Chapter 15

The HLDL Framework

ShiVa comes preloaded with several useful frameworks, which can remove some of the headaches from creating your own Games. As I said before, there are several extremely useful pre-built frameworks available for ShiVa. These frameworks consist of the FPVFramework (First Person View), the OSVFramework (Object Show View), the Car Dynamic Framework (which simulates a car), along with several others. There are also some other frameworks that can be downloaded from the ShiVa Developer Network (http://developer.stonetrip.com/Downloads/) and yet more frameworks that have been written by forum members, such as the Third Person Camera, and the FPS (First Person Shooter).

For the purposes of our Game I am going to use the HLDL framework, which can be downloaded from the ShiVa Developer Network. This is the point where we start to create our Dino Hunter Game, so hold on tight and let's go!

The HLDL framework contains several useful Models, and associated Scripts, and a brief overview of each one is given below:

HLDMain	This is the AIModel that basically runs the HLDL framework.
HLDVisitCamera	This is a simple first person walking Camera that uses Dynamics and Collisions.
HLDDummy	This is a dummy Object.
HLDFadeShapeSensorB	This is a Box version of a Sensor that is dedicated to making Shape Objects fade in, or out.
HLDFadeShapeSensorS	This is a Sphere version of HLDFadeShapeSensorB.
HLDFadeLightSensorB	This is a Box version of a Sensor that is dedicated to making dynamic Light Objects fade in, or out.
HLDFadeLightSensorS	This is a Sphere version of HLDFadeLightSensorB.
HLDSwitchCameraSensorB	This is a Box version of a Sensor that is dedicated to switching between Cameras.
HLDSwitchCameraSensorS	This is a Sphere version of HLDSwitchCameraSensorB.

HLDOpenURLSensorB	This is a Box version of a Sensor that is dedicated to opening a web browser page at a specified URL.
HLDOpenURLSensorS	This is a Sphere version of HLDOpenURLSensorB.
HLDPlayAnimationSensorB	This is a Box version of a Sensor that is dedicated to controlling animated Objects.
HLDPlayAnimationSensorS	This is a Sphere version of HLDPlayAnimationSensorB.
HLDPlaySoundSensorB	This is a Box version of a Sensor that is dedicated to playing a Sound.
HLDPlaySoundSensorS	This is a Sphere version of HLDPlaySoundSensorB.
HLDSwitchSceneSensorB	This is a Box version of a Sensor that is dedicated to switching between Scenes.
HLDSwithSceneSensorS	This is a Sphere version of HLDSwitchSceneSensorB.
HLDPlayMovieSensorB	This is a Box version of a Sensor that is dedicated to controlling the playback of a Movie.
HLDPlayMovieSensorS	This is a Sphere version of HLDPlayMovieSensorB.
HLDBlinkShapeSensorB	This is a Box version of a Sensor that is dedicated to making Shape Objects blink.
HLDBlinkShapeSensorS	This is a Sphere version of HLDBlinkShapeSensorB.
HLDMessageBoxSensorB	This is a Box version of a Sensor that is dedicated to displaying Message Boxes.
HLDMessageBoxSensorS	This is a Sphere version of HLDMessageBoxSensorB.

Phew! That was quite a long list, but it shows just how useful frameworks can be, as they can take a lot of the initial programming away, and let the Game Designer concentrate on making the Game.

This seems like a good place to reiterate that, no matter how simple your Game appears to be, it WILL involve a good deal of Scripting, and also a good deal of patience when debugging. I remember many years ago I programmed a very simple Game on a Commodore Pet computer and, even though there were no graphics as such and it was written in BASIC, it still took a couple of weeks to iron out all the bugs, and get the game play right! (from memory, there were something like 200-300 lines of code!). Later on I wrote my own version of Space Invaders on a Sinclair ZX81 computer using Z80A Machine Language, all in under 1KB of memory I might add!, and that took me weeks and weeks to debug and get right (well, Machine Language is not quite as easy as BASIC to understand!).

YOU HAVE BEEN WARNED!

I'm not deliberately trying to put you off, but I believe in letting people know what they are getting themselves into!

So, if you think you are still up to the challenge of creating a 3D Game read on…..

CREATING THE GAME

Creating our Game using the HLDL framework is pretty much the same as creating the app from the previous chapter, with the major difference being that we will use the HLDL framework instead of coding our own. Firstly, you will need to download the HLDL framework from the ShiVa Developer Network website, and save it somewhere you can find it.

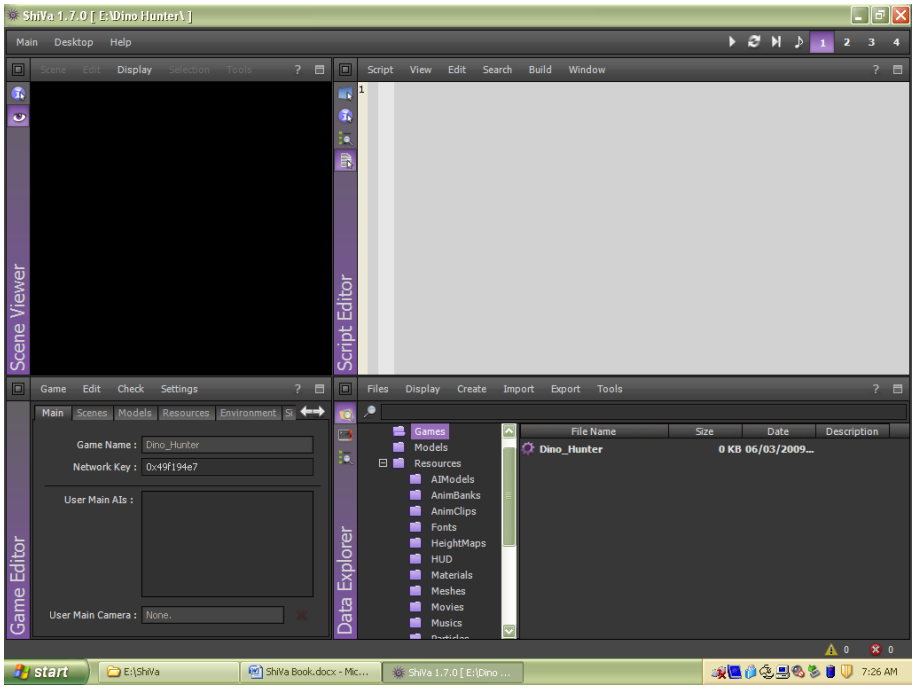
So, let's get started by creating our new Game. Firstly, click on "Main" and then "Settings". In the dialog that appears, click on the "Add" button, and set the base directory to point to where you want to store your new Game. Next, click on "OK", and then click on "Game", followed by "Create", in the Game Editor. Enter the name of our new Game ("Dino_Hunter"?) and click on "OK". You should now have your new Game ready for action in the Game Editor.

Now, open the Data Explorer module and click on "Import" and then "Archive". In the selection box, click on the ellipsis () and navigate to where you downloaded the HLDL framework. Select the ShiVa Archive (.ste) file titled "HLDLibrary-1.0.0", and click on "Open". Next, click on the "Import" button and ShiVa will display its "processing" dialog, and all being well, will import the framework.

Just to check, we should have the following files visible in the Data Explorer:

Games Folder	-	Dino_Hunter
Models Folder	-	All of the Models outlined above
AIModels Folder	-	One for each type of Model outlined above (12 in total)
Fonts Folder	-	DefaultFont & HLDArial
HUD Folder	-	HLDMessageBox
Scripts Folder	-	167 Script Files
Textures Folder	-	DefaultChecker, DefaultChecker2, DefaultChromicReflection, DefaultFont, DefaultSoftReflection & HLDArial

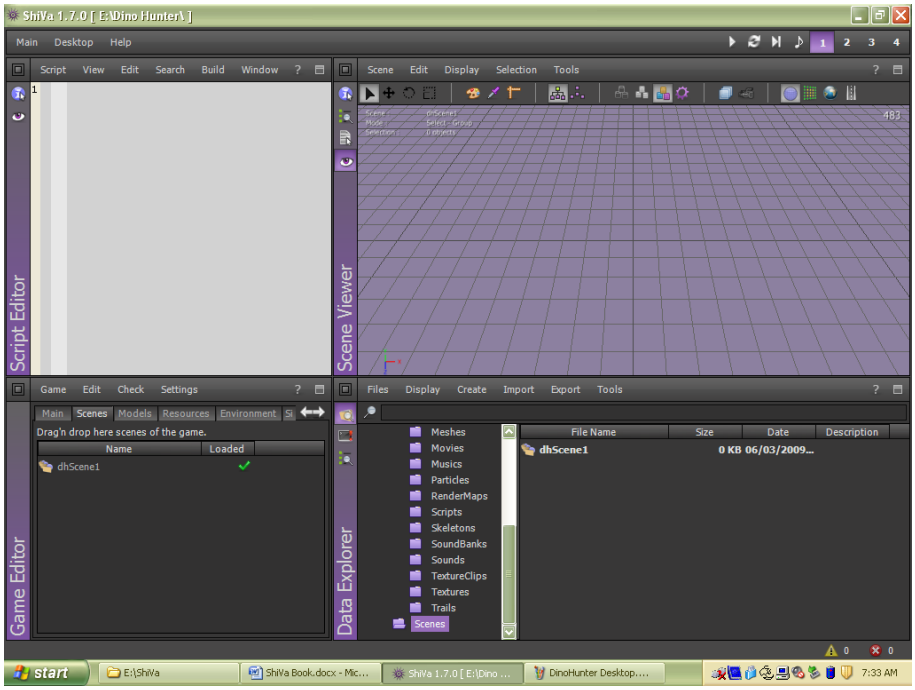
At this stage, your ShiVa desktop should look something like this:



CREATING OUR FIRST SCENE

Now we will create the basic Scene, it won't be anything too flashy, but I'm sure you'll be able to improve it substantially over time.

Open the Data Explorer module, click on "Create", then on "Scene" and enter the name for the Scene ("dhScene1"). Finally, click on "OK" and your new Scene is created. Next, we need to add the Scene to the Game, so drag the Scene from the "Scenes" folder of the Data Explorer to the "Scenes" tab in the Game Editor, and double-click on it in the Game Editor. Your desktop should now look like this:



All you need to do now is to add some Light, and save the Scene. So, to add the Light, drag the "DefaultDynamicLightSet" from the "Models" directory of the Data Explorer to the Scene, ensuring it is placed somewhere near the middle, and in the Scene Viewer module click on "Scene", then on "Save". That's it! The Scene is now saved.

IMPORTING THE REST OF THE GAME ITEMS

Now we will import all the other items required for our Game.

In the files you downloaded from the StoneTrip website you will find a Folder labelled "Dino Hunter". In this Folder and its sub-folders are all the other items that I have put together to get the basic Game up and running.

Models

The Models that you will need to import are:

Weapon0B
Weapon0Bullet
(Tree)
(Dinos)

AnimBanks

The AnimBanks that you will need to import are:

(Dinos)

AnimClips

The AnimClips that you will need to import are:

(Dinos)

Materials

The Materials that you will need to import are:

flingue_Material001
flingue_Material020
flingue_Material021
flingue_Material023
weapon0Bullet_Material001
(Tree)
(Dinos)

Meshes

The Meshes that you will need to import are:

flingue_Mesh000
flingue_Mesh001
flingue_Mesh002
weapon0Bullet_Mesh000
(Tree)
(Dinos)

Musics

The Musics that you will need to import are:

dramatic1
heavystuff
orchestral2
orchestral8
spookysynths

ALL of the above Music is courtesy of Psionic, and can be obtained from the following website:

www.psionic3d.co.uk

Please read the "Music Readme.txt" file for full details of licence granted in respect of these files.

Particles

weapon0Explosion

Skeletons

(Dinos)

Sounds

w0
(Others)

Textures

Snownm (soil_normals)
Soilnm (soil_normals)
Rocknm (soil_normals)
Waternm (soil_normals)
SoilColour (soil_color)
RockColour (4269_v1)
WaterColour (water)
Grass02 – used for the GRASS
Glow01 – used in Particle System
weapon0Bullet_Texture000 – EffectMap0 of weapon0Bullet_Material001
gun – EffectMap0 of flingue_Material020
gun2 – Normal Map of flingue_Material020
control – EffectMap0 of flingue_Material021
lavaDUDV – EffectMap0 of flingue_Material021
Txt_SphereMap_Garden2– EffectMap0 of flingue_Material023
(Dinos)
(Tree)

In the next Chapter, I'll explain how to create the Terrain that will be used in our Game.

IMPORTANT NOTICE!

Unless otherwise stated, all of the items used in this book have been sourced from official ShiVa demos, and may be used in your own applications subject to the following:

StoneTrip Source Application License (SSAL)
Version 1.0, October 23, 2008.

LICENSE

Terms and Conditions for Copying, Distributing, and Modifying

Items other than copying, distributing, and modifying the content with which this license was distributed (such as using, etc.) are outside the scope of this license.

1. You may copy and distribute the source application as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the source application a copy of this License along with the source application. You may not charge a fee for the sole service of providing access to and/or use of the source application via a network (e.g. the Internet), whether it be via the world wide web, FTP, or any other method.

2. You may modify your copy or copies of the source application or any portion of it, thus forming works based on the Content, and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You can only use source code, graphics assets stay the StoneTrip properties.

b) You can't publish an application similar to this one, name and concept stay the StoneTrip properties

3. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to copy, distribute or modify the source application. These actions are prohibited by law if you do not accept this License. Therefore, by distributing or translating the source application, or by deriving works here from, you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or translating the source application.

NO WARRANTY

4. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MIRROR AND/OR REDISTRIBUTE THE source application AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE source application, EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Please refer to the relevant documentation provided in the various Folders for any limitations on the use of any non-StoneTrip items.