Using the G-key SDK with Epic Games' UDK and DLLBind

Contents

Overview

The Logitech Gaming G-key SDK enables to get the current state of G-keys and extra mouse buttons for supported Logitech gaming mice and keyboards. It can be integrated in a UDK game by using DLLBind. It only works when the Logitech Gaming Software is running (8.55+)
Please refer to the Logitech SDK's Doc\LogitechGamingGkeySDK.pdf for details on the SDK's functionality.

Making the G-key SDK work in your UDK game

The following steps show how to make the Logitech SDK work with "UDKGame" that comes as part of the UDK download. Please adapt the steps to your game for things to work.

Steps

1. Download UDK (http://www.unrealengine.com/udk/).
2. Create the following 2 Unreal Script files, and copy them to UDK's Development\Src\MyMod\Classes folder:

TestDLLGameInfo.uc

```
class TestDLLGameInfo extends GameInfo;

defaultproperties
{
	PlayerControllerClass=class'TestDLLPlayerController'
}
```

TestDLLPlayerController.uc

```
class TestDLLPlayerController extends PlayerController
	DLLBind(LogitechGkeyEnginesWrapper);

dllimport final function bool LogiGkeyInitWithoutCallback();
dllimport final function bool LogiGkeyIsMouseButtonPressed(int buttonNumber);
dllimport final function string LogiGkeyGetMouseButtonString(int
buttonNumber);
dllimport final function bool LogiGkeyIsKeyboardGkeyPressed(int gkeyNumber,
int modeNumber);
dllimport final function string LogiGkeyGetKeyboardGkeyString(int gkeyNumber,
int modeNumber);
dllimport final function LogiGkeyShutdown();

exec function LogiInit()
{
	local bool ret;
	ret = LogiGkeyInitWithoutCallback();

	say("LogiGkeyInitWithoutCallback return is: " $ret);
}

exec function LogiIsMouseButtonPressed(int buttonNumber)
{
	local bool ret;
```

```
	ret = LogiGkeyIsMouseButtonPressed(buttonNumber);

	say("LogiGkeyIsMouseButtonPressed return is: " $ret);
}

exec function LogiGetMouseButtonString(int buttonNumber)
{
	local string buttonString;
	buttonString = LogiGkeyGetMouseButtonString(buttonNumber);

	say("LogiGetMouseButtonString: " $buttonString);
}

exec function LogiIsKeyboardGkeyPressed(int gkeyNumber, int modeNumber)
{
	local bool ret;
	ret = LogiGkeyIsKeyboardGkeyPressed(gkeyNumber, modeNumber);

	say("LogiGkeyIsKeyboardGkeyPressed return is: " $ret);
}

exec function LogiGetKeyboardGkeyString(int gkeyNumber, int modeNumber)
{
	local string gkeyString;
	gkeyString = LogiGkeyGetKeyboardGkeyString(gkeyNumber, modeNumber);

	say("LogiGkeyGetKeyboardGkeyString: " $gkeyString);
}

exec function LogiShutdown()
{
	LogiGkeyShutdown();

	say("LogiGkeyShutdown done");
}
```

3. Copy Logitech SDK's Lib\GameEnginesWrapper\x86\LogitechGkeyEnginesWrapper.dll to UDK's Binaries\Win32\UserCode
4. Copy Logitech SDK's Lib\ GameEnginesWrapper\x64\LogitechGkeyEnginesWrapper.dll to UDK's Binaries\Win64\UserCode
5. Open UDK's UDKGame\Config\ DefaultEngineUDK.ini file for editing
    a. Search for: `ModEditPackages=MyMod`
    b. Remove the ; at the beginning of the line
6. Launch UDK's Binaries/UnrealFrontend.exe
    a. Do: Script->Full recompile

Calling Logitech SDK's functions from within the game

Launch game the following way:

- Binaries\Win32\UDK.exe dm-deck?game=MyMod.TestDLLGameInfo

Once the game is running, open the console (hit the ~ key), and type: `LogiInit`, and then hit <enter> key. You should see the "LogiGkeyInitWithoutCallback return is: TRUE" message.

Before trying other commands, especially if trying a mouse, make sure its buttons have been set up to be sent to the Logitech SDK rather than keyboard shortcuts (see Logitech SDK's Demo\ReadMe.txt).

Then use the other commands as defined in the Unreal Script file:

- `LogiIsMouseButtonPressed`
- `LogiGetMouseButtonString`
- `LogiIsKeyboardGkeyPressed`
- `LogiGetKeyboardGkeyString`
- `LogiShutdown`

For example to see if a supported keyboard's G-key 3 in mode 2 is currently pressed, type:

- `LogiIsKeyboardGkeyPressed 3 2`

Then hit <enter>, all the while holding that G-key down.

NOTE: when you run the 'LogiInit' command in the UDK, LGS will automatically create a new profile called UDK if there isn't one already present. However, note that if you run a version of UDK using different architecture (32-bit vs 64-bit), it will not create another new profile automatically. One must be manually created using the LGS software. Therefore, if you wish to use LGS with both 32-bit and 64-bit UDK.exe, please ensure you have a profile for each executable.

For questions/comments, email devtechsupport@logitech.com