Using the Logitech Gaming LCD SDK with Epic Games' UDK and DLLBind

Contents

Overview

The Logitech Gaming LCD SDK enables applications such as games to control the LCDs on supported Logitech gaming mice and keyboards.

It can be easily integrated in a UDK game by using DLLBind.

Please refer to the Logitech SDK's Doc\LogitechGamingLCDSDK.pdf for details on the SDK's functionality.

Making the LCD 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(LogitechLcdEnginesWrapper);



dllimport final function bool LogiLcdInit(string friendlyName, int lcdType);
dllimport final function bool LogiLcdIsConnected(int lcdType);
dllimport final function bool LogiLcdIsButtonPressed(int button);
dllimport final function bool LogiLcdUpdate();
dllimport final function bool LogiLcdShutdown();
dllimport final function bool LogiLcdMonoSetText(int lineNumber, string
text);
dllimport final function int  LogiLcdColorSetBackgroundUDK(byte
colorBitmap[2048], int arraySize);
dllimport final function int  LogiLcdColorResetBackgroundUDK();
dllimport final function int  LogiLcdMonoSetBackgroundUDK(byte
monoBitmap[2048], int arraySize);
dllimport final function int  LogiLcdMonoResetBackgroundUDK();
```

```
dllimport final function bool LogiLcdColorSetTitle(string text, int red , int
green , int blue );
dllimport final function bool LogiLcdColorSetText(int lineNumber, string
text, int red, int green, int blue);

exec function LogitechLCDInit(string friendlyName, int lcdType)
{
      local bool ret;
      ret = LogiLcdInit(friendlyName, lcdType);
      say("LogitechLCDInit return is: " $ret);
}

exec function LogitechLCDIsConnected(int lcdType)
{
      local bool ret;
      ret = LogiLcdIsConnected(lcdType);
      say("LogitechLCDIsConnected return is: " $ret);
}

exec function LogitechLcdIsButtonPressed(int button)
{
      local bool ret;
      ret = true;
      ret = LogiLcdIsButtonPressed(button);
      say("LogitechLcdIsButtonPressed return is: " $ret);
}

exec function LogitechLcdUpdate()
{
      local bool ret;
      ret = LogiLcdUpdate();
      say("LogitechLcdUpdate return is: " $ret);
}

exec function LogitechLcdShutdown()
{
      LogiLcdShutdown();
      say("LogitechLcd SDK shutting down...");
}


exec function LogitechLcdMonoSetText(int lineNumber, string text)
{
      local bool ret;
      ret = LogiLcdMonoSetText(lineNumber, text);
      say("LogitechLcdMonoSetBackground return is: " $ret);
}


exec function LogitechLcdColorSetPageBackGround(int blue, int green, int red,
int alpha)
{
      local int i;
      local int byteAdded;
      local byte pixelMatrix[2048];
      byteAdded = 1;
      for(i=0; i<2048; i++)
```

```
	{
		if((i%4) == 0) pixelMatrix[i] = byte(blue); // blue
		if((i%4) == 1) pixelMatrix[i] = byte(green); // green
		if((i%4) == 2) pixelMatrix[i] = byte(red); // red
		if((i%4) == 3) pixelMatrix[i] = byte(alpha); // alpha
	}
	while(byteAdded > 0)
	{
		byteAdded = LogiLcdColorSetBackgroundUDK(pixelMatrix, 2048);
	}
	if (byteAdded == 0) say("Color page Background set successfully");
	else say("Error setting color page background " $byteAdded);
	LogiLcdUpdate();
}

exec function LogitechLcdMonoSetPageBackGround(int blackOrWhite)
{
	local int byteAdded;
	local int i;
	local byte pixelMatrix[2048];
	byteAdded = 1;
	for(i=0; i<2048; i++)
	{
		pixelMatrix[i] = blackOrWhite;
	}
	while(byteAdded > 0)
	{
		byteAdded = LogiLcdMonoSetBackgroundUDK(pixelMatrix, 2048);
	}
	if (byteAdded == 0) say("Mono page Background set successfully");
	else say("Error setting mono page background " $byteAdded);
	LogiLcdUpdate();
}

exec function LogitechLcdMonoResetBitmap(){
	local int ret;
	ret = LogiLcdMonoResetBackgroundUDK();
	say("LogitechLcdMonoResetBKG return is: " $ret);
}

exec function LogitechLcdColorResetBitmap(){
	local int ret;
	ret = LogiLcdColorResetBackgroundUDK();
	say("LogitechLcdColorResetBKG return is: " $ret);
}

exec function LogitechLcdColorSetTitle(string text, int red , int green , int
blue )
{
	local bool ret;
	ret = LogiLcdColorSetTitle(text, red, green, blue);
	say("LogitechLcdColorSetTitle return is: " $ret);
}

exec function LogitechLcdColorSetText(int lineNumber, string text, int red,
int green, int blue)
{
```

```
        local bool ret;
        ret = LogiLcdColorSetText(lineNumber, text, red, green, blue);
        say("LogitechLcdColorSetText return is: " $ret);
}
```

3. Copy Logitech SDK's Lib\GameEnginesWrapper\x86\ LogitechLcd.dll to UDK's Binaries\Win32\UserCode
4. Copy Logitech SDK's Lib\GameEnginesWrapper\x64\ LogitechLcd.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: `LogitechLCDInit`, and then hit <enter> key. You should see the " `LogiLCDInit` return is: TRUE" message.
To effectively see the app running on your LCD you need to call the `LogitechLcdUpdate` at least once. To keep your app updated on the lcd you have to call this function every frame of your game.

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

- `LogitechLCDIsConnected`
- `LogitechLcdIsButtonPressed`
- `LogitechLcdUpdate`
- `LogitechLcdShutdown`
- `LogitechLcdMonoSetText`
- `LogitechLcdColorSetPageBackGround`
- `LogitechLcdMonoSetPageBackGround`
- `LogitechLcdMonoResetBitmap`
- `LogitechLcdColorResetBitmap`
- `LogitechLcdColorSetTitle`
- `LogitechLcdColorSetText`
- `LogitechLcdColorSetText`

UDK DLLBind specific functions definitions

Due to the limitation of the UnrealScript language of the static byte array size to 2048, to send bitmaps to the LCDs, you will have to use the UDK specific functions.

LogiLcdColorSetBackgroundUDK

The **LogiLcdColorSetBackgroundUDK** fills up a local buffer with the specified partial bitmap array every time it gets called. When the buffer reaches the exact size of 320*240*4 = 307200, this function sends the bitmap array to the connected LCD, calling the standard LogiLcdColorSetBackground function.

Parameters

- partialBitmap : the partial array for the color bitmap you want to send to the SDKs
- arraySize: the size of the partialBitmap you are sending, the maximum value for this parameter is 2048, anything bigger will be truncated.

Return value

It returns 0, if the local buffer has been filled up and the LogiLcdColorSetBackground has been called successfully. While the buffer is filling up, the function returns the number of bytes actually stored in the local buffer. If the buffer is filled up and the function LogiLcdColorSetBackground fails, it returns -1 as error code.

For an example on how to use properly this function check the example script above. In the function LogitechLcdColorSetPageBackGround, you can see this chunk of code :

```
for(i=0; i<2048; i++)
{
      if((i%4) == 0) pixelMatrix[i] = byte(blue); // blue
      if((i%4) == 1) pixelMatrix[i] = byte(green); // green
      if((i%4) == 2) pixelMatrix[i] = byte(red); // red
      if((i%4) == 3) pixelMatrix[i] = byte(alpha); // alpha
}

while(byteAdded > 0)
{
      byteAdded = LogiLcdColorSetBackgroundUDK(pixelMatrix, 2048);
}
if (byteAdded == 0) say("Color page Background set successfully");
else say("Error setting color page background " $byteAdded);
```

The first loop initializes an array of 2048 byte (pixelMatrix) with the given parameters.
In the following while loop, we call the LogiLcdColorSetBackgroundUDK function as many time as needed to fill the complete bitmap using pixelMatrix as partial array for each call.
The loop exit condition means that while the return value is positive, we are adding bytes to the SDKs buffer, when exiting the loop either the background has been set successfully (byteAdded = 0) or an error as occurred (byteAdded = -1).
In order to get the script to enter this loop you have to initialize the byteAdded variable to a positive value (in the example is intitialized to 1).

Note

For a better knowledge on how to dispose the bytes for your color bitmap in the array, check the document LogitechGamingLCDSDK in the Doc folder of this package.

LogiLcdColorResetBackgroundUDK

The **LogiLcdColorResetBackgroundUDK** function resets the local buffer. You should use this function if you receive an error code while filling up the buffer for the bitmap or if you get lost and simply want to restart filling the color buffer from scratch.

Return value

It returns the number of the bytes deleted from the color buffer.

LogiLcdMonoSetBackgroundUDK

The **LogiLcdMonoSetBackgroundUDK** fills up a local buffer with the specified partial bitmap array every time it gets called. When the buffer reaches the exact size of 160*43 = 6880, this function sends the bitmap array to the connected LCD, calling the standard LogiLcdMonoSetBackground function.

Parameters

- partialBitmap : the partial array for the monochrome bitmap you want to send to the SDKs
- arraySize: the size of the partialBitmap you are sending, the maximum value for this parameter is 2048, anything bigger will be truncated.

Return value

It returns 0, if the local buffer has been filled up and the LogiLcdMonoSetBackground has been called successfully. While the buffer is filling up, the function returns the number of bytes actually stored in the local buffer. If the buffer is filled up and the function LogiLcdMonoSetBackground fails, it returns -1 as error code.

For an example on how to use properly this function check the example script above. In the function LogitechLcdMonoSetPageBackGround, you can see this chunk of code :

```
for(i=0; i<2048; i++)
{
      pixelMatrix[i] = blackOrWhite;
}
while(byteAdded > 0)
{
      byteAdded = LogiLcdMonoSetBackgroundUDK(pixelMatrix, 2048);
}
if (byteAdded == 0) say("Mono page Background set successfully");
else say("Error setting mono page background " $byteAdded);
```

The first loop initializes an array of 2048 byte (pixelMatrix) with the given parameters.
In the following while loop, we call the LogiLcdMonoSetBackgroundUDK function as many time as needed to fill the complete bitmap using pixelMatrix as partial array for each call.
The loop exit condition means that while the return value is positive, we are adding bytes to the SDKs buffer, when exiting the loop either the background has been set successfully (byteAdded = 0) or an error as occurred (byteAdded = -1).
In order to get the script to enter this loop you have to initialize the byteAdded variable to a positive value (in the example is intitialized to 1).

Note

For a better knowledge on how to dispose the bytes for your monochrome bitmap in the array, check the document LogitechGamingLCDSDK in the Doc folder of this package.

LogiLcdMonoResetBackgroundUDK

The **LogiLcdMonoResetBackgroundUDK** function resets the local buffer. You should use this function if you receive an error code while filling up the buffer for the bitmap or if you get lost and simply want to restart filling the monochrome buffer from scratch.

Return value

It returns the number of the bytes deleted from the monochrome buffer.

For questions/comments, email devtechsupport@logitech.com