Appendix B. GPGT Lesson Kit Docs

Table of Contents

B.1 GPGT Lesson Kit User's Guide

B.1.1. About The GPGT Lesson Kit

The Game Programmer's Guide to Torque (GPGT) was created with a variety of audiences in mind, including Lone-Wolf Developers, Teams, Instructors, and Students. The members of this broad audience have some unique and several shared requirements.

- **Lone-Wolf Developers** – This part of the audience faces the most challenges. The lone-wolf has to make all of his own content, purchase it from others, or contract help to build it. To help the individual who makes her own content, the kit includes samples of the majority of the engine's scripting feature, 2D classes (GUI controls), and 3D classes. With samples in hand, the lone-wolf developer has a leg up on learning and understanding the Torque Game Engine. See "Lesson Kit Features" below for the list of samplers included with this kit.
- **Team Developers** – This part of the audience faces many of the same challenges as the lone-wolf with the added challenge of collaboration. The kit is designed to facilitate collaborative work using scripts, 2D classes, and 3D classes. If each member of the team has a copy of the kit, it is simple to create sample works and pass them to other team members. They can then drop the work into their own kit (anywhere) and examine it. The kit automatically loads new script samples, 2D samples, and 3D samples which have been created using the rules outlined in "Adding New Lessons and Samples" below. You can think of the kit as a collaboration harness.
- **Instructors and Students** – Instructors and students require all of the features that the lone-wolf and team developers require. Additionally, they may require ways to record and play-back lessons, or to display samples to a distributed audience. The kit has these capabilities too. See Section B.2 "Instructing and Collaborating" below for more details.

This kit is designed to facilitate learning and collaboration through the hands on application and examination of all Torque Game Engine features. To do that, the kit includes these major tools:

Figure B.1.1. GuiMouseEventCtrl

GUI Sampler – This tool provides a harness for examining individual GUI controls in use. The sampler comes with 22 samples demonstrating the features and usage of the most common GUI controls. Additionally, a blank template is supplied that can be used to drop new samples in the kit. See Section B.1.6 to learn how to add new GUI samples.

Figure B.1.1 shows the GUIMouseEventCtrl sample which graphically demonstrates all of the features of this mouse-input capturing control.

Interface Sampler – This tool is used to demonstrate entire interfaces in action.

Creating an interface is a bit more complicated than using the individual GUIs.

Also, not having a specific native control for your particular application should not be a barrier to completion. It is possible to use multiple controls and scripts together to create new HUDs and interface elements.

Figure B.1.2 shows the tech version of the main menu interface supplied with this kit.

See Section B.1.7 to learn how to add new interface samples.

Figure B.1.2. Main Menu (Tech)

Figure B.1.3. 3D Lesson Selector

Sample Script Console – This tool is used to show sample scripts in a classroom setting. Samples may be typed in directly, or loaded from a file.

Figure B.1.4 shows this tool in use.

See Section B.1.5 to learn more about using this tool.

3D Lessons Mission – This tool is used to demonstrate the usage of individual 3D classes and other engine features, used alone or together.

A special mission has been created with a set of utility scripts and a harness to allow individuals and groups to share and examine any 3D content.

Figure B.1.3 shows the 3D lessons selector displaying a summary of the Transforms lesson.

In addition to the many included lessons, it is possible to add new lessons using the supplied template. See B.1.8 to learn how to add new 3D lessons.

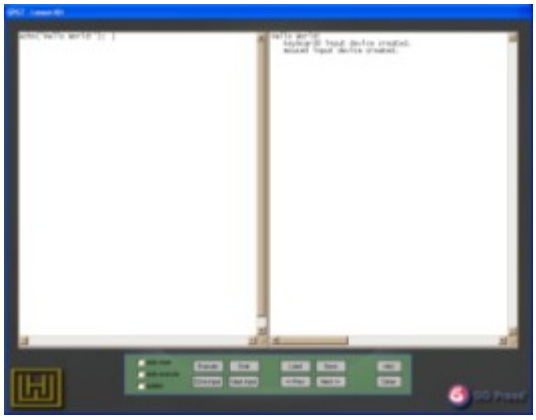


Figure B.1.4. Sample Script Console

B.1.2. Running The GUI Sampler

To run the existing GUI Samples, start the kit and follow these steps:

Start Sampler Tool

Click on the ***GUIs Sampler*** button.

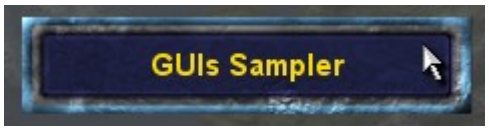


Figure B.1.5. *GUIs Sampler*

Back **Forward** **No Pages**

Figure B.1.6. Sampler navigation buttons

Navigate to Sample

Navigate to the sampler page containing the sample you wish to view using the navigation buttons.

Note: The back button is also tied to the **ESCAPE** key and the forward button is tied to the **SPACE** key.

Run Sample

Click on the button containing the name of the GUI control you wish to see a sample of.

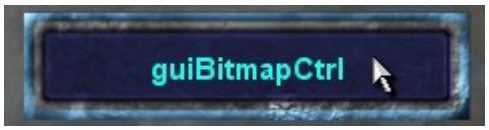


Figure B.1.7. *guiBitmapCtrl*

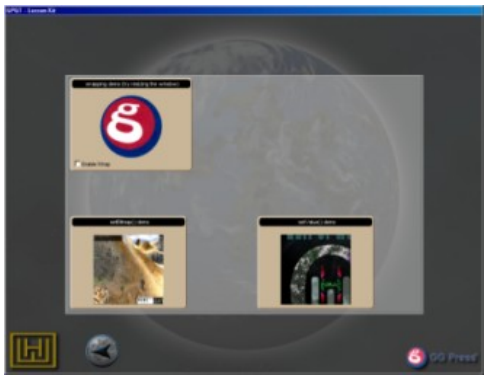


Figure B.1.8. guiBitmapCtrl Sample

B.1.3. Running The Interface Sampler

To run the existing Interface Samples, start the kit and follow these steps:

Start Sampler Tool

Click on the ***Interface Sampler*** button.

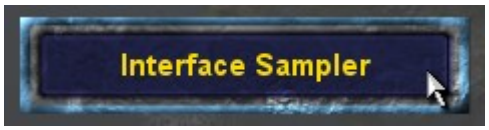


Figure B.1.9. *Interface Sampler*

Back **Forward** **No Pages**

Figure B.1.10. Sampler navigation buttons

Navigate to Sample

Navigate to the sampler page containing the sample you wish to view using the navigation buttons.

Note: The back button is also tied to the **ESCAPE** key and the forward button is tied to the **SPACE** key.

Run Sample

Click on the button containing the name of the GUI control you wish to see a sample of.

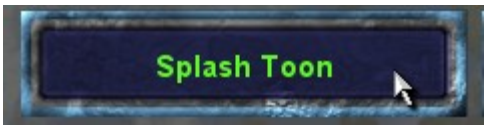


Figure B.1.11. Splash Screen

Figure B.1.12. Toon Splash Sample

B.1.4. Running The 3D Lessons (SinglePlayer)

To run the existing 3D Lessons, start the kit and follow these steps:

Load 3D Lessons Mission

Click on the ***Start Mission...*** button.

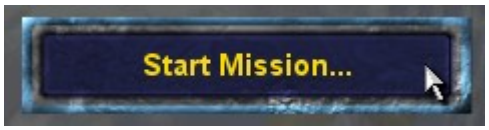


Figure B.1.13. *Start Mission...*

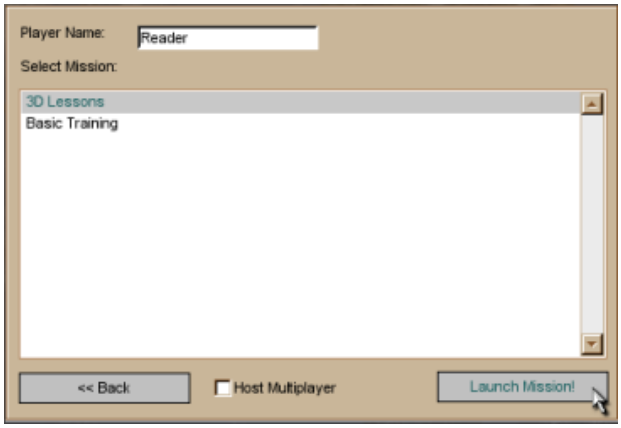


Figure B.1.14. Missions Dialog

Figure B.1.15. Loading the mission...

Launch 3D Lessons Mission

Select **3D Lessons** from the missions list and click the **Launch Mission!** button as in Figure B.1.14.

TGE will load the mission, load datablocks, do some lighting calculations and eventually pop you into the mission. See Figure B.1.15.

When you are dropped into the mission, you will find yourself in an empty grassy cauldron (Figure B.1.16).

Figure B.1.16. In the *3D Lessons* mission!

Open Lesson Selector Dialog

PC Users: Click on the right-mouse button to start the lesson-selector dialog (Figure B.1.17).

Macintosh Users: Click EFM + the mouse to start the lessons-selector dialog (Figure B.1.17).

Figure B.1.17. Lesson Selector Dialog

Navigate To Lesson

You will find that this dialog has several parts related to navigating and selecting a lesson for loading:

- **Lesson Tiles (Figure B.1.18. A)** – These pictures and/or text tiles show the currently selected lesson.
- **Lesson Description Pane (Figure B.1.18. B)** – This pane displays the title of and a summary of the currently selected lesson.
- **Lesson Navigation Buttons (Figure B.1.19.)** – These buttons are used to scroll up and down through the current lesson group(s). They navigate as follows: A – Up 3 lessons; B – Up 1 lesson; C – Down 3 lessons; D – Down 1 lesson
- **Lesson Group Buttons (Figure B.1.20.)** – These buttons select which lessons are included in the lesson tiles scroll group. They select the following lesson groups: A – Volume 1 lessons; B – Volume 2 lessons; C – Community created lessons; D – Q & A solution lessons.

Use these buttons to find and lesson to run, then move on to the next step.

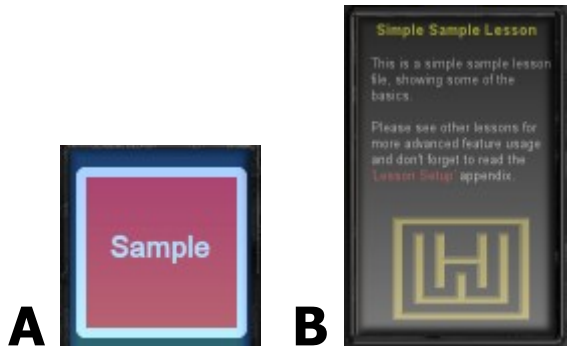


Figure B.1.18. Lesson Tile and Description Pane

Figure B.1.19. Lesson navigation buttons

Figure B.1.20. Lesson group buttons

Figure B.1.21. Lesson selection buttons

Run Lesson

Running a lesson is as simple as finding the one you want (last step) and then clicking the ***Select*** button (Figure B.1.21. A). Alternately, you can just double-click the lesson tile for the lesson you want to run.

If you decide not to run a lesson after all, simply click the ***Cancel*** button (Figure B.1.21.B).

Individual Lessons

Some, but not all lessons will first pop up a dialog box. Be sure to read the contents of this box as it will usually supply additional instructions regarding how the lesson works and how to use it.

Additionally, almost all lessons have these common parts, including:

- **Lesson Compass (Figure B.1.22.)** – The lesson compass will tell you in which direction your avatar is facing at any time during the lesson.
- **Metrics Dialog (Figure B.1.23.)** – This dialog will present the following information (based on what your player is looking at):
  - **FPS**
  - **Client**
    - **ID** – Numeric ID of your client.
  - **Player**
    - **ID** – Numeric ID of player.
    - **DB** – Datablock used by player.
    - **Cur Energy** – Energy points of player.
    - **Max Energy** – Maximum energy points of player.
    - **Damage** – Percentage of maximum damage this player has taken.
  - **Camera**
    - **ID** – Numeric ID of camera.
    - **DB** – Datablock used by camera.

Figure B.1.22. Lesson compass

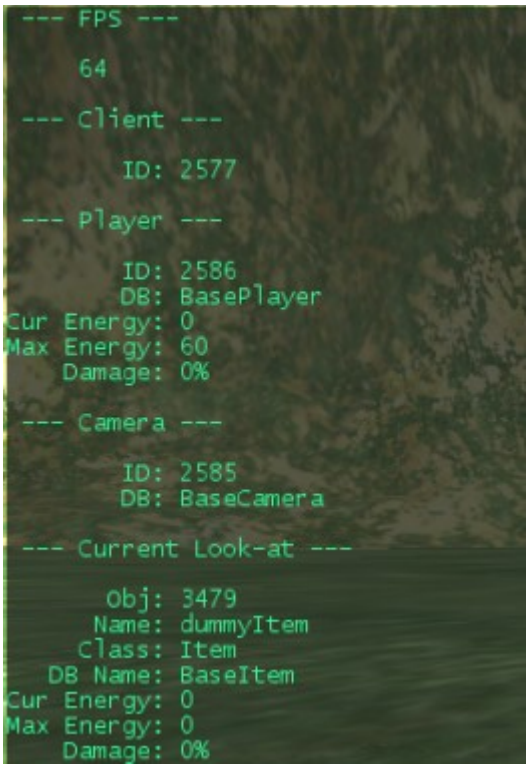


Figure B.1.23. Metrics Dialog

Individual Lessons (cont'd)

- **Current Look-at** (Object under reticle)
  - Obj – Numeric server-side ID of current look-at object.
  - Name – Alphanumeric name of current look-at object.
  - Class – Class name of current look-at object
  - DB Name – Datablock used by current look-at object.
  - **Cur Energy** – Energy points of current look-at object.
  - **Max Energy** – Maximum energy points of current look-at object.
  - **Damage** – Percentage of maximum damage current look-at object has taken.
- **Compass Signs and Position Markers (Figure B.1.24.)** – Lastly, most lessons will have compass point signs North, South, East, and West, as well as position markers along the major axes, marking the 10 meter through 70 meter positions. These are all used to help you correlate lesson parts to that lessons documentation.

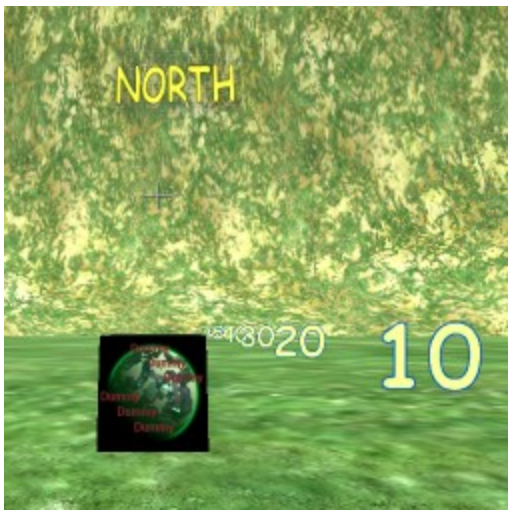


Figure B.1.24. Compass Signs and Markers

Additional instructions are provided later in this appendix for running the lesson kit in multiplayer mode. This mode is provided to allow an instructor/lecturer or collaborators to run 3D lessons such that one individual drives the 3D lesson while others participate as observers. Please see Section ""B.2 Instructing and Collaborating" below.

B.1.5. Running The Sample Script Console

To run the existing GUI Samples, start the kit and follow these steps:

Start Sample Script Console

Click on the ***Sample Script Console*** button.

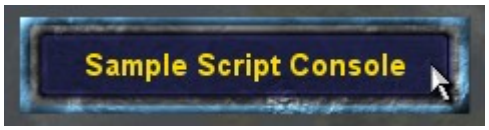


Figure B.1.25. *Sample Script Console*

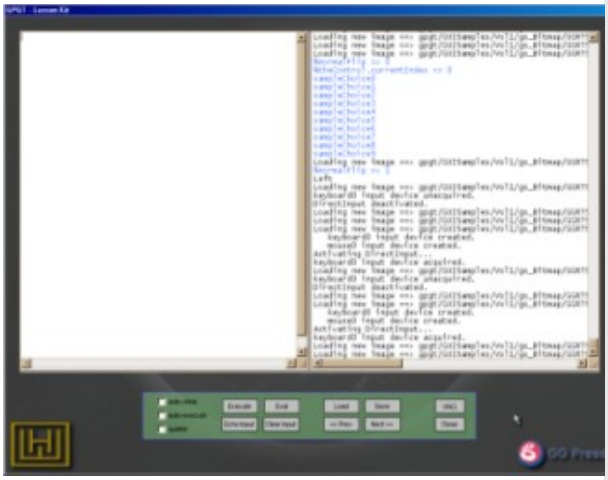


Figure B.1.26. Ready to script

Manually Type Scripts

At this point, you may manually type scripts into the left pane and then evaluate them by pressing the ***Eval*** button (Figure B.1.27). The result of the execution will be shown in the right pane.

Figure B.1.27. Eval button

Run Scripts From File

Alternatively, if you are running the kit in a classroom or lecture setting you may want to prepare scripts in advance and then run them for your class.

Simply click on the Load button (Figure B.1.28) and then using the file selection dialog to select a script for running (Figure B.1.29).

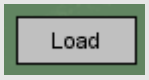


Figure B.1.28. Load button

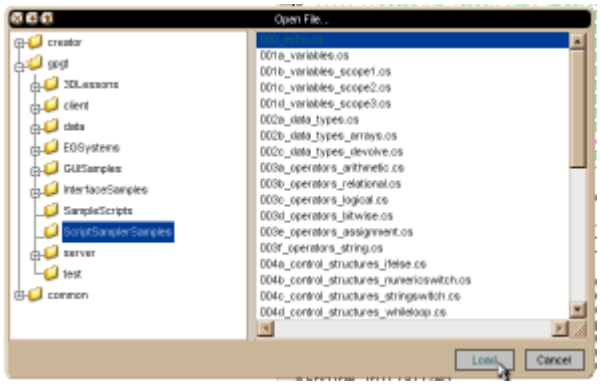


Figure B.1.29. File selection dialog

Sample Script Console Dashboard

The Sample Script Console tool provides a dashboard with several buttons and check boxes.

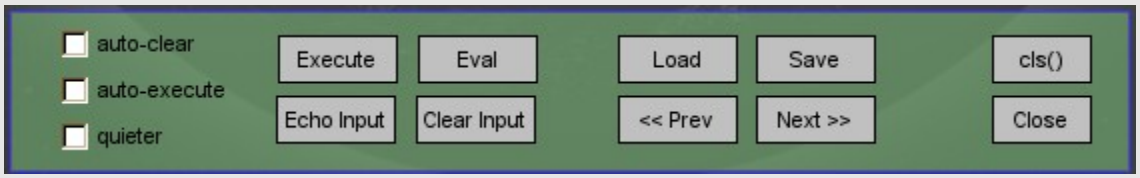


Figure B.1.30. The Sample Script Console Dashboard

- **auto-clear check box** – If this is checked, the output pane will be automatically cleared on each press of ***Execute*** or ***Eval***.
- **auto-execute check box** – If checked, every time a file is loaded it will automatically be executed.
- **quieter check box** – If checked, this will turn off most warnings and messages that appear in the output pane, showing only warnings and output from your script/sample instead.
- **Execute Button** – Pressing this button saves the contents of the input pane to a temporary file and then calls exec() on it. The output of this action is directed to the output pane.
  - **Warning:** This is designed for multi-line entries and entries with comments. This is the preferred method of executing sample scripts.
- **Eval Button** – Pressing this button passes the contents of the input pane to the eval() statement. The output of this action is directed to the output pane.
  - **Warning:** This is designed for single line entries or entries with no comments. Pressing this button when the input pane contains comments may not behave as expected.
- **Echo Input** – Pressing this button will echo the contents of the input pane in the output pane.
- **Clear Input** – Pressing this button will clear the input pane.
- **Load Button** – Pressing this button will start the load file dialog allowing you to load a previously written script.
- **Save Button** – Pressing this button will start the load save dialog allowing you to save the contents of the input pane to a file. Please note, that scripts saves in this manner will not be easily editable in an external editor.
- **<< Prev Button** – Pressing this button loads the previous file found in the last directory files were loaded from using the Load button.
- **>> Next Button** – Pressing this button loads the next file found in the last directory files were loaded from using the Load button.
- **cls() Button** – Pressing this button clears the output pane.
- **Close button** – Pressing this button closes the ***Sample Script Console*** tool.

B.1.6. Adding New GUI Samples

In the future, you may wish to create a GUI Sample using a new GUI control. Assuming your new GUI control is named ***guiMyControl***, you can create a sample for it by following these steps:

1. In the directory "~\gpgt\GUISamples" you will find a directory named "Template". Copy this directory and then rename the copy as "gs_MyControl".
2. Open your new directory "~\gpgt\GUISamples\gs_MyControl". In this directory you will find two files, "Template.cs" and "Template.gui". Rename these two files "gsMyControl.cs" and "gsMyControl.gui" respectively.
3. Open the newly renamed file "gsMyControl.cs" and globally replace the word "Template" with "MyControl". Save and close the file.
4. Open the newly renamed file "gsMyControl.gui" and globally replace the word "Template" with "MyControl". Save and close the file.

At this point, if you have correctly followed the above steps, you will be able to start the lesson kit and find your new GUI Sample button (Figure B.1.31) by running the ***GUI Sampler*** tool.

Figure B.1.31. MyControl GUI Button

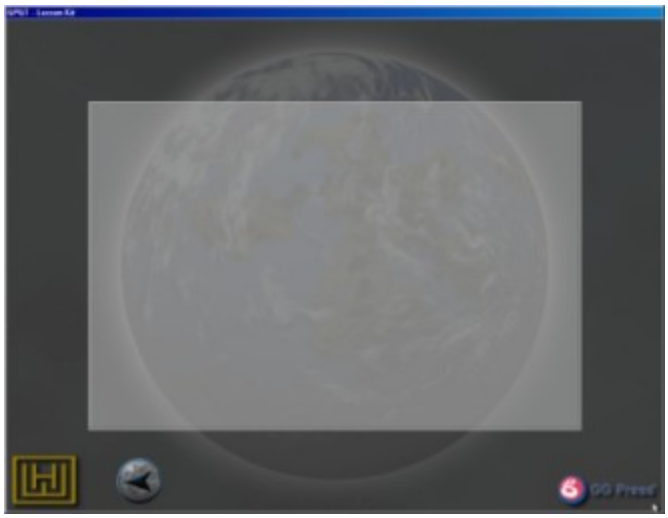


Figure B.1.32. MyControl Sample (starts blank)

Clicking on this new button will start your sample, which is currently blank (Figure B.1.32).

Now, you may edit this GUI to your hearts delight and simply save the edits in the .gui file we just created. Please note, the .cs file we just created is for loading the .gui file and for any scripts or profiles that you might need for your sample. Please see the provided samples for reference.

B.1.7. Adding New Interface Samples

In the future, you may wish to create a Interface Sample. Assuming your new intefaace is named named ***myInterface***, you can create a sample for it by following these steps:

1. In the directory "~\gpgt\InterfaceSamples" you will find a directory named "Template". Copy this directory and then rename the copy as "ifcs_myInterface".
2. Open your new directory "~\gpgt\InterfaceSamples\ifcs_myInterface". In this directory you will find two files, "Template.cs" and "Template.gui". Rename these two files "ifcsmyInterface.cs" and "ifcsmyInterface.gui" respectively.
3. Open the newly renamed file "ifcsmyInterface.cs" and globally replace the word "Template" with "myInterface". Save and close the file.
4. Open the newly renamed file "ifcsmyInterface.gui" and globally replace the word "Template" with "myInterface". Save and close the file.

At this point, if you have correctly followed the above steps, you will be able to start the lesson kit and find your new Interface Sample button (Figure B.1.33) by running the ***Interface Sampler*** tool.

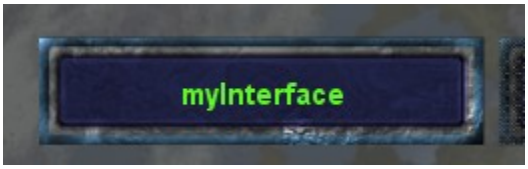


Figure B.1.33. myInterface Interface Button

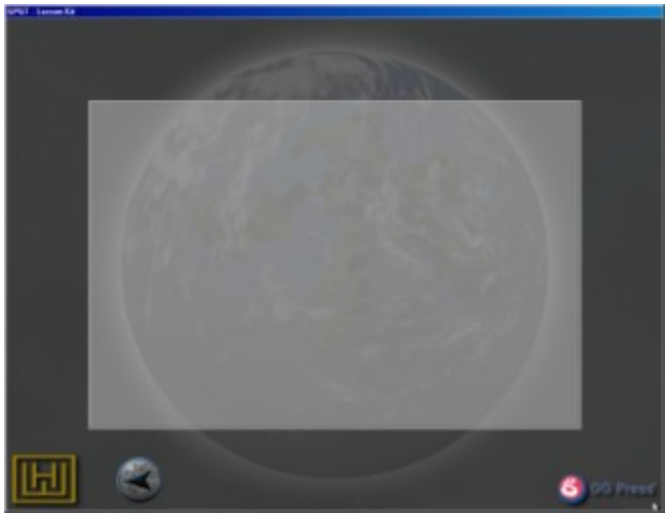


Figure B.1.34. myInterface Sample (starts blank)

Clicking on this new button will start your sample, which is currently blank (Figure B.1.34).

Now, you may edit this interface to your hearts delight and simply save the edits in the .gui file we just created. Please note, the .cs file we just created is for loading the .gui file and for any scripts or profiles that you might need for your sample. Please see the provided samples for reference.

B.1.8. Adding New 3D Lessons

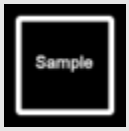
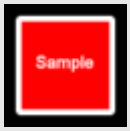
In the future, you may wish to create your own 3D lessons either to demo an idea, as a teaching tool, for for team collaboration and debug work. As with the ***GUI Sampler*** and the ***Interface Sample***, a template is provided to simply this task. Because this is the most complicated kind of lesson/sampler to add, lets take some time to examine the components of a 3D lesson before walking through the steps to build one.

B.1.8.1. 3D Lesson Components

New 3D lessons are created by providing a set of required and optional files (Table 1.). These files follow a specific format. Within these files, we define the contents of our lesson and how it will interact with the lesson environment. The sample lesson files can be found in directory:

"~\gpgt\3DLessons\Volume1\000_SimpleSampleLesson".

Table 1. 3D Lesson Template Files

Filename	Purpose
lesson.map.txt (required)	This is the **lesson map**. This file tells the lesson kit what other files are present in this directory and defines their purpose.
SimpleSampleLeson.cs (required)	This is the **lesson file**. In this file, we define all of the loading and unloading methods of a file.
sample.png **sample_sel.png** **(optional)**	The deselected and selected **lesson tiles** for this lesson. Each measures 210 x 210 pixels.  **deselected**  **selected**
SimpleSampleLesson.ml.txt (required)	The ML formated **lesson summary** file that will be displayed in the description pane of the lesson selector when the tile for this lesson is selected.
clientScripts.cs (optional)	This **client script** file contains any script(s) that should be run on the client, when the lesson is loaded. This can include such things as actionMap definitions, GUI definitions and scripts, etc.

The Lesson Map File

It is the job of the lesson map (lesson.map.txt) to tell the 3D lesson manager what files are present and should be loaded for the current lesson. The general layout of a lesson map is shown in Table 2.

Table 2. Lesson Map File Layout

Line	Name	Description
0	Lesson Filename **(required)**	The filename given to the **lesson file**.
1	Lesson Name **(required)**	A unique one-line description of the lesson 20 characters or fewer in length.
2	Lesson-Object Name **(required)**	This is the name of the scriptObject that defines the lesson.
3	Lesson Summary Filename **(required)**	This contains the name of the **lesson summary** ML file.
4	Lesson Tile Filename Root **(line may be blank)**	This is the root for the **lesson tile** filename. For example for the sample tiles, this string would be 'sample'. If this line is blank, the 'Lesson Name' will be printed on the Lesson Selector Button.
5	Client Script Filename **(line may be blank)**	This is the filename for the **client script** file. If present, this file is executed on every client loading this less.

In our sample lesson (template) we can find a lesson map:

"~\gpgt\3DLessons\Volume1\000_SimpleSampleLesson\lesson.map.txt".

This file contains the following text:

```
SimpleSampleLesson.cs
Simple Sample Lesson
SimpleSampleLesson
SimpleSampleLesson.ml.txt
sample
clientScripts.cs
```

The Lesson File

It is the job of the **lesson file** to define any datablock, functions, and methods used by the lesson. Additionally, it must define set of console methods in the namespace of the lesson scriptObject (as it was named in the **lesson map**).

The order of this file is optional, but is usually organized as follows:

- **Load/Define Datablocks** – Load, using exec(), and/or define any datablocks used in the lesson.
- **Load/Define Functions and Methods** – Load, using exec(), and/or define any functions and methods required for the lesson, excluding those discussed next.
- Define Required Lesson Methods – Define the three required methods, in the namespace of this lesson's, lesson scriptObject.
  - LessonName::**onAdd()** - It is the job of this method to do any lesson initialization, prior to execution.
  - LessonName::**onRemove()** - It is the job of this method to do any special lesson clean-up, not already handled by the lesson manager.
  - LessonName::**ExecuteLesson()** - It is the job of this method to build and run the lesson.

Here is an (edited) example of the **lesson file**:

"~\gpgt\3DLessons\Volume1\000_SimpleSampleLesson\SimpleSampleLesson.cs".

```
//--------------------------------------------------------
// ****** LOAD/DEFINE DATABLOCKS FOR LESSON
//--------------------------------------------------------
//None.

//--------------------------------------------------------
// ****** LOAD/DEFINE FUNCTIONS and METHODS FOR LESSON
//--------------------------------------------------------
function dummyItem::onPickup( ... )
{
   // ...
}

function dummyItem::onCollision( ... )
{
   // ...
}

function serverCmdbumpBlock( ... )
{
   // ...
}
```

```
//--------------------------------------------------------
// ****** DEFINE THE *REQUIRED* LESSON METHODS
//--------------------------------------------------------

function SimpleSampleLesson::onAdd( ... )
{
   // ...
}

function SimpleSampleLesson::onRemove( ... )
{
   // ...
}

function SimpleSampleLesson::ExecuteLesson( ... )
{
   // ...
}
```

DefaultLessonPrep()

Generally, the only job the onAdd() method has is to call the lesson manager supplied function DefaultLessonPrep():

```
function SimpleSampleLesson::onAdd( ... )
{
   DefaultLessonPrep();
}
```

It is the job of the DefaultlessonPrep() function to prepare the lesson area for your lesson to build in. Among the things it does are:

1. Creates a group called objectMarkers, containing 28 markers from which to drop objects.
2. Creates billboards on the ground below each of these markers.
3. Creates signs on the hills of the lesson area, marking N, S, E, and West.

A blank lesson, when loaded, might look something liked Figure B.1.35, which shows lesson marker billboards (10, 20, 30, ...), a sign (North), and the other screen elements we have already discussed, the Lesson Compass and the Metrics Dialog. The markers and signs are placed by DefaultLessonPrep().

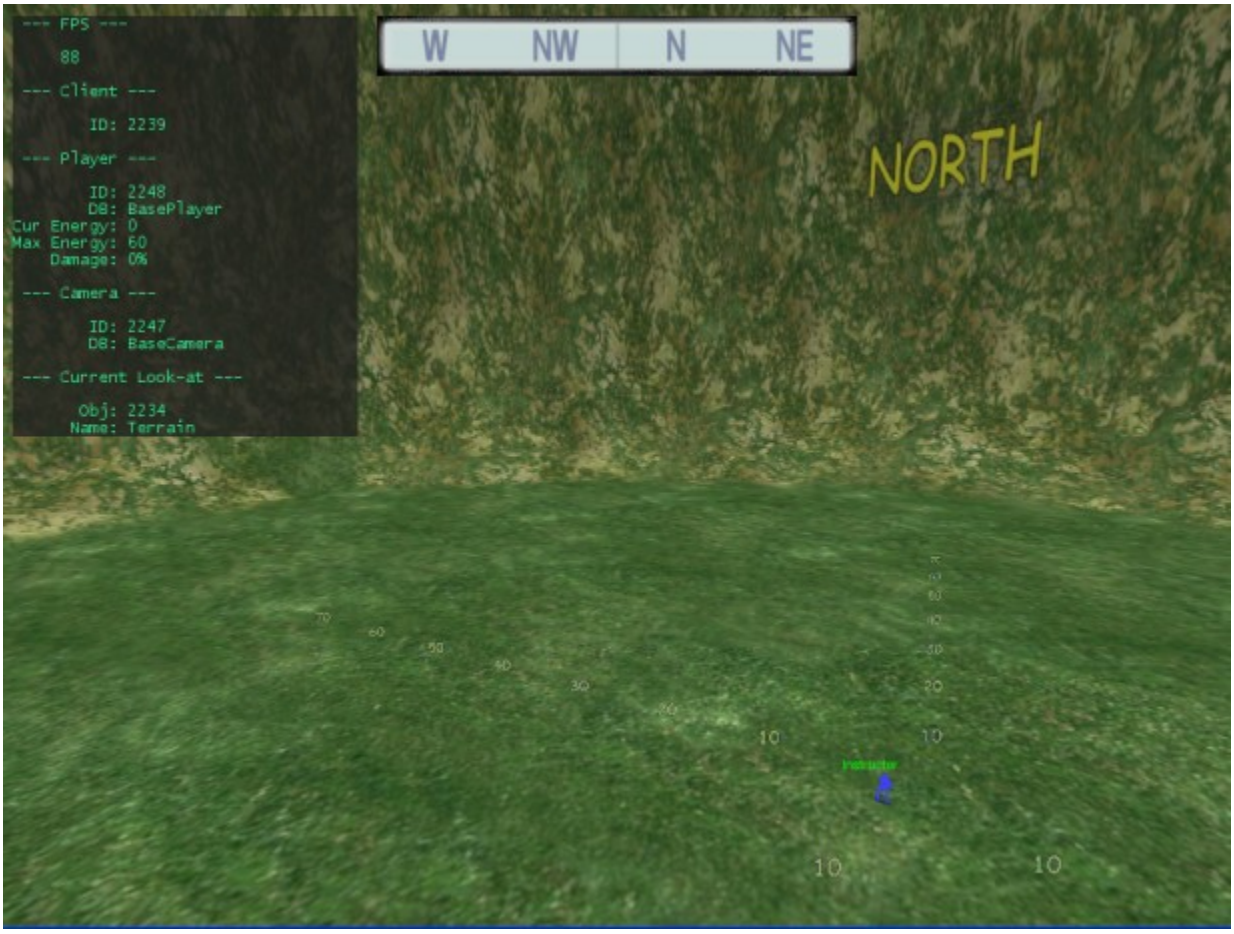


Figure B.1.35. Lesson w/ Marker Billboards and Signs

DefaultLessonPrep() Options

As was just mentioned, when DefaultLessonPrep() is called, the lesson manager will create marker billboards and North-South-East-West signs on the cauldron's sides. Not visible are drop points position above each of the billboards.

The purpose of these drop points is to allow one to drop objects from a known position without having to specify coordinates.

In TGE, the generally accepted compass points and their associated direction vectors are:

North	< 0 1 0 >	**East**	< 1 0 0 >
South	< 0 – 1 0 >	**West**	< -1 0 0 >

Furthermore, the drop points themselves are named and located thus:

Name	Position	Name	Position
North10	< 0 10 25 >	**East10**	< 10 0 25 >
North20	< 0 20 25 >	**East20**	< 20 0 25 >
North30	< 0 30 25 >	**East30**	< 30 0 25 >

Name	Position
North40	< 0 40 25 >
North50	< 0 50 25 >
North60	< 0 60 25 >
North70	< 0 70 25 >
South10	< 0 -10 25 >
South20	< 0 -20 25 >
South30	< 0 -30 25 >
South40	< 0 -40 25 >
South50	< 0 -50 25 >
South60	< 0 -60 25 >
South70	< 0 -70 25 >

Name	Position
East40	< 40 0 25 >
East50	< 50 0 25 >
East60	< 60 0 25 >
East70	< 70 0 25 >
West10	< -10 0 25 >
West20	< -20 0 25 >
West30	< -30 0 25 >
West40	< -40 0 25 >
West50	< -50 0 25 >
West60	< -60 0 25 >
West70	< -70 0 25 >

In addition to these 28 named drop points, the mission specifies a central drop point named "CentralDropPoint".

It should be noted that the billboards below the drop points are actual offset slightly. This way objects may be dropped directly below the drop point and not be on top of the drop point's marker billboard.

Turning Off Billboards or Signs

In some lessons, you may not want the DefaultLessonPrep() method to place either signs or billboard markers. Therefore, there are three four ways to call the function.

The first way, disables the North-South-East-West signs on the hills:

```
// No signs
DefaultLessonPrep( $GPGT::NOSIGNS );
```

The second way disables the billboard markers on the ground.

```
// No billboards
DefaultLessonPrep( $GPGT::NOMARKERS );
```

The third way disables both signs and billboard markers.

```
// No signs or billboards
DefaultLessonPrep( $GPGT::NOMARKERS | $GPGT::NOSIGNS );
```

The fourth and final way enables both signs and billboard markers.

```
// Signs or billboards enabled
DefaultLessonPrep(  );
```

Using Drop Points Manually

In order to use a drop point for placing an object, you may refer to it by name and get its position like this:

```
%dummyBlock = new Item( dummyItem )
{
   dataBlock = "BaseItem";
   position  = North10.getPosition();
   scale     = "2 2 2";
};
```

Alternately, you may use one of three supplied functions to calculate a position and/or execute the object drop.

Drop Point Functions

The Lesson Manager provides three 'drop functions' for placing newly created objects in a lesson:

DropObject(%object , %offsetVector)

```
This function will drop an object to the ground, or on top of the
first object found with a valid collision box below the dropping
object.

The dropping object starts its 'drop' at the position it was
created at and drop from there.

Note 1: To calculate a starting position, use
CalculateObjectDropPosition().

Note 2: This function will not move TSStatic() objects.  Their
positions must be set once and only once, upon construction.  Use
CalculateObjectDropPosition() instead.
```

DropObjectFromMarker(%object , %marker , %offsetVector)

```
This function drops an object from one of the markers, using the
same rules as DropObject().
```

CalculateObjectDropPosition(%oldPosition , %offsetVector)

This function will calculated a drop point without moving the object and returns the calculated drop position.

Arguments and their Defintions (For all above functions):	
%object	The object to be dropped.
%marker	A valid marker to be used as a starting point.
%offsetVector	An offset to adjust the drop by. Final positions will be: "initially calculated drop" + %offsetVector

The Lesson Summary ML File

Every lesson requires a description. That description is placed in the Lesson Summary ML File. A lesson summary should be short, just providing a title for the lesson and a summary of what it does. For example, this is the Lesson Summary for the Simple Sample Lesson (template):

```
<just:center><font:Arial Bold:18>
<spush><color:FFFF00>Simple Sample Lesson
<spop>
<font:Arial:16><br><br>
<just:left>
<lmargin%:5>
This is a simple sample lesson file, showing some of the
basics.<br><br>
Please see other lessons for more advanced feature usage and
don't forget to read the <spush><color:FF2222>'Appendix B -
GPGT Lesson Kit'<spop>.
<br>
<br>
<br>
<just:center>
<bitmap:gpgt/3DLessons/Volume1/000_SimpleSampleLesson/how>
```

If you need to provide extra information about the lesson, either write a document or use the Client Script File to populate and launch the "Lesson Message Dialog".

The Client Script File

The Client Script files is responsible for doing key mappings for a lesson and for populating and displaying the "Lesson Message Dialog", if it is needed.

The Simple Sample Lesson (Template) defines the following Client Script file (edited for legibility):

```
// 1 - Define any functions we need
function bumpIt( %val )
{
   if( !%val ) return;

   commandToServer('bumpBlock');
}

// 2 - Create and define the new action map
new ActionMap( lessonActionMap );

lessonActionMap.bind( keyboard , "b" , "bumpIt" );

lessonActionMap.bindCmd( keyboard, "h", "",
   "canvas.pushDialog( LessonMessage , 100 );");

lessonActionMap.push();

// 3 - Clear, populate, and push 'Lesson Message Dialog'
LessonMessageText.setValue("<font:Arial Bold:14>");

LessonMessageText.addText( ... );

// ...

canvas.pushDialog( LessonMessage , 100 );
```

If we examine this code, we will see that it does the following:

1. It defines a function which will later be called as the result of a key press. This function does nothing more than send a command to the server.
2. It creates, populates, and enables an action map named "**lessonActionMap**". Be aware, you must **ALWAYS** name the lesson action map "**lessonActionMap**", otherwise the lesson manager will no know how to unload it later.
3. It clears the last message from the 'lesson message dialog', adds a new message(s), and then pushes it onto the canvas.

B.1.8.2. Step-by-Step Creation of New 3D Lesson

Step#1 – Duplicate The Template

As your first step in making a new 3D lesson, please make a copy of the directory:

"~\gpgt\3DLessons\Volume1\000_SimpleSampleLesson"

, renaming it as:

"~\gpgt\3DLessons\Volume1**My3DLesson**"[1]

Then, in our new directory, rename

- "SimpleSampleLesson.cs" as "***My3DLesson*** .cs",
- "SimpleSampleLesson.ml.txt" as "***My3DLesson***.ml.txt",
- "sample.png" as "**my3dlesson**.png", and
- "sample_sel.png" as "**my3dlesson**_sel.png".

Step #2 – Update The Lesson Map

Continuing with the creation of your new 3D Lesson, modify the contents of the lesson map to look like this:

```
My3DLesson.cs
My 3D Lesson
My3DLesson
My3DLesson.ml.txt

clientScripts.cs
```

In this implementation, we are specifying that:

- the **lesson file** is named "**My3DLesson**.cs",
- the arbitrary name of the lesson is "My 3D Lesson",
- the scriptObject for your lesson will be named "**My3DLesson**",
- the **lesson summary** file will be named "**My3DLesson**.ml.txt",
- no lesson tile is specified (a blank tile will be supplied and the lesson name "My 3D Lesson" will be printed on it), and
- the **client script** file will be named "clientScripts.cs".

[1] In the future you may rename the lesson however you like, but for this first attempt, we will assume the lesson is named "**My3DLesson**".

Step #3 – Modify The Lesson File

Because this is just a sample, we will merely open the lesson file "**My3DLesson**.cs" and rename every instance of "SimpleSampleLesson" as "My3DLesson".

Although no modifications are necessary for this example, be aware that by copying the template, we have now duplicated the definition of dummyItem::onPickup(), dummyItem::onCollision(), and serverCmdbumpBlock()

Step #4 – Modify The Lesson Summary ML File

Because this lesson will not change the functionality of the template, we only need to open the file Lesson Summary ML File "**My3DLesson**.ml.txt" and change the **BOLD** parts of of this code:

```
<just:center><font:Arial Bold:18>
<spush><color:FFFF00>My 3D Lesson
<spop>
<font:Arial:16><br><br>
<just:left>
<lmargin%:5>
This is My 3D Lesson, showing some of the basics.<br><br>
Please see other lessons for more advanced feature usage and
don't forget to read the <spush><color:FF2222>'Appendix B -
GPGT Lesson Kit'<spop>.
<br>
<br>
<br>
<just:center>
<bitmap:gpgt/3DLessons/Volume1/My3DLesson/how>
```

Step #5 – Modify The Client Scripts

Although no modifications are necessary for this example, be aware that by copying the template, we have now duplicated the definition of 'bumpBlock()'.

B.1.9. Adding New Sample Scripts

In order to add new scripts that will be automatically loaded, simple create a script file (.cs) containing your new functions and be sure that the first three letters of the file's name are: **sts**.

Examples names:

- stsVol1_BasicScripting.cs
- stsVol1_AdvancedScripting.cs

Place you new file(s) in the 'gpgt' directory or in a sub-directory of 'gpgt' and the next time the kit is started, it will automatically find and loads the script file and all the function definitions therein.

B.2 Instructing and Collaborating

The GPGT Lesson Kit provides the ability for one person to host a 3D Lesson while other users participate remotely.

B.2.1. Running The 3D Lessons Server (MultiPlayer)

To run the GPGT Lesson Kit in Instructing/Collaborating mode, you must start a lesson server. To do so, simply follow these steps:

Load 3D Lessons Mission

Click on the ***Start Mission...*** button.

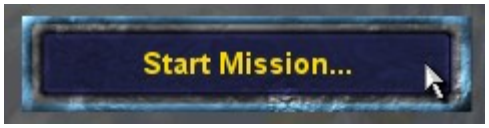


Figure B.2.1. *Start Mission...*

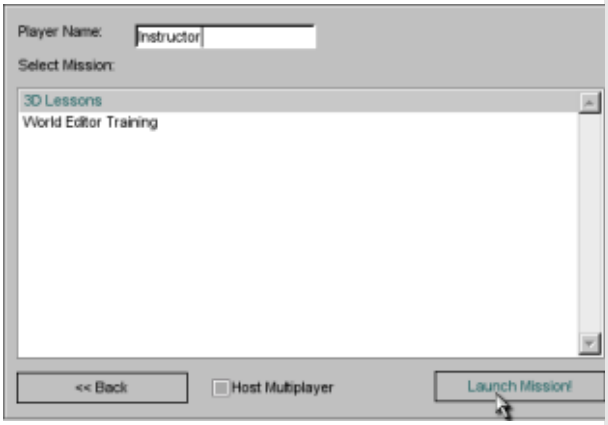


Figure B.2.2. Missions Dialog

Figure B.2.3. Loading the mission...

Launch 3D Lessons Mission

Select **3D Lessons** from the missions list, be sure that '**Host Multiplayer**' is checked/filled, and click the **Launch Mission!** button as in Figure B.2.2.

TGE will load the mission, load datablocks, do some lighting calculations and eventually pop you into the mission. See Figure B.2.3.

When you are dropped into the mission, you will find yourself in an empty grassy cauldron (Figure B.2.4).

Figure B.2.4. In the *3D Lessons* mission!

The remainder of the navigation steps proceed exactly as the do for singleplayer mode (documented in Section B.1.4 above).

Once the server is started, clients must attach. Read on to learn how clients attach to the lesson.

B.2.2. GPGT Lesson Client (LAN)

To run the GPGT Lesson Kit in Instructing/Collaborating mode, you must start a lesson server. To do so, simply follow these steps:

Join Server

Click on the ***Join Server...*** button.

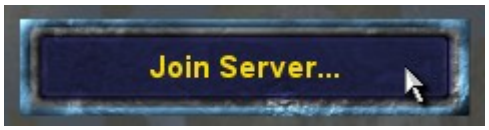


Figure B.2.5. Join Server...

Figure B.2.6. Server Query Dialog

Server Query

The **Lan Server** radio button is automatically clicked (you may click it again to repeat the query). Momentarily, the local server should be listed in the left scroll list. Clicking on an entry in this list will display details about that server in the right window. Once you have selected the server you wish to connect to you may optionally type a name for your player and the click on the **Join** button.

TGE will load the mission, load datablocks, do some lighting calculations and eventually pop you into the mission.

As a client, your interactions are somewhat limited. You are only allowed whatever freedom the instructor's/collaborator's lesson allows you. You cannot load new lesson. only the person hosting the 3D Lessons can do this.

B.2.3. GPGT Lesson Client (Remote Networks)

The GPGT Lesson Kit can also allow clients on remote networks to connect to the server, but to do so requires the following:

- The server must be outside of any firewall, or otherwise made visible on the Internet.
- The server machine or another machine elsewhere must be running a Master Server.

We did not discuss Master Servers in GPGT as this is a topic for a future book. However a rudimentary Master Server has been supplied and instructions for using it are provided below under Section "B.2.5. Running The Included Master Server".

Once a master server is running and at least GPGT server is advertising itself on the master server, the following steps can be followed by any client connected to the Internet:

Join Server

Click on the ***Join Server...*** button.

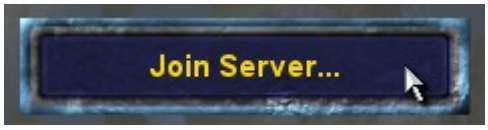


Figure B.2.7. Join Server...

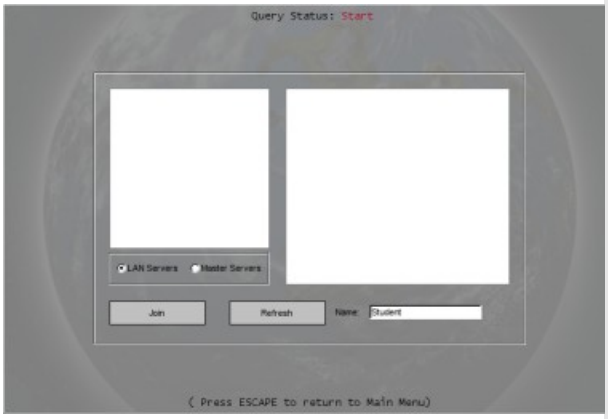


Figure B.2.8. Server Query Dialog

Server Query

Select the **Master Servers** radio butto. Momentarily, a list of available servers should be listed in the left scroll list. Clicking on an entry in this list will display details about that server in the right window. Once you have selected the server you wish to connect to you may optionally type a name for your player and the click on the **Join** button.

TGE will load the mission, load datablocks, do some lighting calculations and eventually pop you into the mission.

As a client, your interactions are somewhat limited. You are only allowed whatever freedom the instructor's/collaborator's lesson allows you. You cannot load new lesson. only the person hosting the 3D Lessons can do this.

B.2.4. Special Considerations For Lesson Hosting

As the person hosting a lesson, you control the level of interaction available to a client, but you must keep the following facts in mind:

- All Clients Run Client Scripts – Be sure not to provide any interaction features you do not want available to a client via keystrokes (action maps). Alternately, you may provide some features but not document them in the "Lesson Description Dialog".
- Every Client Will be Represented by a Single Blue Guy Avatar – In this edition of the GPGT Lesson kit, all clients arrive in a lesson as a player, represented by the Blue Guy. Future editions may provide other options, but for now, a lesson area can get very crowded if the maximum of 63 (as set by the kit not TGE) clients are allowed to attach. Be sure to state ground rules for those who have attached. At any time, as the host, you may kill client connection by looking at the player, noting its client ID, and then deleting that client from the console (~):

123.delete();

- Clients May Not Start New Lessons – Only the host may change the current lesson

B.2.5. Running The Included Master Server

On the accompanying disk, you will find a ZIP file named "masterserver.zip". It contains a master server implemented as a PERL script. This Master Server supplies the minimum set of features required to advertise and find remote servers. It does not provide any security features and should only be run in a safe computing environment.

B.2.5.1. Running the Master Server

To start the master server follow these directions (you must have PERL installed for this to work):

1. Unzip the masterserver.zip file to a directory of your choice.

2. Open the file "c3masterserver.pl" and find the line containing:

```
my $PORT = 7777;
```

3. Choose a port number for your server to advertise (28002 for example) and change this line of code to reflect the port number:

```
my $PORT = 28002;
```

4. Save the PERL script.

5. Execute the PERL script by double clicking it or ryping this on the command line:

```
perl c3masterserver.pl
```

B.2.5.2. Advertising the Lesson Server

Now that you have the master server running, to make the lesson server visible to clients, do this:

1. Edit the file "gpgt\client\prefs.cs" and find the line that contains this variable: $pref::Master0.

2. Edit the line so that it has the same IP address as the machine running the IP server (I usually run both the lesson server and the master server on the same machine). Also be sure that the port number is the same as the one you selected above. For example on my local network I use this:

```
$pref::Master0 = "2:192.168.123.15:28002";
```

3. Save the file and start the lesson server by following the direction listed in Section B.2.1. above.

B.2.5.3. Connecting Remote Clients

Now that you have the master server and the lesson server running, clients may connect by doing this:

1. Edit the file "gpgt\client\prefs.cs" and find the line that contains this variable:
   $pref::Master0.

2. Edit the line so matches the values used in Section B.2.5.2 above. For example:

```
$pref::Master0 = "2:192.168.123.15:28002";
```

3. Save the file and start the GPGT Lesson Kit.

4. Have students/collaborators connect by following the steps listed in Section B.2.3 above.

B.3 Recording Lesson Kit Sessions

It is possible to record a lesson kit session. That is, you may run the kit and record all inputs. Subsequently, another user can load these recorded inputs and replay the entire sequence. This is a standard feature (known as Journaling) of the Torque Game Engine which is often used for debugging failures, but which also lends itself to demonstrating exact sequences of events to a remote audience. In this short section, I will show you how to record a lesson, how to play it back, and they list a few prerecorded that you may play to see a demonstration of the lesson playback.

B.3.1. Recording A Lesson Kit Session

In order to record a lesson kit session, simply start the GPGT lesson kit from the command line as follows:

```
gpgt -jSave mySession.jrn
```

This will start the GPGT lesson kit and record all inputs to a journal file named "mySession.jrn" in the local directory. At the end of the session, this file will contain all inputs for th session. Any user may now play the session back exactly are recorded.

B.3.2. Replaying A Recorded Lesson Kit Session

In order to replay a lesson kit session, simply start the GPGT lesson kit from the command line as follows:

```
gpgt -jPlay mySession.jrn
```

This will start the GPGT lesson kit in journal-replay mode and will play back the journal "mySession.jrn" located in the local directory. Of course, this relies on the existence of the journal file, but I am assuming that you just recorded one following the steps above in section B.3.1. If not, please record a journal and then try replaying it, or replay one of the included sessions as listed in Section B.3.3. below.

Please note, during the replay of a lesson kit session, no external inputs will be accepted by the GPGT lesson kit until the session is over.

B.3.3. Some Prerecorded Lesson Kit Sessions

In order to provide you some samples, the following GPGT lesson kit sessions have been prerecorded for you:

Lesson Kit Session	Description
GUISampler1.jrn	Demonstrates guiTextListCtrl.
GUISampler2.jrn	Demonstrates guiMouseEventCtrl.
ToonInterfaces.jrn	Demonstrates Toon Interfaces.
HUDs.jrn	Demonstrates the three HUD samples.
3DLessons.jrn	Demonstrates several 3D Lessons.
SampleScriptsConsole.jrn	Demonstrates usage of Sample Scripts Console.

B.4 GPGT 3D Lesson Descriptions

B.4.0. Lesson Template

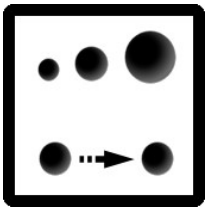
B.4.0.1. SimpleSampleLesson

This is a lesson template. You can use it to create new lessons. Please see the "Lesson Kit User's Guide" appendix for instructions on creating your own lessons.

Example	Description	Location
Bouncing Box	Press the 'b' key to apply an impulse to the bottom of this box.	North 10

B.4.1. SceneObject

B.4.1.1. SceneObject Transforms

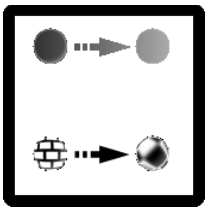


This lesson shows how to use the various translation, rotation, and scaling methods provided by the SceneObject class.

Example	Description	Location
leftEgg (wrong)	leftEgg shows that using code like: `%obj.position = "10 10 10";` will not translate an object.	North 10
rightEgg (correct)	rightEgg shows that using code like: `%obj.setTransform( "10 10 10" );` will translate an object.	North 10
initializedEgg setEgg	initializedEgg uses a scale value provided in the new block, while setEgg is scaled as a result of a call to setScale().	East 10
smallEgg normalEgglargeEgg	Eggs scaled in to various sizes.	West 10

B.4.2. ShapeBase (General)

B.4.2.1. ShapeBase Rendering



Shows various shapeBase supported rendering features, including:
- Fading
- Fading & Hiding
- Cloaking Levels
- Blank cloakTexture
- Re-Skinning

Marker	Description	Object(s)
North 10	Demonstrates the ability to fade an shape out of view. Also shows that a faded object can still be collided with.	fadeEgg
East 10	Demonstrates that setHidden() will take a shape out of the world, including its collision mesh.	FadeHideEgg
South 10	Demonstrates re-skinning with setSkinName().	--
West 10	Demonstrates Various Levels of Alpha in a cloakTexture.	normalCloak (left) cloakMore (middle) totalCloak (right)
West 30	Demonstrates cloaking with cloakTexture not set.	NoCloakEgg

B.4.2.2. ShapeBase Damage

Explores various shapeBase supported damage features, including:
- Damaging
- Disabling
- Destroying
- Exploding
- repairing
- Invincibility

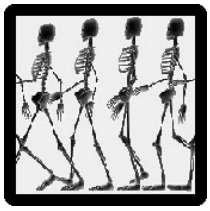
Marker	Description	Object(s)
North 10	Demonstrates damage and self-repair.	SelfHealingBlock
East 10	Demonstrate damage and disabling.	DisableGears
South 10	Demonstrates damage, destruction, post-destruction rendering, and explosion.	ExplodeGears
West 10	Demonstrates Invincibility.	InvincibleBlock

B.4.2.3. ShapeBase Energy

Explores various shapeBase supported energy features, including:
- Energy Consumption
- Recharging

Marker	Description	Object(s)
North 10	Demonstrates energy consumption and recharging	SelfRechargingBlock
East 10	Demonstrated disabling due to lack of energy.	DisableEnergyGears

B.4.2.4. ShapeBase Animations



Explores various shapeBase supported animation features, including:
- Blended
- Cyclic

Marker	Description	Object(s)
North 10	Demonstrates combinations of cyclic blended animations.	BlendAnimAlone0 BlendAnimAlone1 BlendCombo

B.4.2.5. ShapeBase Sounds

Explores use of sound threads for shapes, including differenct behaviors of:
- MONO (16-bit) Sounds
- STEREO (16-bit) Sounds

Marker	Description	Object(s)
North 10	Demonstrates difference between a MONO and a STEREO sound file when used for 3D sounds.	soundEgg

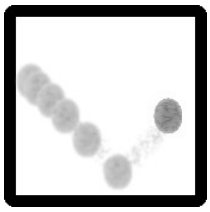
B.4.3. Item

B.4.3.1. Item Rendering

Show various Item/ItemData supported rendering features, including:
- Constant Lights
- Pulsing Lights
- Constant Light + Fading
- Constant Light + Cloaking
- Pulsing Light + Cloaking

Marker(s)	Description	Object(s)
North 10 North 20 North 30	There are three items (eggs) at this marker locations. These eggs each emit a constant light of either Red, Green, or Blue.	ConstantLightEgg0 ConstantLightEgg1 ConstantLightEgg2
East 10 East 20 East 30	There are three items (eggs) at this marker locations. These eggs each emit a pulsing light (1.5 second period).	PulsingLightEgg0 PulsingLightEgg1 PulsingLightEgg2
South 10	One egg is rendered with a constant yellow light and then faded in and out over time. Notice that the light also fades in and out to match the translucency of the egg.	FadeAndHide-ConstantLightEgg
West 10	One egg is rendered with a constant yellow light and then cloaked. Notice that the light is still fully visible.	CloakedConstantLightEgg
West 20	One egg is rendered with a pulsing yellow light and then cloaked. Notice that the light is still fully visible.	CloakedPulsingLightEgg

B.4.3.2. Item Physics

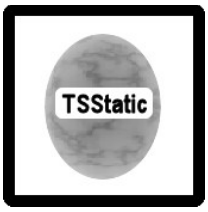


Show various Item/ItemData supported physics features, including:
- Static vs. non-static
- Rotating
- Sticky vs. non-sticky
- Variations on friction
- Gravity Effects
- Elasticity

Marker	Description	Object(s)
North 10	In this sample, two eggs are repeatedly dropped from a low height. The left egg is non-static and thus falls, the right egg is made static and stays where it is placed.	NonStaticEgg StaticEgg
North 20	Demonstrates the effect of setting Item.rotate to true.	RotatingEgg
North 50	Demonstates stickiness property. Both eggs have been made bouncy, but one is sticky and the other is not. The sticky one does not bounce while the non-sticky one does.	StickyEgg NonStickyEgg
South 10	This sample demonstrates friction and its effect on a sliding object. From left to right, the items have the following friction values: -2.0, 0.0, 0.3, and 2.0. The eggs are all simultaneously placed and then boosted towards the hillside to show the effect of different frictions. Notice that the one with a negative friction actually speeds up and the one with a high friction stops almost immediately.	NegativeFrictionEgg (-2.0) ZeroFrictionEgg (0.0) MediumFrictionEgg (0.3) HighFrictionEgg (2.0)
West 30	This sample demonstrates the results of various elasticity settings. From left to right, the eggs have the following elasticity settings: 0.0 , 0.1, 0.5, 1.0, and 2.0. Notice, that the 0.0 actually hesitates and then settles, 0.1 settles immediately, 1.0 does not actually bounce back to its starting position, and 2.0 bounces higher and higher. The lesson here is that elasticity calculations have some error, so you must adjust your game settings to match your need. Do not assume the values will just work without testing them.	NoBounceEgg (0.0) LowBounceEgg (0.1) MediumBounceEgg (0.5) FullBounceEgg (1.0) OverBounceEgg (2.0)

B.4.4. TSStatic

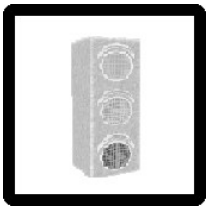
B.4.4.1. Static Shapes



Shows very simple uses of TSStatic.

B.4.5. ShapeBaseImageData (Images)

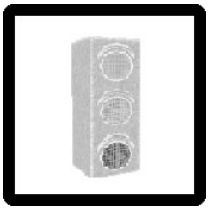
B.4.5.1. Image State Machines



Shows sample usage of ShapeBaseImage state machine. This is the sample from the book with a slight modification to the Green Light - Yellow Light - Red Light ... repeat timing.

B.4.5. InteriorInstance

B.4.6.1. Interiors

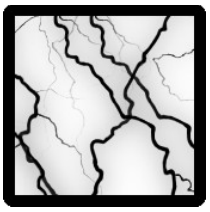


Shows sample InteriorInstance features, , including:
- LOD
- Re-lighting

Marker	Description	Object(s)
North 10	A sample interior is shown. It has 5 LOD levels. You may switch between them manually and examine the effect of relighting.	InteriorInstance

B.4.7. Mission Objects

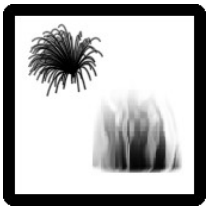
B.4.7.1. Lightning



This lesson demonstrates:
- Lightning
- Thunder
- Lightning Scripting

Marker	Description	Key Strokes
--	Lighting Strike (Generated ONLY).	1
--	Use generated lightning.	g
--	Use textured lightning.	t

B.4.7.2. Particles



Particles lesson, showing sample definitions of:
- ParticleData
- ParticleEmitterData
- ParticleEmitterNode Data
- ParticleEmitterNode

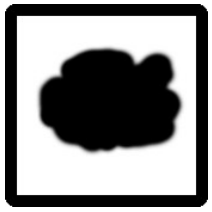
Marker	Description	Object(s)
North 10	Bubbles.	--
East 10	Two variations on smoke.	--
South 10	Two variations on fire. The left fire uses animated particles, and the right is the same particle generated in various sizes.	--
West 10	Sparks.	--

B.4.7.3. Precipitation (Sky Object)

This lesson demonstrates a few variations on precipitation:
- Raining Cats 'n Dogs (key 1)
- Snowing (key 2)
- Cartoon Rain (key 3)

Name	Description	Keystrokes
Raining Cats 'n Dogs	A goofy example exploring basic rain with splashes and sound.	Press Key 1
Snowing	A second example demonstrating a soundless snowfall. **Note:** Because we are using the same mission, and because Sky.windEffectPrecipitation is true, we've made the mass of the snowflakes quite high so they will not be affected much by the wind.	Press Key 2
Cartoon Rain	A final example, using low-mass rain-drops plus light turbulence.	Press Key 3

B.4.7.4. StormClouds (Sky Object)



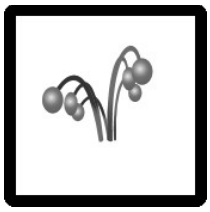
Simple demonstration of stormClouds feature.

Clouds fade in (100% on) and out (0% on) on key press.

Marker	Description	Keystrokes
--	In this simple example, the clouds fade in and out based on a keystroke.	Press Key 1 to Toggle

B.4.8. Special Effects

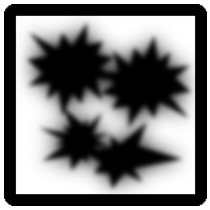
B.4.8.1. Debris



Debris lesson, showing examples of:
- 3D Debris
- Debris w/ Fire Trail (particles)
- Fast Moving Debris
- Fading Debris
- Non-Fading (popping) Debris
- Debris w/ Low Friction
- Bouncing Debris
- Static on maxBounce

Marker	Description	Object(s)
North 10	Basic 3D Debris, with no fade.	--
North 30	2D Debris.	--
East 10	Fast Moving 3D Debris.	--
East 30	3D Debris with flaming trail.	--
South 10	Basic 3D Debris, with random fade.	--
West 10	Bouncy 3D Debris.	--
West 30	Replace w/ Static 3D Debris.	--
West 70	Sliding Debris. (Uses large debris so you can see them.)	--

B.4.8.2. Explosions



Explosion lesson, showing examples of:
- Sub-Explosions
- Particle Usage
- Debris
- Camera Shake

Marker	Description	Object(s)
North 20	Alternates between big explosion with debris, smoke, and camera shake **AND** a confetti explosion.	--

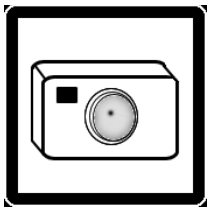
B.4.8.3. Projectiles

Projectiles lesson, showing examples of:
- Ballistic Projectiles
- Non-Ballistic Projectiles
- Delayed Activation
- Projectile Particle Usage
- Projectile Explosions

Marker	Description	Object(s)
North 10	Slow moving non-ballistic projectile.	--
South 10	A ballistic-projectile demonstrating delayed activation, elasticity, and friction.	--
West 10	A purely ballistic projectile.	--

B.4.9. Game Elements

B.4.9.1. Game Views



In this lesson, we explore the effects of:
- POV Settings
- FOV Settings.
- setFOV()
- setZoomSpeed()

Marker	Description	Keystrokes
--	Switch camera datablock.	1
--	Switch player datablock.	2
--	Increase FOV.	Up Arrow
--	Decrease FOV.	Down Arrow
--	Increase Zoom Speed.	Right Arrow
--	Decrease Zoom Speed.	Left Arrow
--	Zoom.	E
--	Toggle free-camera mode.	CTRL+C
--	Toogle 1st and 3rd POV.	TAB

Experiment with various combinations of camera datablocks, player datablocks, POV modes, and camera free modes. Also experiment with zooming. For example, try these settings:

- Player DB: testAvatar8
- Camera DB: testCamera2
- POV: 3rd
- Zoom Speed: 2000

Now experiment with zooming in and out by changing the zoom value with the up and down arrow keys.

B.4.9.2. Simple Inventory

In this lesson, the following inventory actions are demonstrated:
- pickup
- drop/throw
- use

Marker	Description	Keystrokes
North 10 .. 40 South 10 .. 40	Coin inventory items that self-replace, fade in/out, and emit light. They can be Thrown/Dropped.	Press t to throw coin.
East 10 .. 40 West 10 .. 40	Holy Hand grenade inventory items that self-replace, fade in/out, and emit light. They can be Used or Thrown/Dropped.	Press g to use. (explodes) Press CTRL + g to throw.

B.4.9.3. Basic Vehicles

Vehicle Sampler, showing:
- Box Car
- Box Hover
- Psionic Jeep
- Flying Vehicle

Marker	Description	Keystrokes
North 10	Shows three mountable, dismountable, and drivable vehicles.	Press Key 1 – Box Car Press Key 2 – Psionic Jeep Press Key 3 – Box Hover Press Key 4 – Box Flyer
North 10	Same vehicles can be 'bounced.	Press b to bounce vehicle.