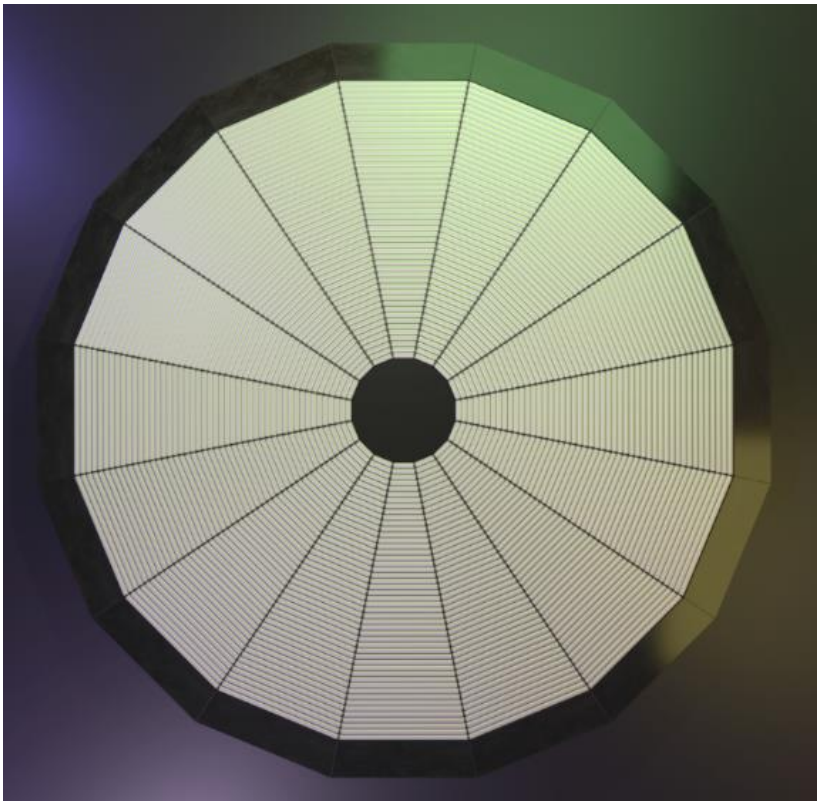
Controlling applications using the Omnideck

Omnifinity



Introduction

This document outlines the various ways you can control an application using the Omnideck. For details on each mode please contact us.

History

Version	Date	Notice
V1.17.5-Beta	2018-05-07	Finally updated API to use velocity vector instead of position vector. Added documentation about the API lifecycle and information flow.
V1.17.4-Beta	2018-03-05	Performed split of API and implementation. Added some example scenes.

Contents

Different ways to control the Omnideck

We can provide various ways to control a game. The prefered solution is to use our Omnideck API which is tailored for game engines.

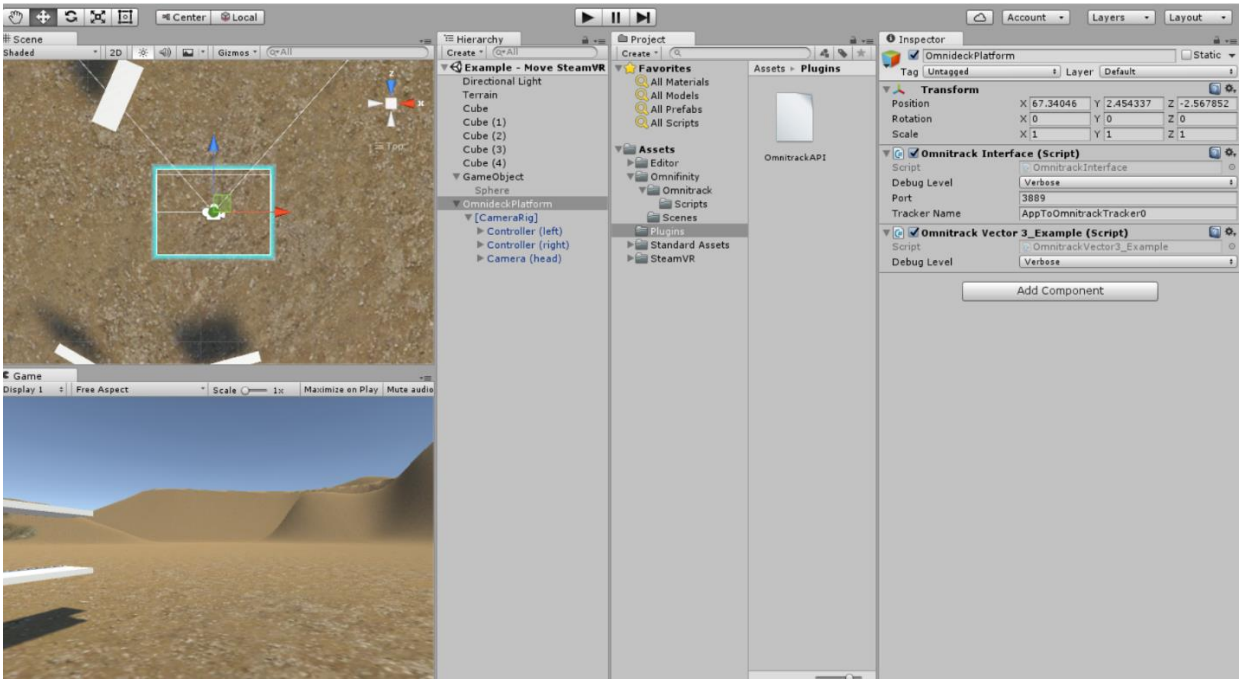
Method	What	Integration quality level	Comment
A	Native integration using the Omnideck API	3	Available today: Unity API Available soon: Unreal API
B1 B2	B1 Touchpad emulation B2 Touchpad control with custom Omnideck controller driver	1 2	Available today: HTC Vive Omnideck driver
C	Custom integration with a specific application	Depends on functionality of application	
D	Keyboard and mouse	0	An emulator exists but this is not a recommended solution

Method A – Native integration using the Omnideck API

We provide an API for popular game engines allowing you to get best-in-class integration of the Omnideck in your application.

The Unity API can be freely downloaded from github:
https://github.com/Omnifinity/Unity-SteamVR-API

We currently use the HTC Vive system as our prefered Tracking/VR-system. If you use another Tracking/VR-system please contact us and we can help you with the integration.



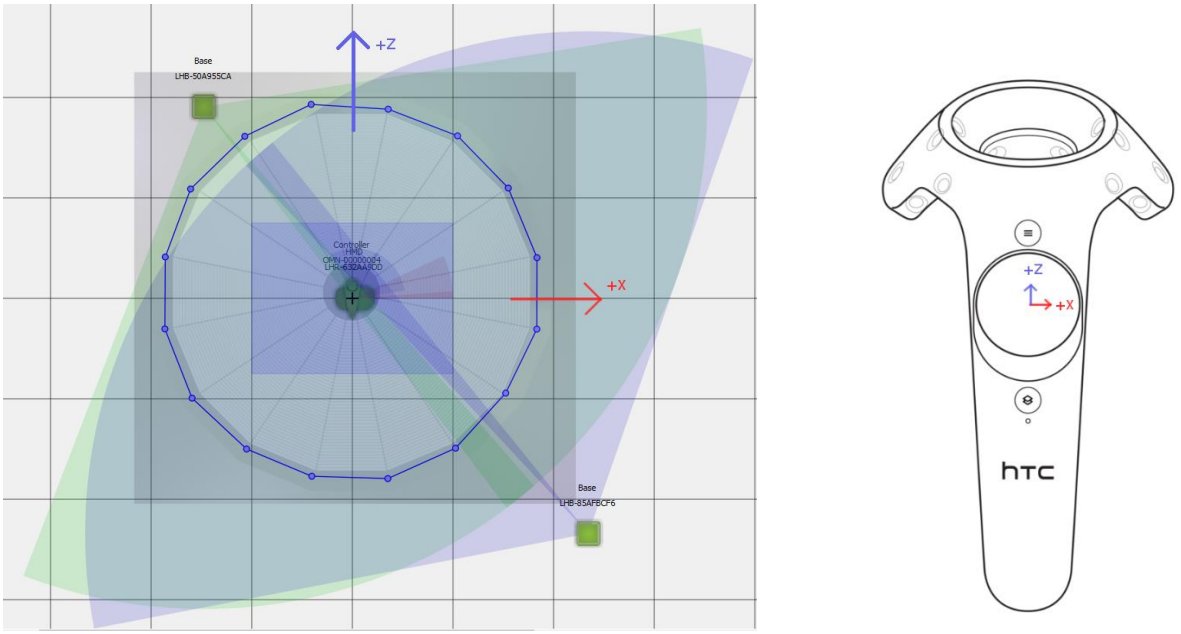
Figur 1 Unity API implementation example

Method B1 – Touchpad emulation

Note: Beta/experimental feature. This can break at any point of time.

We can provide a beta-driver for the HTC Vive system using touchpad emulation to control applications that already support touchpad locomotion today.

This is an experimental feature that can be used as an initial step to try your application on the Omnideck – hopefully with little overhead. Any serious implementations should however rely on Method A.



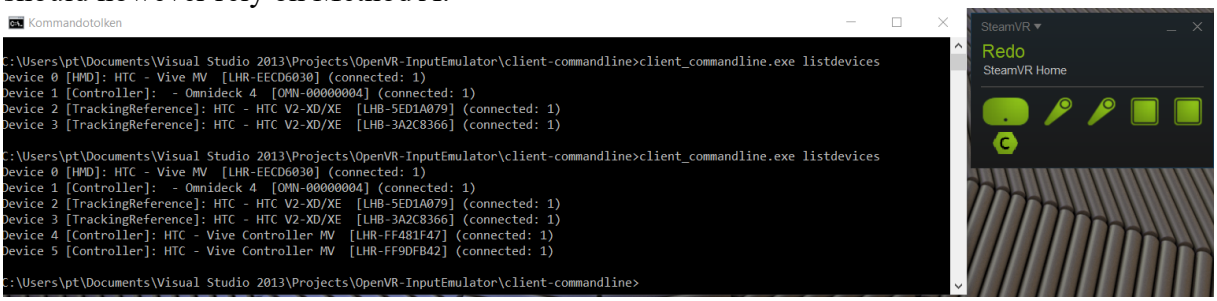
Figur 2 Overview of the HTC Vive tracking space and chaperone bounds

Method B2 – Touchpad control with custom Omnideck controller driver

Note: Beta/experimental feature. This can break at any point of time.

We can provide a beta-driver for the HTC Vive system that creates a new Omnideck controller device that sends out touchpad events. You can fetch this data in Unity or Unreal and control the movement of your character/camera. The implementation is similar to the path you would take when adding support for e.g. the Vive Tracker (https://www.vive.com/us/vive-tracker/).

This is an experimental feature that can be used as an initial step to try your application on the Omnideck with a bit more overhead compared to Method B1. Any serious implementations should however rely on Method A.



Figur 3 Custom HTC Vive driver for the Omnideck

Method C – Custom integration with a specific application

We can provide you with details on data formats to make it possible to control your application using the Omnideck.

Method D – Keyboard and mouse

Although we can provide emulators for keyboard movement (WASD) we do not recommend this discrete method of controlling any serious VR application since it is incompatible with how humans locomote in general.

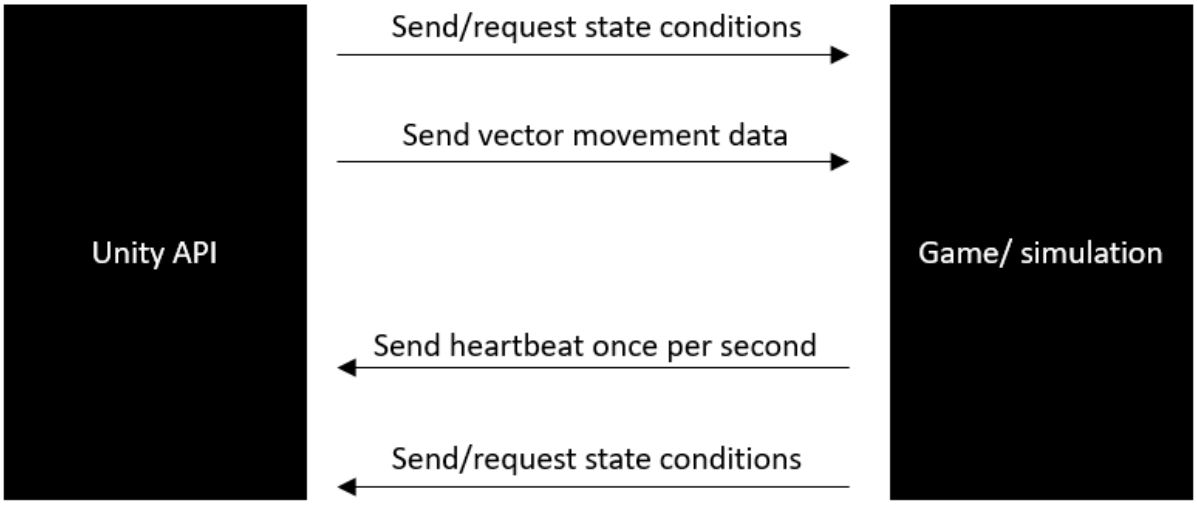
The lifecycle of your simulation in conjunction with our API

The example scenes will show you how to start working with our API.

When you have implemented our API and your simulation is running there are few things happening under the hood inside our API:

- Your simulation will tell Omnitrack when it has started/stopped – look inside the Start()- and OnApplicationQuit() methods to find out the details.
- Each frame (based on the framerate of the tracking data) the simulation will request/receive character movement Vector3 data ([m/s]) for your usage – look inside the AcquireTrackingData() method.
- Each second the simulation will send a hearbeat telling Omnitrack that it is running – look inside the SendHearbeat() method.

Based on the user's movement on the Omnideck a character movement Vector3 data is calculated for your usage. It is your responsibility to move a character around in your scene and take into account whatever ground/wall collision routines you work with.



Figur 4 Information stream between the Unity API and your simulation