Developer side documentation

The legacy mode

The legacy mode (or virtual joystick) uses vjoy to translate the LEA inputs to standard HID joystick inputs. LEA use 4 virtual joysticks with 128 buttons, 8 axis and 4 POVs each. Keyboard keys are emulated by standard windows API.

The extended input mode

The extended input mode is a bidirectional system, that allows not only to gets the inputs from the LEA clients, but also to change dynamically the clients' parameters and controls.

Architecture

Game to clients data flow

The game is connected to the LEA server and send data to it via a TCP port on loopback. The server dispatch all the data to the clients that can use it via TCP/IP.

Clients to game data flow

The clients send the data to the server over TCP/IP which send it to the game via a TCP port on loopback. Depending on the selected configuration, the server may wait the game for asking the changes before sending to the game the complete change list (pull mode), or the server can send to the game the changes as soon as they are received from the clients (push mode). This is the game that request the connection mode while negotiating with the server.

Data encoding

Exchanged data is in UTF-8 XML. The format is:

```
<?xml version="1.0" encoding="UTF-8"?>

<command commandType=”62”>

        <EMData EMTag=”theEMTag1Button” EMValue=”True”/>

        <EMData EMTag=”theEMTag2Axis” EMValue=”0.5;0.6”/>

        <EMData EMTag=”theEMTag2POV” EMValue=”18000”/>

</command>
```

Note that the XML declaration is optional, and there is a random number of EMData elements, corresponding to the number of input changes. The EMTag is the string representing the game command. The EMValue is the new value of the command. The EMValue can be True or False for a button like command. For the axis the EMValue has a value as float between 0 and 1 following by the original value of the slider (not modified by the axis modifier), the two separated by a semi colon. You probably want to use the first float in your game. Finally the POV EMValue is an integer int degree x 100 (for example a POV at 180° is represented by 18000).

To send a pull request just send to the server:

<?xml version="1.0" encoding="UTF-8"?>

<command commandType=”61”/>

The XML declaration is optional too.

Using the SDK

License

The SDK is under a MIT license, but if you find a bug or an improvement we would be delighted if you kindly share it with the community on GitHub.

Description

The SDK contains all necessary functions to communicate with the server. It’s written in an asynchronous and event based way. Please note that the SDK is partially community maintained, but we check all versions before they are been deployed.

The files TCPLayerLite.cs and commands.cs are necessary to communicate with the server. The file Program.cs contains an example of configuration and communication with the server. The SDK compile in a program that install the sampleSDK game and project if not already installed and display in console the clients’ outputs.

You can download the LEA SDK on our website (http://www.pangosys.com/LEA/SDK.hml).

Support

Please ask us via our forum or ticket system if you need help, we will help you as fast as we can. If you found a bug please let us know, this way we can enhance our products and you can benefit a full performance experience. You can even propose an update on our SDK via our ticket system.

If you have any request concerning development of a specific control, please ask us. We will evaluate your request and might incorporate it in the next build.

Plans for future

For now, LEA is mostly a remote input device, but we plan to expend its capabilities. Actually, we are working on some radar implementations (2D sector/circular radar, 3D sector/circular radar) and some advanced inventory and actions systems.