Omnifinity Unity API - Beta

Version: 1.15.12-Beta

HTC Vive Edition

Nomenclature

Term	**Description**
Omnifinity	Solution provider for natural movement in the virtual world
Omnideck 6	Active omnidirectional treadmill
Omnitrack	Software to control the speed of the Omnideck 6 and to send tracking data and states of the human to a game or simulator
Tracker	A object tracked in 6 DOF (XYZYPR)
Sensor	An Tracker object that also have ”input-button” states.
HMD	Head-mounted display such as Gear VR
Unity 3D	Game / simulation development environment

Introduction

This document describes the *Omnifinity Unity API Beta* which allows you to integrate your game/simulation to work with the Omnideck. This document focuses on integration with the *HTC Vive* although we've left a fair amount of previous explanations on how optical tracking systems work with the Omnideck.

The archive comes with a stripped down version of our main Omnitrack software called *OmnitrackGameClient.ex.* This (console-) program handles communication between the HTC Vive tracking system and your game and should be running at all times when your game is running.

Below is a schematic view of a fully fledged Omnitrack system side to side with how the Unity API and your game fits in.

When your game is run on a complete Omnideck system the movement of your body and head will determine the location and viewing orientation of your viewpoint. You have full freedom of movement - if you look to the left while walking forward you will walk past any obstacles next to your shoulder. If you lean your body down or look up to the sky – so does your character. If you physically stop moving so does your character.

There is no support for WASD- or mousebased movement in this software package.

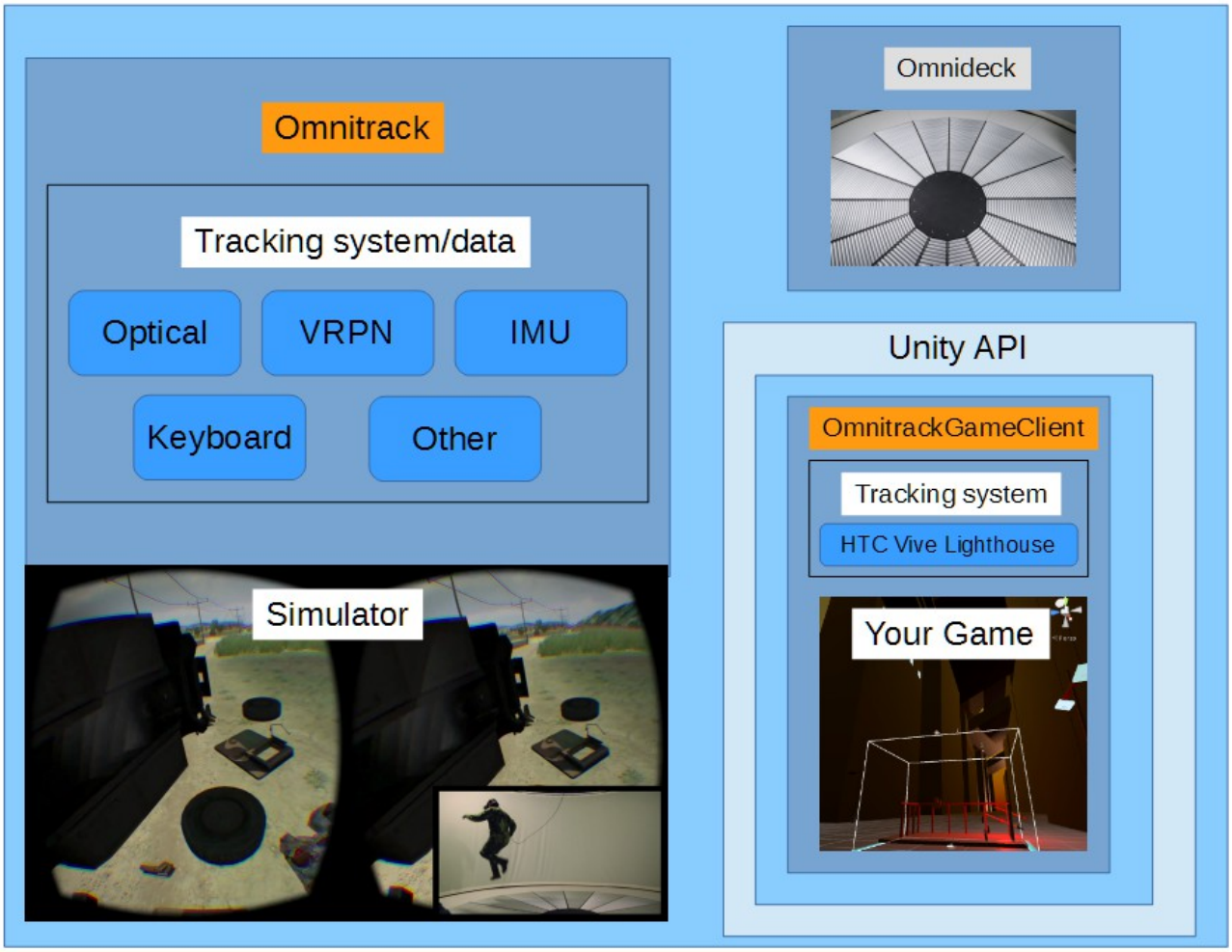

Illustration 1: System view of Omnitrack and the Unity API

How tracking of the human works on the Omnideck

Our recommended tracking method for full body tracking is based on infrared light reflective markers, continously tracked by an optical tracking camera system, see image below. This is a rather expensive setup compared to the HTC Vive but the explanation of how this works is included in this document since it is a good introduction to how motion is tracked and translated on the Omnideck.

To control the speed of the Omnideck 6 and the position and viewpoint of the human in the simulation the simplest tracker-type should use a single head-mounted tracker mounted on a suitable location on a head-mounted display. If you use a projection system or a CAVE the marker is usually placed on a baseball cap or on the glasses driving the viewpoint in the CAVE setup. If you use the HTC Vive this information will come from position of the HMD.

A setup with only one marker attached to the head allows for unlimited 6 DOF position tracking of the viewpoint in the simulation. Here, the movement of the body and the head are ”semi”-disconnected, meaning that the position and rotation of the body and the head are the same. However, the movement velocity of the body is not at all connected to the viewing direction in the HMD (as common in most traditional games with WASD-input). This means that when you walk around in the simulation the position of your avatar matches the movement of your body while the viewing direction matches the one given by your head – just as it works in real life.

You can easily expand this setup and mount an additional marker on the human – strategically placed on the hip/back – to end up with a completely disconnected head-body setup. Thus you can track the position and rotation of the body and head independently. This can be extended further to allow for full body motion tracking with a 1-to-1 mapping between the physical- and virtual you.

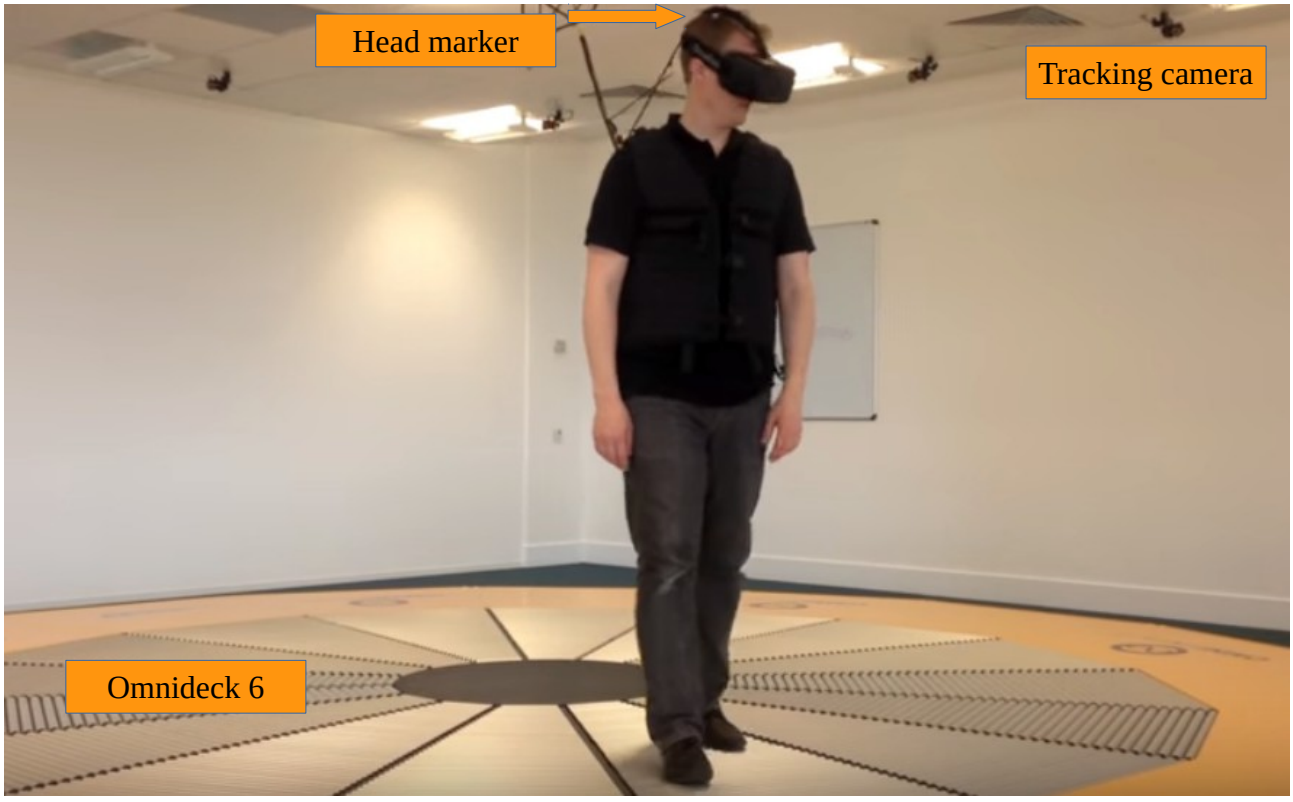

Illustration 2: User walking around on the Omnideck 6

Contents of the API

The Unity API is based around tracker data communication between the OmnideckGameClient.exe and a Unity Character Controller (which itself encapsulates the HTC Vive [CameraRig]). This allows for collision detection between objects and ground by default.

This version of the API uses a custom UDP-datapacket carrying various tracker data (described in source code only at the moment).
We aim to expand this and migrate toward efforts such as using VRPN included in the OSVR as time goes by.

Unity Scene

Inside the Unity-package you will find one (1) example scene. The actual code base (written in C#) is located in the Scripts-folder and the communication between the Omnideck and Unity. The example scenes is provided for Unity 5.3.4f1.

Scene 1: Omnifinity_InterfaceSample

In the example scene called *"Omnifinity_InterfaceSample"* you will find a Game Object called *OmnitrackCharacterController* (you will find a prefab with the same name in the Prefabs folder). This game objects is key to integrating support for the Omnideck in your game.

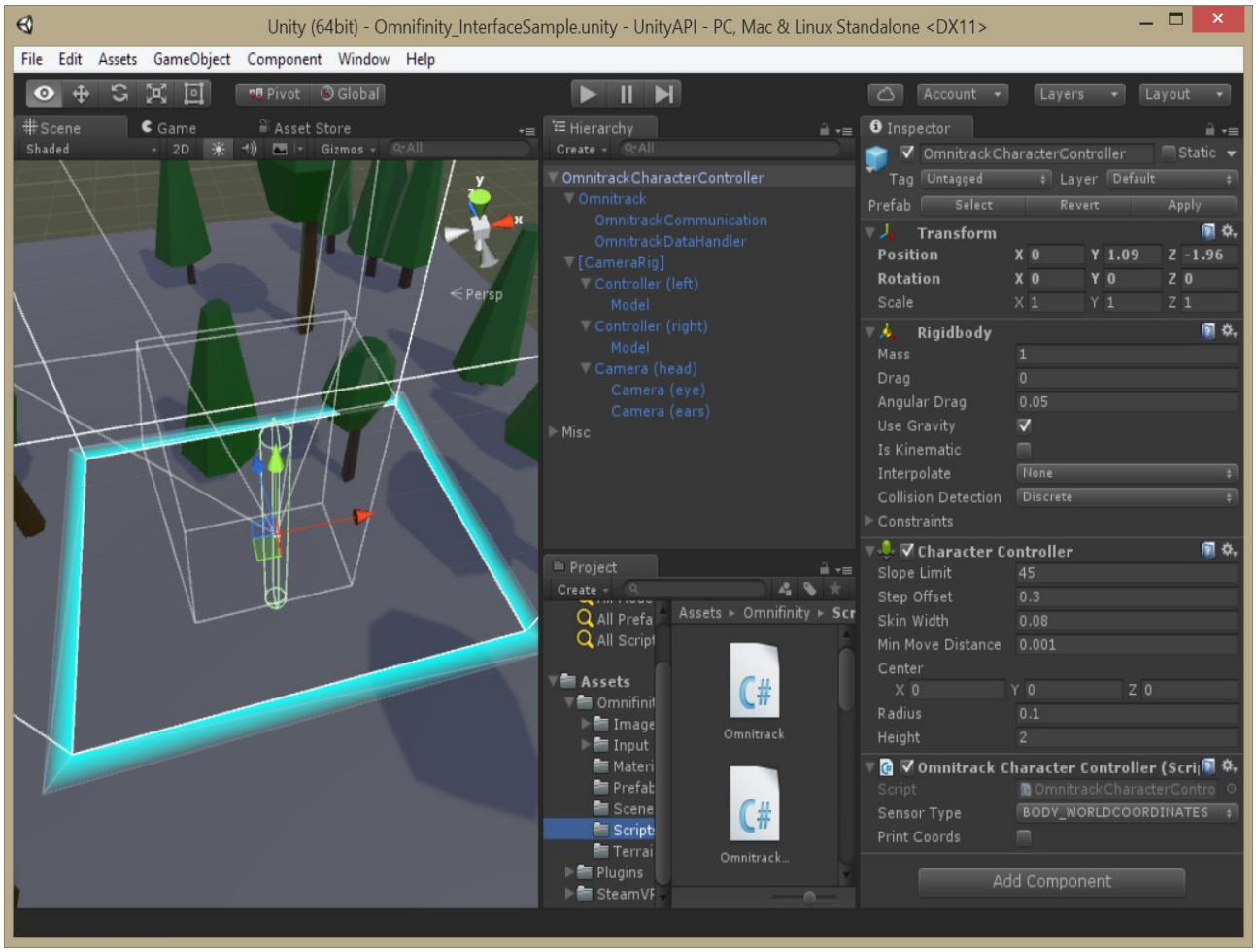

Illustration 3: OmnitrackCharacterController with one Omnitrack and one HTC Vive [CameraRig] child game object

The *OmnitrackCharacterController* game object encapsulates two child objects:

1. *Omnitrack*

   Contains three scripts:

   - *Omnitrack.cs*

     Contains public methods for acquiring things like the position and rotation of tracked objects (e.g. the HTC Vive HMD and the two hand controllers).

   - *OmnitrackCommunication.cs*
     Receives tracking data from the Omnideck on a specific port over UDP (default port: 11016)

   - *OmnitrackDataHandler.cs*
     Creates neccesairy links between the *Omnitrack*-gameobject/component and the tracked sensors.

2. *[CameraRig]*
   The HTC Vive Camera rig that comes with Steam VR. Unmodified.

OmnitrackGameClient.exe

An executable called *OmnitrackGameClient.exe* is included in the archive – this is a simplified version of the software the Omnideck uses in a normal production environment. You will use it to

1. send tracking data from your HTC Vive setup to drive the movement of the OmnitrackCharacterController (--usehandcontrollerasbody 1)
2. or send simulated tracking data making your player move around in a circular pattern (--sendcircularpathtrackingdata 1)

Run the program in a Command prompt without any arguments to get a list of valid parameters.

How to integrate your game with the Omnideck 6

There are a few simple steps you need to go through in order to create support for the Omnideck in your game or simulation.

Extending a Character Controller

The simplest way to allow movement of the HTC Vive playfield is to encapsulate it inside a standard unity Character Controller. Please read on how this is done below.

1. Import SteamVR

Using the Asset store download and install the latest SteamVR API.

2. Import the Omnifinity Unity API Package

Import the Omnifinity Unity API Package by either doubleclicking on the appropriate Omnitrack .unitypackage-file that you've just downloaded, or use the regular menu item selection:

Assets > Import Package > Custom Package

Navigate to *Omnifinity > Prefabs a*nd drag the *OmnitrackCharacterController* prefab into your scene. This prefab references public methods in Omnitrack.cs, namely:

public Vector3 getBodyWorldPosition() { … }

public Vector3 getBodyWorldVelocity() { ... }

public Quaternion getBodyWorldQuaternion() { ... }

public Vector3 getHeadPosition() { … }

public Vector3 getLeftHandPosition() { … }

...

Use the ***getBodyWorldPosition()*** or ***getBodyWorldVelocity()*** – to get the body's world X-/Z-/Y-position or velocity and use it for moving the implementation of your game character. This is in [m] or [m/s] respectively.

If you look in the *Update()* method in OmnitrackCharacterController.cs you'll see calls to *MoveObject().* This method acquires the speed of a sensor (e.g. hand controller) and moves the Character Controller using a call to *CharacterController.SimpleMove()* based on a supplied velocity. This allows for collosion detection with objects.

3 Test if data communication with the Omnideck works

This is done using the *OmnitrackGameClient.exe* program contained in the archive. This console program should be run on your game computer. There is a batch-file to help you – please adapt it to your preference. Note that you need to specify the same IP as your computer (Note: do not use

'localhost' at the moment).

If you run the program without any parameters you will see some typical options that can change.

When your Game and the OmnitrackGameClient is running, if you **press and hold** the grip button and move it around you should see a slight movement of your camera in the world (its about 1/10th of the movement of your hand). If the camera moves around you know that your game will run with the Omnideck. Your final integration should have this movement enabled by default and not only with the grip button. It is there so that you can enable/disable Omnitrack support using the handcontroller.

If you **press and hold** the trigger button the character controller will roam freely on the ground.

Note: If you are already using port 11016 for something else you need to change the *"Omnitrack Data Port"* number in the DataSimulator.exe argument list - as well as in the receiving dataframe dataobject in Unity.

Troubleshooting

- Check your firewall settings and make sure you have opened the correct ports

Hacking on your own

We know that some or you know your way around the code and will easily port this to other simulators and game engines – infact we highly encourage you to do so! And while you are at it – why not share your progress with us?

Release notes

V1.15.12 Beta

Features

- Added support for the HTC Vive by default.
- Oculus support is deprecated until the Oculus Touch system is released.

Bugfixes

- None

Known issues

- None

V1.15.8 Beta

Features

- Fixed bug causing position and orientation of sensor objects to not update if the user has disabled *isPositionEnabled* and *isRotationEnabled* booleans.
- Added prefab *OmnitrackSimple* that does not interact with a Unity Character Controller. Use this prefab to extract sensor data if you already have your own Character Controller.
- Added more debugging output for main Omnitrack.cs script.

Bugfixes

- None

Known issues

- Same as in V1.15.5 Beta

V1.15.7 Beta

Features

- Enabled public access to manually define important data-, communication and player game objects through GUI drag-and-drop. Now one can either define a Tag on those objects or drag-and-drop the objects in question to create the required links. See documentation for explanation.
- Added more debugging output.

Bugfixes

- None

Known issues

- Same as in V1.15.5 Beta

V1.15.6 Beta

Features

- Fixed an issue with the VRPN data.
- Restructured documentation layout and added some images to clarify how markers are mounted and how the integration of Omnitrack in Unity looks like.

Bugfixes

- None

Known issues

- Same as in V1.15.5 Beta

V1.15.5 Beta

Features

- Total restructuring allowing for easy access to tracking data of each sensor through public methods in *Omnitrack.cs.*

Bugfixes

- None

Known issues

- Unfinished documentation on how to add Oculus Rift support.
- Same as in V1.15.1 Beta

V1.15.4 Beta

Features

- Added an *OmnitrackInterface* class enabling simple and direct access to the movement vector and rotation quaternion of a tracker. Simplifies implementation of Omnideck movement in existing games.

Bugfixes

- None

Known issues

- Same as in V1.15.1 Beta

V1.15.3 Beta

Features

- VRPN is now supported and you can input two trackers, one controlling the head/body and one controlling the left hand.

Bugfixes

- None

Known issues

- Same as in V1.15.1 Beta

V1.15.2 Beta

Internal release, not publically communicated.

V1.15.1 Beta

Bugfixes

- <cut>

Known issues

- Public support for other protocols such as VRPN is in the works.
- When not using the Oculus Rift Camera, the orientation of the DK2 drifts over time. The current solution is to look towards north and the external tracking system will reset the heading at that fixed angle. The code to account for this is however not included in this package but will be made available to you when you want to test your simulation on an actual Omnideck system.