OSVR – Unreal Engine 4 plugin

Installation

Follow this steps to integrate OSVR into your UE4 project:

1. install OSVR on your system
2. be sure that your project have at least one C++ file
3. close UE4Editor
4. copy the "*OSVR/*" plugin directory into "***<YourProject>****/Plugins/*"
5. copy all content of "***<YourProject>****Plugins**OSVR**ThirdParty**OSVR**bin*\" into "***<YourProject>****Binaries*\"
6. open you project and let UBT to recompile the OSVR plugin
7. under "*Window → Plugins*" ensure that OSVR is the only plugin of type "*Head Mounted Display*" enabled (e.g., be sure to have "*Oculus Rift Plugin*" and "*SteamVR Plugin*" both disabled)

Packaging

After having packaged your project:

1. copy all content of "***<YourProject>****Plugins**OSVR**ThirdParty**OSVR**bin*\" into "***<OutputDir>****WindowsNoEditor****<YourProject>****Binaries*\"

Blueprint API

APIs identify OSVR devices/resources using the same OSVR resource path notation.

Following UE4 input management, as general rule:

- state of continuous valued resources ("axis" inputs in UE4) is queried by polling, using functions defined in the **OSVR Blueprint Function Library**.
- state of discrete valued resources ("action" inputs in UE4) is reported by events. As UE4 currently does not support the definition of global events, OSVR events are wrapped in **OSVRInputComponent** and **OSVRActor** classes.

OSVRActor

Predefined Actor class containing an OSVRInputComponent. It can be used as a base class for custom Actors or can ben placed into a level to manage OSVR action events directly from the "*Level Blueprint*".

OSVRInputComponent

Component exposing OSVR action events. You can use it to bind custom actions to OSVR action events like buttons press, etc.

```
void OnPositionChanged (FName name, FVector position);
```

Fired when the position of an OSVR resource changes.

Outputs:

- `name`: OSVR resource path;
- `position`: the new position.

`void` **`OnOrientationChanged`** `(FName name, FRotator orientation);`

Fired when the orientation of an OSVR resource changes.

Outputs:

- `name`: OSVR resource path;
- `position`: the new orientation.

`void` **`OnButtonStateChanged`** `(FName name, EButtonState::Type state);`

Fired when the state of a button of an OSVR resource changes.

Outputs:

- `name`: OSVR resource path;
- `state`: the new button state `(PRESSED, NOT_PRESSED)`.

`void` **`OnAnalogValueChanged`** `(FName name, float value);`

Fired when the value of a single-valued analog resource changes (e.g. a gamepad trigger analog control).

Outputs:

- `name`: OSVR resource path;
- `value`: the new value, in [0, 1].

Blueprint function library

`static FVector` **`GetInterfacePosition(FName`** `Name);`

Return the current position of an OSVR resource.

Inputs:

- `name`: OSVR resource path.

`static FRotator` **`GetInterfaceRotation(FName`** `Name);`

Return the current orientation of an OSVR resource.

Inputs:

- `name`: OSVR resource path.

`static EButtonState::Type` **`GetInterfaceButtonState`**`(FName Name);`

Return the current state of an OSVR button.

Inputs:

- `name`: OSVR resource path.

`static float` **`GetInterfaceAnalogValue`**`(FName Name);`

Return the current value of a single-valued OSVR analog resource (e.g. a gamepad trigger analog control).

Inputs:

- `name`: OSVR resource path.

`static void` **`SetCurrentHmdOrientationAndPositionAsBase`**`();`

Set the current position and orientation of OSVR HMD device as "*rest*" position.