Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

In-Situ Virtual Reality

Luiz Velho
Leo Carvalho
Djalma Lucio

Technical Report TR-17-03 Relatório Técnico

March - 2017 - Março

In-Situ VR

Luiz Velho Djalma Lucio Leo Carvalho

***Abstract*—In this techinical report, we describe the environment at the VISGRAF Laboratory for developing multi-user virtual reality experiments.**

I. Introduction

In-Situ VR is a creative environment for the development of shared virtual reality applications. This hardware and software infra-structure implements multi-user interaction, full positional tracking of the players, and objects in order to create a complete sense of immersion in a tangible space. In this way, it is possible to bridge the gap between the physical and virtual spaces in VR applications.

This research is part of an ongoing collaboration with the Future Reality Lab of NYU and has its foundations on the Holojam Platform. Holojam started as a multi-user interactive painting experience at the VR Village of SIGGRAPH 2015.

II. Holojam

The Holojam VR Environment is shared-space virtual reality platform developed by the Future Reality Lab of New York University under the leadership of Prof. Ken Perlin [1].

This platform enables content creators to build complex location-based multiplayer VR experiences in a simple, unified Unity project. The development framework provides an extensible and clean interface, allowing for rapid prototyping and extension. Additionally, it abstracts away specific VR hardware, promoting a flexible and customizable creation of virtual reality experiences.

In this section we describe the main aspects of Holojam.

A. Server-Client Architecture

Holojam has a server-client architecture composed of two main components: the *Holojam Server* and the *Holojam SDK*.

1) Holojam Server: is a C++ program that receives and sends tracking data. The server connects with clients through either multicast or unicast. When starting the server, define the IP address of the network interface that will communicate through multicast, and also the IP address of the interface to communicate through unicast. In the VISGRAF Lab , the server runs in a computer with two network interfaces: the wireless interface (IP 192.168.0.104), used by default for Multicast; and the ethernet interface (IP 147.65.6.153), used by default for Unicast. Multicast communication is sent by address 224.1.1.1:1611. See Fig. 1

A unicast link can be established by:

- File ips.txt, where each row contains the IP of a client
- Typing u in the server interface (also added to ips.txt)
- Pinging server in a client (it is not added to ips.txt).

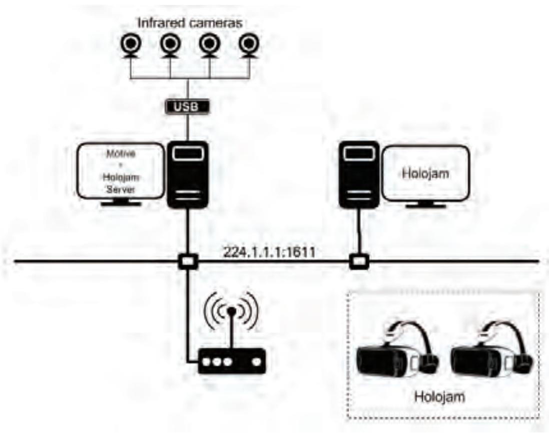

Fig. 1. VR Network environment at VISGRAF Lab .

Holojam Server uses Google's "Protocol Buffers" to serialize the data, making it usable across programming languages and platforms. An *Update* is a data structure storing a set of rigid bodies at a certain moment, where each tracked rigid body information is defined using a data structure called *LiveObject*. The server sends and receives serialized Updates through the network. There is a label for each Update, that can be used to identify what is the origin of this data. If the label is "ping", a unicast is created between the server and the sending client.

Data from Optitrack Motive (see Section II-B1) is received through an embedded NatNet Client. The mocap data is converted to one or more Update data structures and sent to clients with label "motive".

2) Holojam SDK: The SDK contains a Unity project with a sample scene that connects to the Holojam server and shows the tracked objects in a 3D virtual room. Some objects are ready to be used: some faces, wands, a pair of hands, a pair of feet, a cube, a table, and a device (laptop). Each one of these is a Unity object with a component called HolojamView.

When the scene is running and there is a Holojam Server sending mocap data, the position and orientation of each HolojamView object are adjusted if the server sends information about this object.

Each HolojamView object must have a label equal to one of the following: VRn, VRn_lefthand, VRn_righthand, VRn_leftankle, VRn_rightankle, VRn_wand, with $n = [1, 4]$, and VR_box, VR_laptop, VR_table, vive, vive_controller_left, vive_controller_right.

B. Tracking

The physical position and orientation of the real elements in a Holojam scene can be obtained either through a Mocap System, such as Optitrack Motive, or through a high-end VR Headset, such as the HTC Vive. As described in Section **??** they send rigid body tracking data to the Holojam Server so that it can be broadcasted to the Unity Holojam Clients.

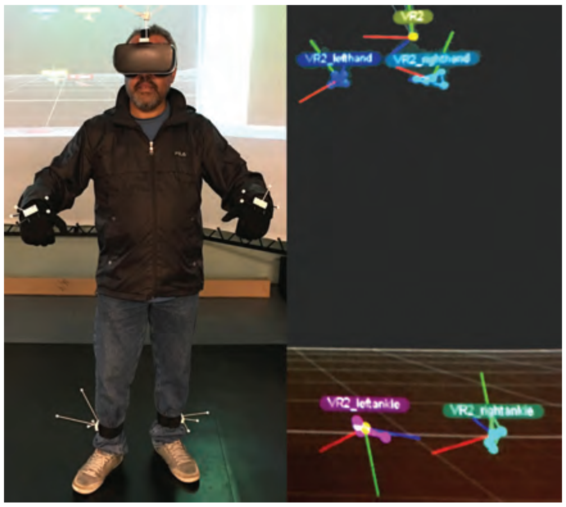

Fig. 2. Real Player with rigid body markers tracked by Optitrack Motive.

1) Optitrack Motive: For each rigid body to be tracked there must be at least 3 markers, positioned in a unique asymmetric way. It is recommended to use some redundant markers to avoid problems with occluded markers and to improve the data precision during tracking.

Each tracked rigid body must be labelled accordingly to Holojam name conventions. For example VR1 for headset of actor 1, VR1_righthand for the right hand of actor 1, VR2_leftankle for the left foot of actor 2, VR1_wand for the first wand, etc. See Fig 2.

Optitrack can be configured to send the mocap data through the network using NatNet SDK [2].

2) HTC Vive: Employs the SteamVR Lighthouse System for tracking the Vive Headset, Controllers and Tracker Accessory [3]. This system uses the so-called Base Stations that sweep laser beams in the environment to guide triangulation performed in each movable element. The rigid body tracking data is available to the Unity Client and sent by the SDK to the Holojam Server.

III. Unity Development Framework

The authoring of a VR experience with Holojam is done in Unity [4] using the components of the Holojam SDK.

There are three important aspects to consider for developers, namely: the Holojam Unity objects and prefabs; the Virtual Actor support; and the Holojam Viewer.

A. Concepts

The Holojam Unity objects implement the support for multiplayer VR, that includes communication with the Holojam Server, location-aware components, and Avatar presence.

1) Server: This object features a script component called ”Holojam Network”, which contains some fields indicating how to communicate with the server.

Field ”Multicast Address” indicates the IP address used to receive data from the server. It is not relevant if there is already a unicast connection between this client and the server (as long as it is a valid multicast address), because in that case the client will receive the data directly from the server.

Field ”Server address” can be used to indicate directly the IP of the server. This address is used to send data to the server, but if the indicated address is not local (not starting with 192) Holojam will ping this address, which creates a unicast connection between the server and this client. All the data that are sent and received use the Update data structure. When Holojam sends an object, it uses the label ”SendData” for the Update.

2) Space: The physical available space is mapped to the virtual space through the Holobounds. This feature provides the player with a feedback of the space limits by showing a virtual grid when a wall is near. See Fig. 3.

Holobounds object indicates the boundary of the tracking room. This boundary can be defined using one HolojamView object as a calibrator. During runtime, put the calibrator at one of the four corners of your room and press the button labelled ”C” to define the xy position of that corner. Repeat the process for each corner. It can also be done to calibrate the z position of the floor and of the ceiling.

The script component called Fence is responsible for showing a fence when the build actor gets far from the boundaries of the tracking room.

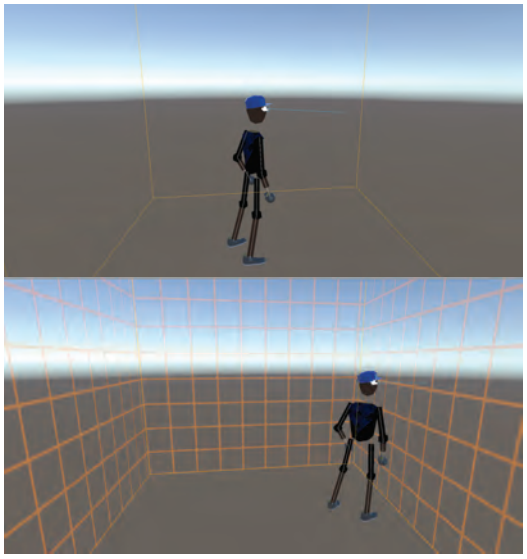

Fig. 3. Holobounds and Fence

3) Wands: This object has three Stylus objects. Each Stylus represents one wand object sent by Holojam Server. There is a script named Stylus, that is used to draw lines in the space using the wand. This works if the wand has buttons, like a

Wiimote, where button ”A” draws lines, and button ”B” erases the lines. See Fig. 4

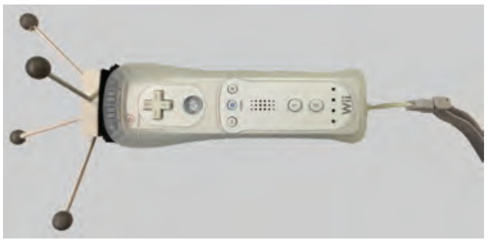
Fig. 4. Wand based on Wiimote.

4) Trackables: Contains other objects that can be included in the scene, also coming from tracked objects. The SDK demo scene includes a Cube, a Table and a Dev (laptop) object. See Fig. 5.

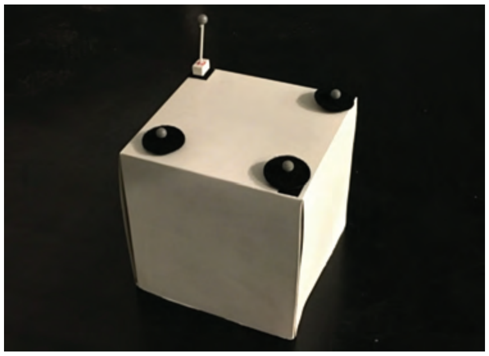
Fig. 5. Trackable Cube Object.

5) Synchronizables: are virtual objects that can be sent from one computer to others running Holojam. They can be used, for example, to exchange messages. The SDK contains three objects that are Synchronizables: SyncedCube, HoloTime and Messenger. All of them contain a script component where it is possible to define the object label, a flag ”Use Master PC”, that should be disabled when working on mobile devices, and a flag ”Sending” that should only be set true in one device running Holojam, because it will be sent to the server and then received by other devices running Holojam.

When the position or rotation of a sending object is changed, this object will move accordingly in any other device that includes an object with the same label.

HoloTime object contains a Time property, which is continuously increasing during runtime.

Messenger object contains a text input, that can be used to send text to other devices. The messages are shown in the scene with a Text Mesh object.

B. Actors

The virtual player is created from tracked reference markers on the real player using inverse kinematics. The Actor Manager reconstructs the full body of the Actor from pairables elements corresponding to hands, feet an head.

1) ActorManager: object contains a script to add or remove actors, and a script to define the ”build Tag”, that indicates which HolojamView is the main object of the scene. The main camera will be positioned and oriented according to the build tag object, and this object will not be rendered.

2) Actor Controller: The ActorManager object also has child objects that correspond to the actors present in the scene. Each actor has an Actor Controller component that can be used to define a label, the Tag (HEADSET1, HEADSET2, etc) and the color used for that actor.

3) Pairable: object contains four objects: two for hands and the other two for feet, where each has a script component called Pairable, that is used to create a connection between the hand or foot with an actor.

To pair hands and feet to an actor, bring each hand/foot close to the head, after a time (according to the value defined in ”Pair Time” parameter) the pairing is set, and the corresponding limb is rendered. It is important to pair the right hand before pairing the left hand, and pair the right foot before the left foot. Bring a hand or foot close to the head again to unpair it.

4) Inverse Kinematics: is done by a component in PairTarget object (inside each Actor in the scene). To simplify the pairing process, we, on VISGRAF Lab , added the script named Fixed Pairables that can be used to pre-define the pairables (hands and feet) for this actor. Each pairable will be automatically paired with this actor as long as it is tracked. If Fixed Pairables is enabled, the Sphere Collider will be disabled to avoid undesired pairings or unpairings.

C. VR Camera

The visualization of the Player’s point of view is done through the VR Camera. It is associated with each real player and its Avatar.

1) Viewer: object contains a script that controls the main camera of the scene. The position of the viewer is defined by the position of the actor with the tag ”build”. The orientation of the viewer can be configured to use the orientation of the build actor, or to use the IMU from other devices, like Oculus or Gear VR. To control this, change the field ”Tracking Type” with one of the three options:

- Optical: the actor orientation is used for the viewer
- Legacy: the optical orientation is used with a smooth interpolation
- IMU: the orientation from device sensors is used (inertial measurement unit). Wrong orientations are avoided by a frequent adjustment to the optical orientation.

2) NewViewer: In Viewer Script, when the TrackingType is set to IMU, the system uses the orientation given by the HMD device, and it is corrected when it is too discrepant from the orientation from the optical data. This causes some unpleasant shifts during runtime. We created a simpler scheme to use IMU and optical data with a script called NewViewer. In this script we can select the tracking type to IMU or OPTICAL. When using OPTICAL, it behaves like the old Viewer Script, discarding the IMU orientation, and using only the optical data. When IMU is set, the system uses the orientation given by the HMD device, but at every frame the camera is slightly pushed into the direction of the optical orientation. This way, the orientation never deviates from the optical orientation, and it is smooth like the pure IMU orientation. Also, the NewViewer Script can send back to the server the final position and orientation of the camera, as another HolojamView object, using Synchronizable objects. It is only necessary to inform the label of input object (from where the optical information is taken), and the label of the output object (with the real position and orientation of the camera).

3) VR Camera: This is the main camera of the scene. It contains another object called VR Console, that is useful to show some messages to the viewer.

D. Depoyment

The application can be built in several devices, including smartphones. Change the target platform in *File-Build* Settings. If you change it to Android, connect the phone with USB port and then click ”Build and Run”. The application will be compiled and sent to the phone.

If you want to use a headset like Oculus or GearVR, then activate option ”Virtual Reality Supported” on *Edit-Project_Settings-Player*, and also set Tracking Type to ”IMU” in Viewer object.

The HTC Vive requieres the Steam software, as well as the SteamVR Plugin for Unity.

IV. In Situ VR Environment

The In Situ VR environment at the VISGRAF Lab provides an extensive infrastructure for research and development of projects related to new media. Among other facilities, it integrates two separate lab spaces for Situated VR. Currently, one space houses an Optitrack Motion Capture System from Natural Motion Inc [2], while the other space accomodates an HTC Vive play area.

A. Motion Capture Stage

The Optitrack System is composed of 12 cameras and the Motive Tracker and Body, a software suite for rigid and articulated body tracking. See Fig. 6.

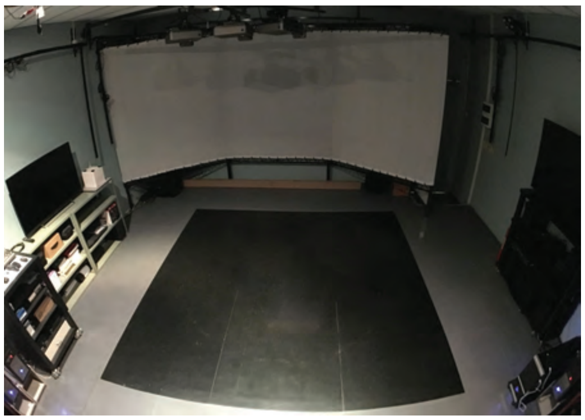

Fig. 6. Motion Capture Stage with Optitrack System..

B. VR Equipment

The virtual reality equipment available in the Lab is very complete and includes most of the existing technologies currently in the market.

Since this area advances at rapid pace, our goal is to test and evaluate the pros and cons of different options in order to determine the best solutions.

In terms of Head Mounted Displays (HMDs), we are experimenting with the following equipment: The Google Cardboard and Daydream [5]; the Samsung GearVR [6]; the Oculus Rift [7]; and the HTC Vive [3].

Except for the HTC Vive that uses the Lighthouse tracking, all other Headsets have their position tracked by the Optitrack/Motive system.

C. Holojam Accessories

The real players and objects participating in a Holojam system need to have their position and orientation tracked and sent to the Holojam Server.

We have designed special rigid body markers to be used in our MoCap Stage for the Holojam platform. They include markers for the Headsets, hands, and ankels of the players, the Wands, and also a Cube and a Table. See Figure 7

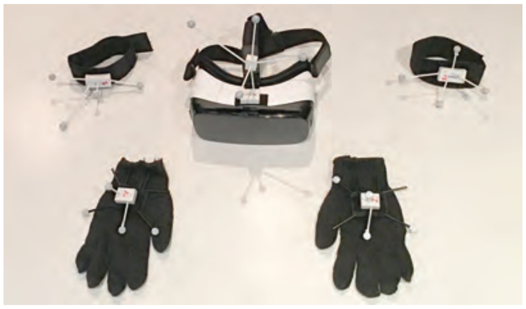

Fig. 7. Holojam Objects with Markers

V. Conclusion

We described the infra-structure at the VISGRAF Laboratory for the development of situated participatory virtual reality applications. This kind of experience creates a multi-user, interactive location-aware shared space where players engage in immersive mixed reality with a sense of presence.

The system makes possible a tangible connection with both virtual / real objects and the ambient space — which maintains a sense of physicality in the VR environment..

References

[1] Future Reality Lab. (2016) Holojam. [Online]. Available: https://github.com/futurerealitylab/Holojam
[2] Optitrack. (2007) Motive. [Online]. Available: http://optitrack.com/
[3] Steam VR. (2016) Vive. [Online]. Available: https://www.vive.com/us/
[4] Unity Technologies. (2005). [Online]. Available: https://unity3d.com/
[5] Google. (2016) Daydream. [Online]. Available: https://vr.google.com/daydream/
[6] Samsung. (2015) Gear vr. [Online]. Available: https://www.oculus.com/gear-vr/
[7] Facebook. (2016) Oculus rift. [Online]. Available: https://www.oculus.com/rift/