Using the Steering Wheel SDK with Epic Games' UDK and DLLBind

Contents

Overview

The Logitech Gaming Steering Wheel SDK enables control to different game controllers, such as steering wheels, gamepads and joysticks.

It can be integrated in a UDK game by using DLLBind. It only works when the Logitech Gaming Software is running (5.10 or later).

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

Making the Steering Wheel 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(LogitechSteeringWheel);

struct LogiControllerPropertiesData
{
    var bool forceEnable;
    var int overallGain;
    var int springGain;
    var int damperGain;
    var bool defaultSpringEnabled;
    var int defaultSpringGain;
    var bool combinePedals;
    var int wheelRange;
    var bool gameSettingsEnabled;
    var bool allowGameSettings;
};

struct DIJOYSTATE2ENGINES {
    var int lX;
    var int lY;
    var int lZ;
```

```
    var int lRx;
    var int lRy;
    var int lRz;
    var int rglSlider[2];
    var int    rgdwPOV[4];
    var byte   rgbButtons[128];
    var int lVX;
    var int lVY;
    var int lVZ;
    var int lVRx;
    var int lVRy;
    var int lVRz;
    var int rglVSlider[2];
    var int lAX;
    var int lAY;
    var int lAZ;
    var int lARx;
    var int lARy;
    var int lARz;
    var int rglASlider[2];
    var int lFX;
    var int lFY;
    var int lFZ;
    var int lFRx;
    var int lFRy;
    var int lFRz;
    var int rglFSlider[2];
};



dllimport final function bool LogiSteeringInitialize(int ignoreXInputControllers);
dllimport final function bool LogiUpdate();
dllimport final function bool LogiIsConnected(const int index);
dllimport final function string LogiGetFriendlyProductName(const int index);
dllimport final function bool LogiIsDeviceConnected(const int index, const int deviceType);
dllimport final function bool LogiIsManufacturerConnected(const int index, const int
manufacturerName);
dllimport final function bool LogiIsModelConnected(const int index, const int modelName);
dllimport final function bool LogiButtonTriggered(const int index, const int buttonNbr);
dllimport final function bool LogiButtonReleased(const int index, const int buttonNbr);
dllimport final function bool LogiButtonIsPressed(const int index, const int buttonNbr);
dllimport final function bool LogiGenerateNonLinearValues(const int index, const int
nonLinCoeff);
dllimport final function int  LogiGetNonLinearValue(const int index, const int inputValue);
dllimport final function bool LogiHasForceFeedback(const int index);
dllimport final function bool LogiIsPlaying(const int index, const int forceType);
dllimport final function bool LogiPlaySpringForce(const int index, const int offsetPercentage,
const int saturationPercentage, const int coefficientPercentage);
dllimport final function bool LogiStopSpringForce(const int index);
dllimport final function bool LogiPlayConstantForce(const int index, const int
magnitudePercentage);
dllimport final function bool LogiStopConstantForce(const int index);
dllimport final function bool LogiPlayDamperForce(const int index, const int
coefficientPercentage);
dllimport final function bool LogiStopDamperForce(const int index);
dllimport final function bool LogiPlayFrontalCollisionForce(int index, int magnitudePercentage);
dllimport final function bool LogiPlaySideCollisionForce(const int index, const int
magnitudePercentage);
dllimport final function bool LogiPlayDirtRoadEffect(const int index, const int
magnitudePercentage);
dllimport final function bool LogiStopDirtRoadEffect(const int index);
```

```
dllimport final function bool LogiPlayBumpyRoadEffect(const int index, const int
magnitudePercentage);
dllimport final function bool LogiStopBumpyRoadEffect(const int index);
dllimport final function bool LogiPlaySlipperyRoadEffect(const int index, const int
magnitudePercentage);
dllimport final function bool LogiStopSlipperyRoadEffect(const int index);
dllimport final function bool LogiPlaySurfaceEffect(const int index, const int type, const int
magnitudePercentage, const int period);
dllimport final function bool LogiStopSurfaceEffect(const int index);
dllimport final function bool LogiPlayCarAirborne(const int index);
dllimport final function bool LogiStopCarAirborne(const int index);
dllimport final function bool LogiPlaySoftstopForce(const int index, const int
usableRangePercentage);
dllimport final function bool LogiStopSoftstopForce(const int index);
dllimport final function bool LogiSetPreferredControllerProperties(LogiControllerPropertiesData
properties);
dllimport final function bool LogiGetCurrentControllerProperties(const int index, out
LogiControllerPropertiesData properties);
dllimport final function int  LogiGetShifterMode(const int index);
dllimport final function bool LogiPlayLeds(const int index, const float currentRPM, const float
rpmFirstLedTurnsOn, const float rpmRedLine);
dllimport final function DIJOYSTATE2ENGINES LogiGetStateENGINES(const int index);
dllimport final function void LogiSteeringShutdown();

exec function LogiSWInitialize(int ignoreXInputControllers)
{
	local bool ret;
	ret = LogiSteeringInitialize(ignoreXInputControllers);

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

exec function LogiSWUpdate()
{
	local bool ret;
	ret = LogiUpdate();

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

exec function LogiSWGetStateUDK(int index)
{
	local DIJOYSTATE2ENGINES ret;
	ret = LogiGetStateENGINES(index);

	say("x-axis position is: " $(ret.lX));
	say("y-axis position is: " $(ret.lY));
	say("z-axis position is: " $(ret.lZ));
	say("x-axis rotation is: " $(ret.lRx));
	say("y-axis rotation is: " $(ret.lRy));
	say("z-axis rotation is: " $(ret.lRz));
	say("extra axes positions 1 is: " $(ret.rglSlider[0]));
	say("extra axes positions 2 is: " $(ret.rglSlider[1]));


}

exec function LogiSWGetFriendlyProductName(int index)
{
	local string ret;
	ret = LogiGetFriendlyProductName(index);
```

```
        say("LogiSWGetFriendlyProductName return is: " $ret);
}

exec function LogiSWIsConnected(int index)
{
        local bool ret;
        ret = LogiIsConnected(index);

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

exec function LogiSWIsDeviceConnected(int index, int deviceType)
{
        local bool ret;
        ret = LogiIsDeviceConnected(index, deviceType);

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

exec function LogiSWIsManufacturerConnected(int index, int manufacturerName)
{
        local bool ret;
        ret = LogiIsManufacturerConnected(index, manufacturerName);

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

exec function LogiSWIsModelConnected(int index, int modelName)
{
        local bool ret;
        ret = LogiIsModelConnected(index, modelName);

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

exec function LogiSWButtonTriggered(int index, int buttonNbr)
{
        local bool ret;
        ret = LogiButtonTriggered(index, buttonNbr);

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

exec function LogiSWButtonReleased(int index, int buttonNbr)
{
        local bool ret;
        ret = LogiButtonReleased(index, buttonNbr);

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

exec function LogiSWButtonIsPressed(int index, int buttonNbr)
{
        local bool ret;
        ret = LogiButtonIsPressed(index, buttonNbr);

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

exec function LogiSWGenerateNonLinearValues(int index, int nonLinCoeff)
{
        local bool ret;
```

```
        ret = LogiGenerateNonLinearValues(index, nonLinCoeff);

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


exec function LogiSWFrontalCollision(int index, int magnitudePercentage)
{
        local bool ret;
        ret = LogiPlayFrontalCollisionForce(index, magnitudePercentage);

        say("LogiSWFrontalCollisionreturn is: " $ret);
}

exec function LogiSWGetNonLinearValue(int index, int inputValue)
{
        local int ret;
        ret = LogiGetNonLinearValue(index, inputValue);

        say("LogiSWGetNonLinearValue is: " $ret);
}

exec function LogiSWHasForceFeedback(int index)
{
        local bool ret;
        ret = LogiHasForceFeedback(index);

        say("LogiSWHasForceFeedback is: " $ret);
}


exec function LogiSWIsPlaying(int index, int forceType)
{
        local bool ret;
        ret = LogiIsPlaying(index, forceType);

        say("LogiSWFrontalCollisionreturn is: " $ret);
}

exec function LogiSWPlaySpringForce(int index,const int offsetPercentage, const int
saturationPercentage, const int coefficientPercentage)
{
        local bool ret;
        ret = LogiPlaySpringForce(index, offsetPercentage, saturationPercentage,
coefficientPercentage);

        say("LogiSWPlaySpringForceis: " $ret);
}

exec function LogiSWStopSpringForce(int index)
{
        local bool ret;
        ret = LogiStopSpringForce(index);

        say("LogiSWStopSpringForce is: " $ret);
}

exec function LogiSWPlayConstantForce(int index, int magnitudePercentage)
{
        local bool ret;
        ret = LogiPlayConstantForce(index, magnitudePercentage);
```

```
        say("LogiSWPlayConstantForce is: " $ret);
}

exec function LogiSWStopConstantForce(int index)
{
        local bool ret;
        ret = LogiStopConstantForce(index);

        say("LogiSWStopConstantForce is: " $ret);
}

exec function LogiSWPlayDamperForce(int index, int coefficientPercentage)
{
        local bool ret;
        ret = LogiPlayDamperForce(index, coefficientPercentage);

        say("LogiSWPlayDamperForce is: " $ret);
}

exec function LogiSWStopDamperForce(int index)
{
        local bool ret;
        ret = LogiStopDamperForce(index);

        say("LogiSWStopDamperForce is: " $ret);
}

exec function LogiSWPlaySideCollisionForce(int index, int magnitudePercentage)
{
        local bool ret;
        ret = LogiPlaySideCollisionForce(index, magnitudePercentage);

        say("LogiSWPlaySideCollisionForce is: " $ret);
}


exec function LogiSWPlayDirtRoadEffect(int index, int magnitudePercentage)
{
        local bool ret;
        ret = LogiPlayDirtRoadEffect(index, magnitudePercentage);

        say("LogiSWPlayDirtRoadEffectis: " $ret);
}

exec function LogiSWStopDirtRoadEffect(int index)
{
        local bool ret;
        ret = LogiStopDirtRoadEffect(index);

        say("LogiSWStopDirtRoadEffect is: " $ret);
}

exec function LogiSWPlayBumpyRoadEffect(int index, int magnitudePercentage)
{
        local bool ret;
        ret = LogiPlayBumpyRoadEffect(index, magnitudePercentage);

        say("LogiSWPlayBumpyRoadEffect is: " $ret);
}

exec function LogiSWStopBumpyRoadEffect(int index)
{
```

```
        local bool ret;
        ret = LogiStopBumpyRoadEffect(index);

        say("LogiSWStopBumpyRoadEffect is: " $ret);
}

exec function LogiSWPlaySlipperyRoadEffect(int index, int magnitudePercentage)
{
        local bool ret;
        ret = LogiPlaySlipperyRoadEffect(index, magnitudePercentage);

        say("LogiSWPlaySlipperyRoadEffect is: " $ret);
}

exec function LogiSWStopSlipperyRoadEffect(int index)
{
        local bool ret;
        ret = LogiStopSlipperyRoadEffect(index);

        say("LogiSWStopSlipperyRoadEffect is: " $ret);
}

exec function LogiSWPlaySurfaceEffect(int index, int type, int magnitudePercentage, int period)
{
        local bool ret;
        ret = LogiPlaySurfaceEffect(index,  type, magnitudePercentage, period);

        say("LogiSWPlaySurfaceEffect is: " $ret);
}

exec function LogiSWStopSurfaceEffect(int index)
{
        local bool ret;
        ret = LogiStopSurfaceEffect(index);

        say("LogiSWStopSurfaceEffect is: " $ret);
}

exec function LogiSWPlayCarAirborne(int index)
{
        local bool ret;
        ret = LogiPlayCarAirborne(index);

        say("LogiSWPlayCarAirborne is: " $ret);
}

exec function LogiSWStopCarAirborne(int index)
{
        local bool ret;
        ret = LogiStopCarAirborne(index);

        say("LogiSWStopCarAirborne is: " $ret);
}

exec function LogiSWPlaySoftstopForce(int index, int usableRangePercentage)
{
        local bool ret;
        ret = LogiPlaySoftstopForce(index, usableRangePercentage);

        say("LogiSWPlaySoftstopForce is: " $ret);
}
```

```
exec function LogiSWStopSoftstopForce(int index)
{
	local bool ret;
	ret = LogiStopSoftstopForce(index);

	say("LogiSWStopSoftstopForce is: " $ret);
}

exec function LogiSWSetExampleControllerProperties()
{
	local LogiControllerPropertiesData properties;
	local bool ret;
	properties.forceEnable = false;
	properties.overallGain = 0;
	properties.springGain = 0;
	properties.damperGain = 0;
	properties.defaultSpringEnabled = false;
	properties.defaultSpringGain = 0;
	properties.combinePedals = false;
	properties.wheelRange = 100;
	properties.gameSettingsEnabled = false;
	properties.allowGameSettings = false;

	ret = LogiSetPreferredControllerProperties(properties);
	say("LogiSWSetPreferredControllerProperties is: " $ret);

}


exec function LogiSWSetPreferredControllerProperties(LogiControllerPropertiesData properties)
{
	local bool ret;
	ret = LogiSetPreferredControllerProperties(properties);

	say("LogiSWSetPreferredControllerProperties is: " $ret);
}

exec function LogiSWGetCurrentControllerProperties(int index)
{
	local LogiControllerPropertiesData properties;
	LogiGetCurrentControllerProperties(index, properties);
	say("forceEnable is: " $(properties.forceEnable));
	say("overallGain is: " $(properties.overallGain));
	say("springGain is: " $(properties.springGain));
	say("damperGain is: " $(properties.damperGain));
	say("defaultSpringEnabled is: " $(properties.defaultSpringEnabled));
	say("combinePedals is: " $(properties.combinePedals));
	say("wheelRange is: " $(properties.wheelRange));
	say("gameSettingsEnabled is: " $(properties.gameSettingsEnabled));
	say("allowGameSettings is: " $(properties.allowGameSettings));


}

exec function LogiSWGetShifterMode(int index)
{
	local int ret;
	ret = LogiGetShifterMode(index);

	say("LogiSWGetShifterMode is: " $ret);
}
```

```
exec function LogiSWPlayLeds(int index, float currentRPM, float rpmFirstLedTurnsOn, float
rpmRedLine)
{
        local bool ret;
        ret = LogiPlayLeds(index, currentRPM, rpmFirstLedTurnsOn, rpmRedLine);

        say("LogiSWPlayLeds is: " $ret);
}

exec function LogiSWShutdown()
{       LogiSteeringShutdown();

        say("LogiSWShutdown is shutdown");
}
```

3. Copy Logitech SDK's Lib\x86\LogitechSteeringWheel.dll to UDK's Binaries\Win32\UserCode
4. Copy Logitech SDK's Lib\x64\LogitechSteeringWheel.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: `LogiSWInitialize`, and then hit <enter> key. You should see the "LogiSWInitialize return is: true" message.

Before trying other commands, make sure the LogiSWInitialize has returned true value. If it returns false, means your main window has not been initialized yet.

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

- `LogiSWUpdate`
- `LogiSWGetStateUDK`
- `LogiSWGetFriendlyProductName`
- `LogiSWIsConnected`
- `LogiSWIsDeviceConnected`
- `LogiSWIsManufacturerConnected`
- `LogiSWIsModelConnected`
- `LogiSWButtonTriggered`
- `LogiSWButtonReleased`
- `LogiSWButtonIsPressed`
- `LogiSWGenerateNonLinearValues`
- `LogiSWFrontalCollision`

- `LogiSWGetNonLinearValue`
- `LogiSWHasForceFeedback`
- `LogiSWIsPlaying`
- `LogiSWPlaySpringForce`
- `LogiSWStopSpringForce`
- `LogiSWPlayConstantForce`
- `LogiSWStopConstantForce`
- `LogiSWPlayDamperForce`
- `LogiSWStopDamperForce`
- `LogiSWPlaySideCollisionForce`
- `LogiSWPlayDirtRoadEffect`
- `LogiSWStopDirtRoadEffect`
- `LogiSWPlayBumpyRoadEffect`
- `LogiSWStopBumpyRoadEffect`
- `LogiSWPlaySlipperyRoadEffect`
- `LogiSWStopSlipperyRoadEffect`
- `LogiSWPlaySurfaceEffect`
- `LogiSWStopSurfaceEffect`
- `LogiSWPlayCarAirborne`
- `LogiSWStopCarAirborne`
- `LogiSWPlaySoftstopForce`
- `LogiSWStopSoftstopForce`
- `LogiSWSetExampleControllerProperties`
- `LogiSWSetPreferredControllerProperties`
- `LogiSWGetCurrentControllerProperties`
- `LogiSWGetShifterMode`
- `LogiSWPlayLeds`
- `LogiSWShutdown`

For example to start playing the dirt road effect on the first controller type :

- `LogiSWPlayDirtRoadEffect 0 x` (where x is the magnitude percentage)

Then hit <enter>

For questions/comments, email devtechsupport@logitech.com.