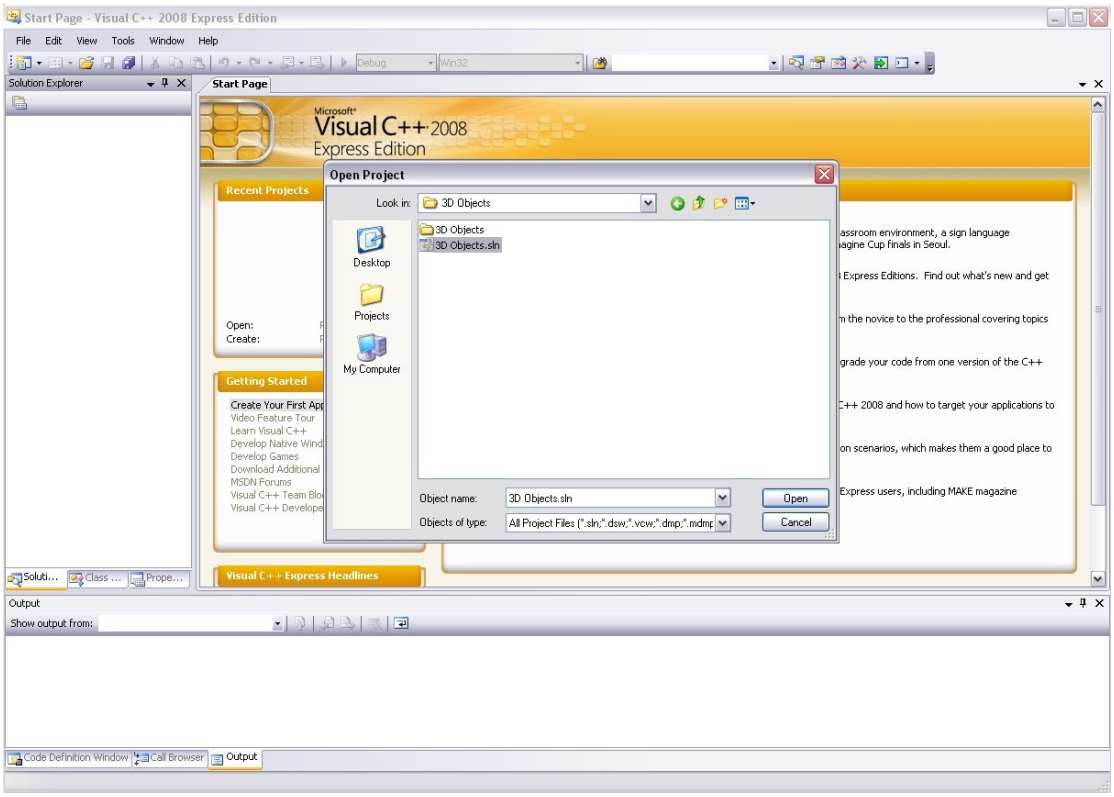
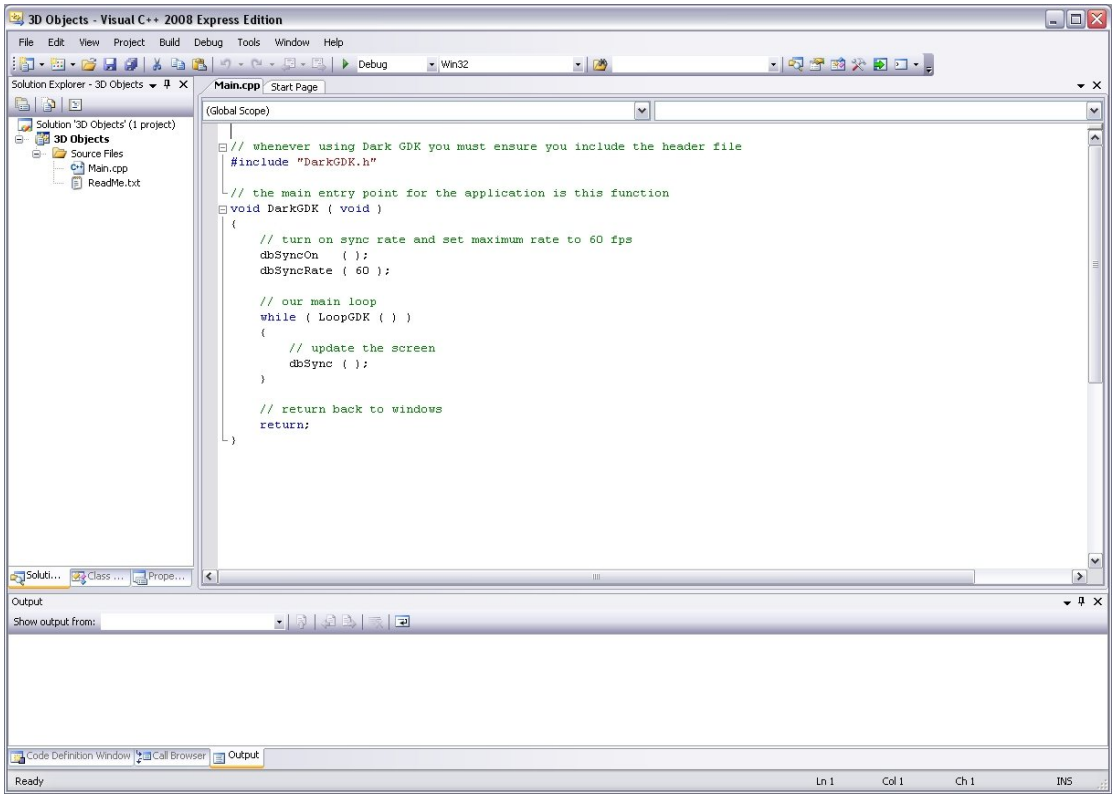
3D Objects

Introduction
In this tutorial we're going to learn how to load a 3D object and animate it.

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 3D Objects folder. Finally open the solution file for 3D Objects.



The project contains a source file named Main.cpp. This file contains the basic code required for a typical Dark GDK program.



Loading a 3D object

We are now going to load a 3D model into our program. The model we are using has been created in a 3D modeller and exported into the X format.

The file is called “Colonel-X.x” and it has been placed in the same directory as our project.

To load this object into Dark GDK we can call the dbLoadObject function. This function takes two parameters. The first parameter is the filename of the model and the second parameter is the ID number. We already know the filename and for an ID number we can use a value of 1 e.g.

```
dbLoadObject ( “Colonel-X.x”, 1 );
```

Add this new line before the programs main loop as follows:

```
#include "DarkGDK.h"

void DarkGDK ( void )
{
	dbSyncOn   ( );
	dbSyncRate ( 60 );

	dbLoadObject ( "Colonel-X.x", 1 );

	while ( LoopGDK ( ) )
	{
		dbSync ( );
	}

	return;
}
```

Before we continue it is necessary to add in one further change – the camera needs to be repositioned so we can view the object correctly. To position the camera we can call the function dbPositionCamera. This function takes three parameters defining the X, Y and Z coordinates in 3D space. To get a closer view of the model we need to move the camera on the Y and Z axis. Let’s try positioning the camera at 0, 50, -80:

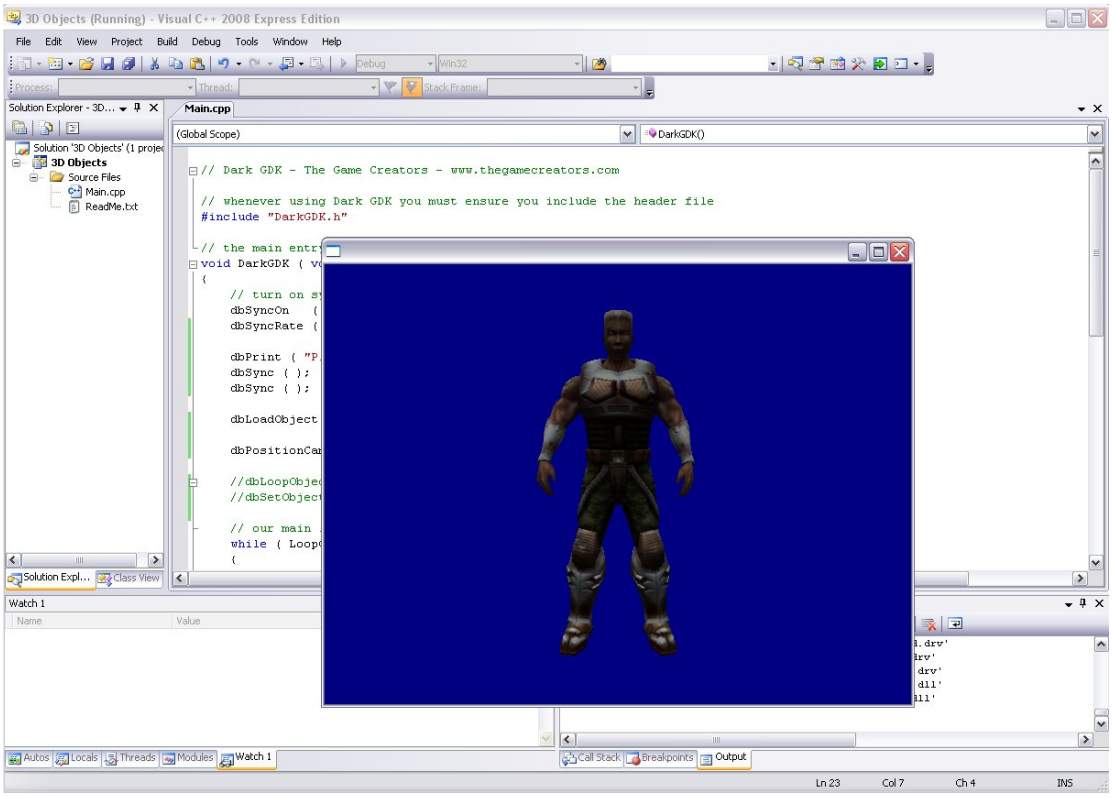
```
	dbPositionCamera ( 0, 50, -80 );
```

Add this line in directly before we enter the main loop:

```
#include "DarkGDK.h"

void DarkGDK ( void )
{
	dbSyncOn   ( );
	dbSyncRate ( 60 );

	dbLoadObject ( " Colonel-X.x ", 1 );

	dbPositionCamera ( 0, 50, -80 );

	while ( LoopGDK ( ) )
	{
		dbSync ( );
	}
```

```
        return;
}
```

Now go to the Debug menu and select Start Debugging:



The model has loaded and our camera has been repositioned so we can view it correctly.

Animating a 3D object

The model we have loaded contains a variety of animations. These animations can be played back by using a few simple functions in Dark GDK. One of these functions is named dbLoopObject. There are three variations of this function:

```
void dbLoopObject ( int iID );
void dbLoopObject ( int iID, int iStart );
void dbLoopObject ( int iID, int iStart, int iEnd );
```

The first function will take the specified object and playback all of its animation frames and then repeat them. The second function allows you to specify what frame to start from while the third function allows you to specify a start and end frame. For our example we will play through all the frames of animation that this model contains so we can simply call dbLoopObject and pass in our objects ID number e.g:

```
dbLoopObject ( 1 );
```

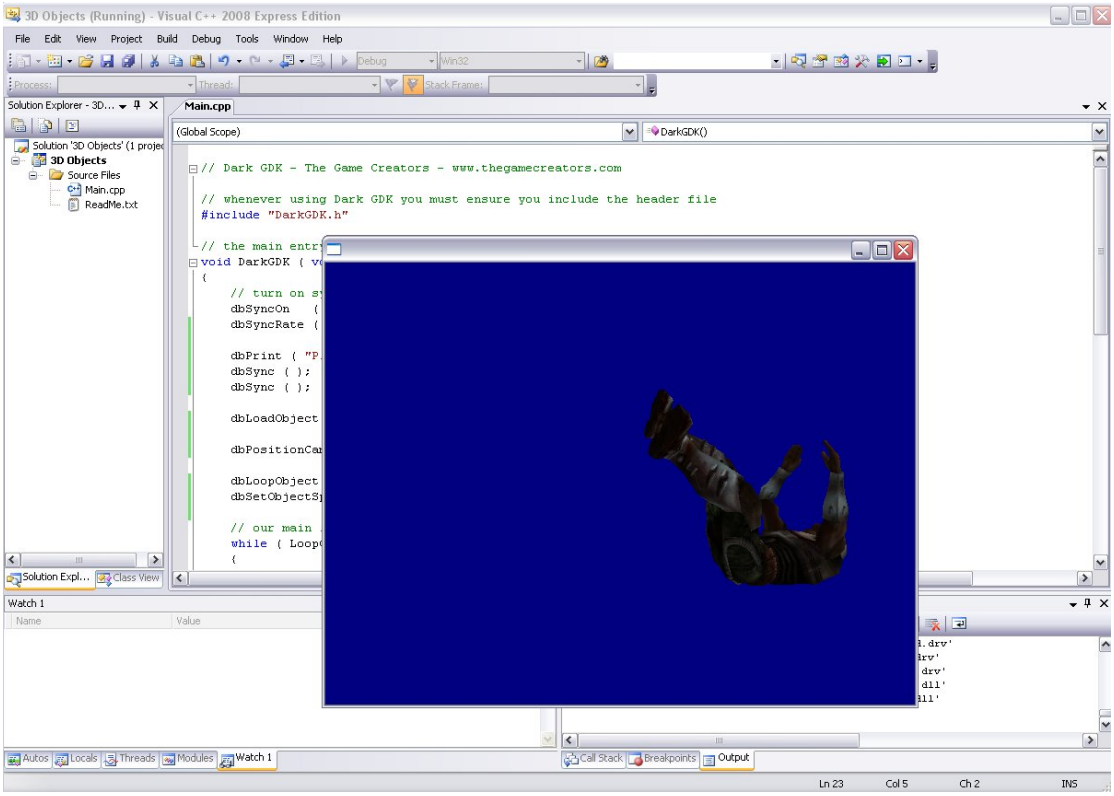
This line can be added in after we have positioned the camera. If you were to compile and run the program at this stage you would find the animation plays back, however it will do so very slowly. This is because the default speed that animation plays back at may vary from model to model. To change the rate at which animation plays back we can use the function dbSetObjectSpeed. This function takes two parameters. The first is an ID number for the object. The second is a value defining how fast the animation will playback. Lower numbers will result in slow playback while higher numbers will result in a faster playback. Our model needs quite a high playback speed so we can set the value to 40 e.g.:

```
	dbSetObjectSpeed ( 1, 40 );
```

Add this line in directly after the call to dbLoopObject:

```
#include "DarkGDK.h"

void DarkGDK ( void )
{
	dbSyncOn   ( );
	dbSyncRate ( 60 );

	dbLoadObject ( " Colonel-X.x ", 1 );

	dbPositionCamera ( 0, 50, -80 );

	dbLoopObject ( 1 );
	dbSetObjectSpeed ( 1, 40 );

	while ( LoopGDK ( ) )
	{
		dbSync ( );
	}

	return;
}
```

Now compile and run the program and sit back and watch as the model is run through all its animations:



Conclusion

With just a few lines of code we've been able to load a 3D model and animate it.

The 3D formats supported by the DarkGDK are: 3DS, X, DBO, MD2, MD3 and MDL.