Programmable Immersive Peripheral Environmental System (PIPES): A Prototype Control System for Environmental Feedback Devices

Chauncey Frend[a], Michael Boyles[a]

[a]Indiana University Pervasive Technology Institute, 2709 E 10th St., Bloomington, IN 47408, US

ABSTRACT

This paper describes an environmental feedback device (EFD) control system aimed at simplifying the VR development cycle. Programmable Immersive Peripheral Environmental System (PIPES) affords VR developers a custom approach to programming and controlling EFD behaviors while relaxing the required knowledge and expertise of electronic systems. PIPES has been implemented for the Unity engine and features EFD control using the Arduino integrated development environment. PIPES was installed and tested on two VR systems, a large format CAVE system and an Oculus Rift HMD system. A photocell based end-to-end latency experiment was conducted to measure latency within the system. This work extends previously unpublished prototypes of a similar design. Development and experiments described in this paper are part of the VR community goal to understand and apply environment effects to VEs that ultimately add to users' perceived presence.

Keywords: Virtual Reality, Multi-Sensory Feedback, Immersive Systems, Presence

1. INTRODUCTION

We present a new device, PIPES, for more easily extending standard VR systems with EFDs. The augmentation of VEs with EFDs such as those that provide wind, warmth or vibration effects for the user are sparsely applied to VEs. The study of this design practice has been concluded as having useful results. Moon and Kim[10] measured a statistically significant increase in user's perceived presence with their *WindCude* EFD system. Additional sensory input increases a user's memory of objects within the environment according to Dinh's multi-sensory VR memory study[2]. Furthermore Deligiannidis[1] observed improved task performance within a VE using vibrotactile and wind EFDs from their VR Scooter simulator. The apparent need for further study using EFDs will benefit from PIPES simplification through hardware and software tools.

We describe common problems and requirements in adapting EFDs to a VR system that are solved in the PIPES hardware design. Limitations and future improvements are also presented. Sensing and communicating environmental behaviors from the graphics engine to be rendered by the EFDs is difficult in general. PIPES software design should be easy to learn and user friendly. We describe an example of a typical EFD configuration featuring fans interfacing with the Unity game engine through a PIPES node.

2. BACKGROUND

An often referenced early example of an EFD control system is the *Sensorama Simulator* created and patented by Morton Heilig[4] in the 1960s. This apparatus presented a wide array of environmental feedback including wind, vibration, and olfactory scents presented in parallel to a stereoscopic film. Heilig's progressive design proposed the incorporation of EFDs into an immersive system. Portions of this design have been referenced, duplicated, and studied within the domain of virtual reality research. All of these studies seek useful results related to improving user experiences within VEs. The broader goals of PIPES are to relax the required electronics knowledge in applying EFDs to VEs and to eventually lead the way to an accepted EFD incorporation within VR systems to further investigative research.

The Engineering Reality of Virtual Reality 2015, edited by Margaret Dolinsky, Ian E. McDowall,
Proc. of SPIE-IS&T Electronic Imaging, Vol. 9392, 939209 · © 2015 SPIE-IS&T
CCC code: 0277-786X/15/$18 · doi: 10.1117/12.2083410

2.1 Existing EFD control methods

Figure 1 abstracts a model for controlling EFDs. An environmental event is sensed within the VE simulation communicated through some electronic control device, and ultimately rendered from the EFD. In the following examples we omit citing specific VE scenegraph software and focus on the electronic control devices and EFDs. Hülsmann[6] cites the control of 115 volt ADDA Axialfans and Infrared lamps using a MultiDim MKIII Dimmerpack. The event communication is completed by serial DMX-protocol. The combination of the MultiDim system and DMX-protocol is a common control configuration within the theater and lighting industry.

Deligiannidis and Robert used a different control approach in their multimodal feedback study *The vr scooter*[1]. As vr scooter users were passively monitored for environmental events, within the VE, such as wind speed or vibrotactile cues events are sent to what Deligiannidis and Robert call a *Relay Controller and Input Status Interface box*. This box contains an electronic relay control board and an input status interface board. These circuit boards are cited as the AR-16 Relay Interface and the STA-16 Status Input Interface both available from Electronic Energy Controls Inc[5]. The protocol used for event communication was RS-232. This electronic control approach using the AR-16 and STA-16 devices is a common practice within the home energy automation industry.

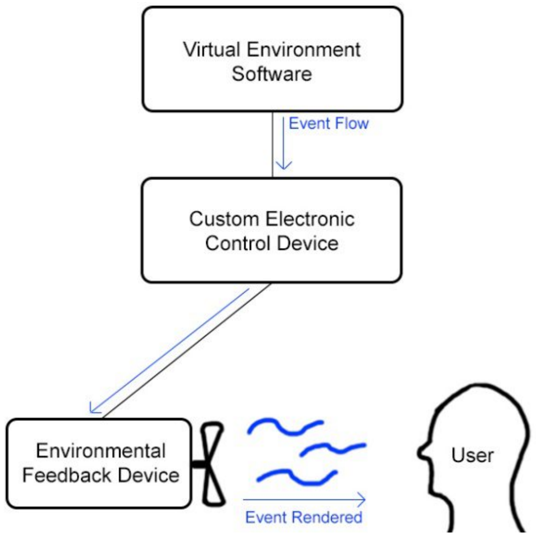
Virtual Environment Software
Event Flow
Custom Electronic Control Device
Environmental Feedback Device
Event Rendered
User

Figure 1. A general model of controlling EFDs.

2.2 EFD control system requirements

We have identified 5 requirements for an EFD control system:

R1 EFD controller should perform with little to no latency in real-time.

R2 An array of common EFDs should be supported in any quantity or combination. (e.g. fans, heaters, vibration motors)

R3 VE environmental conditions should be easily configurable within the VE development environment.

R4 The system needs to support custom behaviors using standard EFDs parameters such as intensity levels, timers, and sequences.

R5 EFD controller hardware should not add cable-clutter to an existing VR system and be relatively easy to setup.

3. IMPLEMENTATION

PIPES is designed to control stationary EFDs as opposed to head mounted or worn EFDs. Lehmann's[7] wind output design study showed results of a higher perceived presence from stationary EFDs to head worn EFDs. This design condition is reasonable considering most EFDs are not designed to be head mounted. A single PIPES node represents a channel based design. Each of these channels contain a microprocessor and an electronic gate to control a single EFD. This channel design offers versatility in adapting PIPES to different VR systems. For example a VR system hosting a standing user might require more channels than a VR system hosting a seated user because of the standing user's ability to traverse a larger floor space.

The PIPES prototype can be seen in figure 2. This prototype consists of a modified power strip where each of the plug electrodes are independent. Each of the electrodes are connected to a corresponding channel with one electronic gate that controls the electrical current flow. By default these channels are off. Each electronic gate connects to an attached Arduino UNO microprocessor system[3].

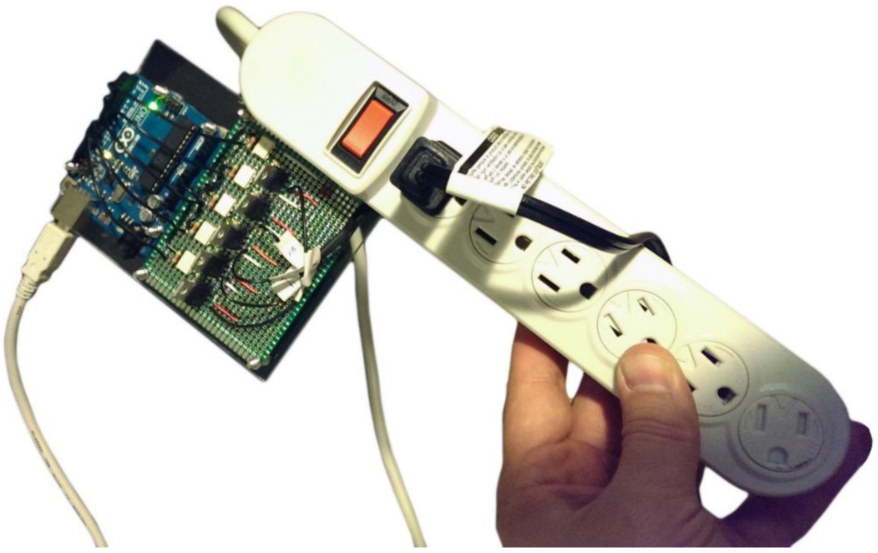

Figure 2. The PIPES prototype device.

Only two cords leave the device. A USB cable that plugs into the VR system host computer. The other is a standard wall electrical socket plug. The USB cable is responsible for serial communication between the VE development software and the Arduino ATMega328 microprocessor and providing 5 volt power. The wall socket plug delivers standard wall power that will be switched on/off by the electronic gates. EFDs plug directly into the power strip sockets.

The controlling circuit of PIPES has a low voltage portion housing the Arduino UNO microprocessor system. The Arduino UNO was chosen because it is a commonly used microprocessor open development platform. The hardware and integrated development environment of the Arduino UNO can be freely licensed under the "Creative Commons Attribution-ShareAlike 3.0 License" [3].

3.1. Electronic channel topology

Wired into the output channels of the microprocessor are the electronic gates. These gates are constructed from MOC3020 opto-isolators to allow the 5 volt portion of the circuit to elicit control of the high voltage portion. The MOC3020 is connected to a BTA 16-600B triac facilitating high voltage on/off switching. Figure 3 shows the schematic of a single channel. The combination of the microprocessor, opto-isolator, and triac represent a single channel within the PIPES design.

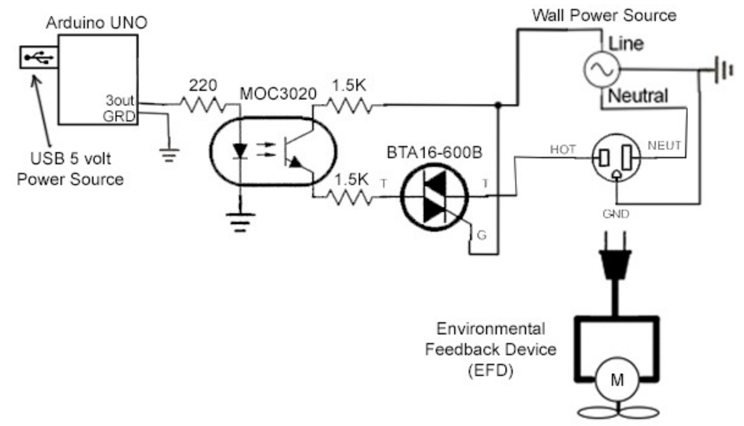

Figure 3. A schematic of a single PIPES electronic channel.

3.2 Software

PIPES software design includes a plugin component compatible with the Unity development editor as well as an EFD behavior control firmware uploaded to the Arduino microprocessor.

Table 1. PIPES default data packet constructor values.

PIPES packet constructor options		
	EFD Channel #1	EFD Channel #n
EFD off	0	0
EFD on	1	1
EFD low intensity	2	2
EFD medium intensity	3	3
EFD high intensity	4	4

Within the Unity editor the PIPES plugin is imported as a *.unitypackage* file containing a PIPES.cs C# class, models, and unity *prefab* objects. Developers need only to add the PIPES script into the scene hierarchy and add environmental event sensors to their user's GameObject. PIPES is currently able to sense Unity wind zones, collider triggers for heat zones, and collider triggers for vibration zones. Developers can apply these zones and triggers freely to a VE using the Unity editor's drag-and-drop interface.

The PIPES.cs class contains default public parameters for each channel allowing developers to access EFD channel states directly from other scripts. These public parameters are integer values each assigned to a single channel. The values comprise a single data packet that is sent to the microprocessor firmware to be parsed and ultimately rendered from each PIPES channel EFD. Table 1 displays optional packet constructor values to control any number of EFD channels. The length of a data packet is equal to the quantity of EFD channels being used plus one checksum value for error checking. For example the PIPES prototype hardware pictured in figure 2 supports up to 6 EFD channels so the appropriate length of a data packet to control this device would be 6 constructor values plus 1 value for error checking.

Environmental sensors, attached as children to the user GameObject within the VE, inherit the user's transformations and rotations. This allows for the hardware EFDs to be arranged in a static real-world position around the user. Figure 4 shows five virtual wind sensors arranged around a user GameObject within the unity editor. Each of these virtual sensors are comprised by a non-rendered cylinder 3D model with a particle system emitter positioned within. As these virtual sensors travel with the user, Unity wind zones translate the emitted particles in accordance with the Unity physics simulation. This effectively acts like a weathervane. Once these particles collide with the interior walls of the cylinder, appropriate wind events are captured and communicated to the PIPES class and ultimately rendered from one or more of the associated connected fans. This approach to sensing wind was created as a result of the limited wind sensing support built into the Unity scripting library.

The default behavioral control firmware uploaded to the microprocessor is designed to parse incoming data packets and helps to control the behavior of the attached EFD. Developers have access to modifying this firmware. For example, fans are great for rendering wind, but most household fans are built simply for on/off use. Intensity control is a desirable feature and can easily be achieved by adding a timer based pulse width modulation loop as an optional on state. Additional pulse width modulation loops can be created for low, medium, and high intensity behaviors. This strategy can also be applied to vibration motor behavior. This is however not recommended for household heaters or infrared heat lamps because they are not designed to handle pulse behaviors, and may result in a safety hazard.

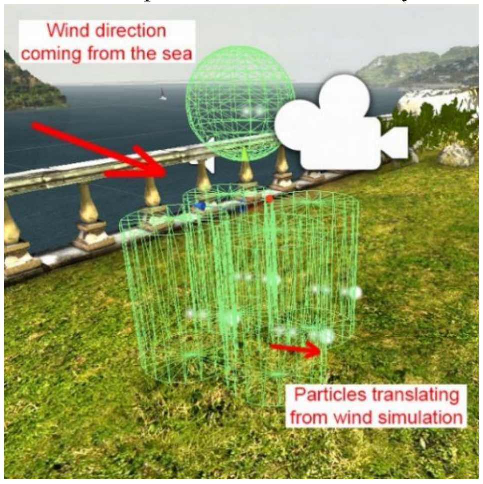

Figure 4. PIPES virtual wind sensors attached to a user.

4. EVALUATION

PIPES was installed and tested on two existing VR systems. The open Unity environments known as Car Tutorial[14] and Tuscany[15] were adapted for wind EFDs and navigated in both PIPES installations. Figure 5 shows PIPES connected fans in a reconfigurable CAVE featuring an ART optical tracking system. This system is driven by a single host PC and a network connected tracking PC.

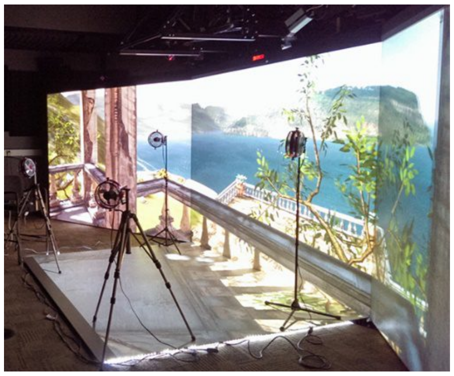

Figure 5. PIPES with fans installed in CAVE VR system

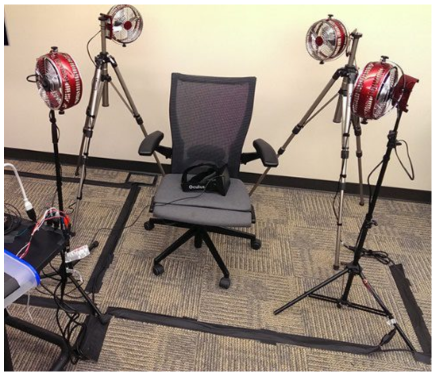

Figure 6. PIPES with fans installed on HMD VR system

Figure 6 shows the PIPES connected fans augmenting a seated user HMD VR system featuring the Oculus Rift. User tracking was performed by the internal oculus rotational sensor array. This system was also driven by a single host PC. During installation of both systems it was observed that the current PIPES prototype required extension cords. This is in contrast to the requirement [R5] that an EFD control system not add cable clutter. By disassociating PIPES channels into a wireless network of single channels this issue will be avoided, but may add cost and latency to the system. Individual channels plugged directly into wall sockets that receive serial packets wirelessly will require minimal cable clutter.

During software conversions only VR display and tracking configurations were modified and the PIPES software configuration was left unchanged. The connected EFDs were controlled effectively during both evaluations.

Anecdotal user feedback was positive. Five users navigated the Car Tutorial[14] and Tuscany[15] Unity environments. Each user navigated the VEs using a CAVE and a HMD VR system. Each VR system displayed the VE with and without environmental feedback from 4 fans arranged around the user. Users all preferred the environmental feedback. The environmental feedback was described by users with words such as; *relaxing*, *impressive*, and *realistic*. Pausch[12] incorporated simulated wind from a single fan in a HMD based VR system and found that most users did not notice the simulated wind. These differing results in anecdotal user feedback might be attributed to the use of multiple fans in contrast to Pausch's use of a single fan.

Development in adapting PIPES to existing VEs proved to be elegant within the Unity editor, but clumsy within the Arduino IDE. A design flaw of the current prototype is that the Arduino is connected to multiple channels. This flaw is apparent when developing Arduino behaviors that require multi-threaded processing. Timer functions for example halt the entire firmware loop and prevent parallel processes. A solution to this issue is the incorporation of a microprocessor per channel or a multithreaded microprocessor controlling multiple channels.

4.1 PIPES latency study

Using EFDs in a VR system poses the challenge of operating in real-time. Meehan[8] recorded users' heart rate levels to be higher in a stressful VE when system end-to-end latency was lowest. When high system latency was added users no longer perceived presence in the VE. It is critical that environmental feedback also not be delayed in order to preserve the user's perceived presence within the overall VE.

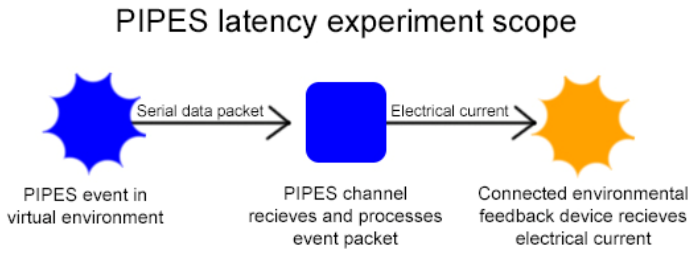
PIPES latency experiment scope
Serial data packet
Electrical current
PIPES event in virtual environment
PIPES channel recieves and processes event packet
Connected environmental feedback device recieves electrical current

Figure 7. The scope of PIPES latency experiment.

The experiment designed to measure the system latency was inspired by Miné's[9] photodiode based measurement system designed to measure tracking system latency. Miné's experiment aimed to quantify latency within the entire system between real-world movements to graphical rendered movement. The scope of our experiment starts as an event sensed within the VE. Next the EFD behavior is communicated to PIPES as a serial data packet and PIPES parses the packet. Finally the electrical gate is opened exposing the connected EFD to wall socket electrical current. Figure 7 diagrams the flow of the experiment. Figure 8 illustrates the hardware configuration. The *Photocell timer circuit* consists of an Arduino UNO microprocessor and a connected analog photocell component. The firmware in this device was designed to start a timer delimited by milliseconds once a key is pressed within the VE. The VE signals PIPES with a serial packet requiring that the connected EFD be turned on. A household lamp serves as our EFD. The light emitted by the lamp is captured by the photocell timer circuit which stops the timer.

We chose to use a household lamp as the EFD in order to minimize any EFD-specific latency. Hülsmann[6] measured ~3 seconds of latency caused by fans. The total amount of latency will vary drastically depending on the type of EFD. A household lamp represents an EFD with minimal latency. This allows us to more accurately assess the PIPES system itself.

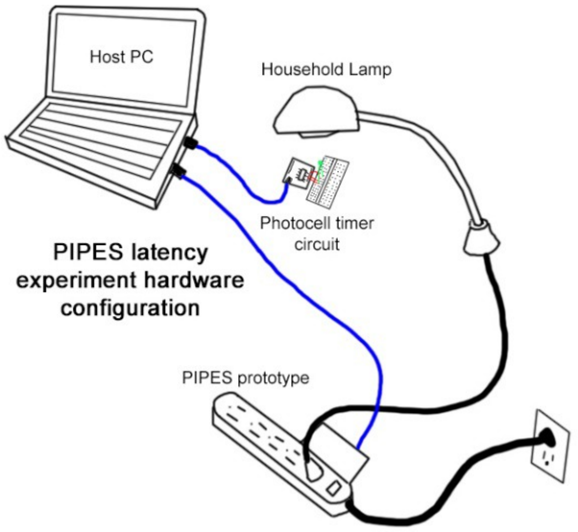

Figure 8. The hardware configuration for measuring system latency.

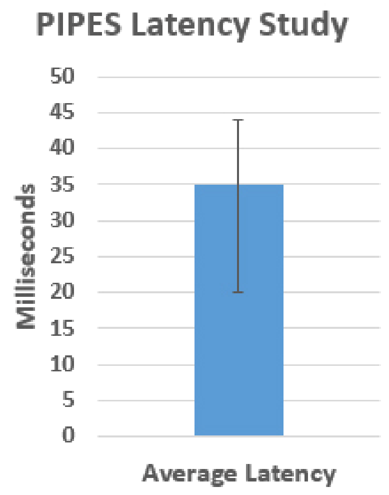

Figure 9. PIPES end-to-end latency results.

The results of our experiment are charted in figure 9. We measured an average system latency of 35.1 milliseconds with a high point of 44 milliseconds and a low point of 20 milliseconds. A source of our data variation is attributed to the noisy data generated from the analog photocell component. 20 measurements were recorded.

We believe this average latency is within an acceptable range to provide consistent real-time EFD control. PIPES is a remarkable EFD control system.

5. FUTURE WORK

The fundamental technology of PIPES is incomplete in regards to scalability. By re-architecting the design and utilizing a scalable network based communication software such as the Virtual Reality Peripherals Network (VRPN)[13], PIPES could offer a truly scalable solution for EFD control within established VR systems.

Community resources such as source code, printable circuit board files, and 3D printable enclosure files will be released as the utility of, interest in, and demand for PIPES increases.

Incorporating more existing types of EFDs with PIPES into VR systems will increase the diversity of environmental feedback. The creation of new EFDs to increase the immersive quality of VEs is apparent. By adding diversity to a VR system's EFD array the potential for new user experiences is likely. For example, Yu's[16] work in designing air canon EFDs offer utility in projecting olfactory feedback as well as pulsed air flow. Multi-functional EFDs will enrich the diversity of an EFD array while offering remarkable user experiences.

As more EFDs are established new methods of virtual environmental sensing will be required. Furthermore accepted VE development methods for mapping simulated environmental conditions that are rarely rendered from EFDs will be explored. Examples of these may include humidity, air particle collisions, or precipitation.

6. CONCLUSIONS

This paper describes PIPES, an environmental feedback device control system for virtual environments. Other control systems were discussed and requirements identified. PIPES was evaluated for installation, user feedback, development efficiency, and control system latency. Results were positive, but improvements were identified and discussed.

ACKNOWLEDGMENTS

The Advanced Visualization Lab at Indiana University is a unit of the Research Technologies division of University Information Technology Services; Research Technologies is a Pervasive Technology Institute Cyberinfrastructure and Service Center.

REFERENCES

[1] Deligiannidis, Leonidas, and Robert JK Jacob. "The vr scooter: Wind and tactile feedback improve user performance." 3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium, (2006).
[2] Dinh, Huong Q., et al. "Evaluating the importance of multi-sensory input on memory and the sense of presence in virtual environments." Virtual Reality, 1999. Proceedings., IEEE, (1999).
[3] FAQ support page of Arduino creative commons project http://arduino.cc/en/Main/FAQ (2014).
[4] Heilig, Morton L. "Sensorama simulator." U.S. Patent No. 3050870. 28 Aug., (1962).
[5] Home page of EECI, "Electonic Energy Control Inc," (EECI), http://eeci.com (2014).
[6] Hülsmann, Felix, et al. "Wind and warmth in virtual reality-Requirements and chances." Proceedings of the Workshop Virtuelle & Erweiterte Realität 2013., (2013).
[7] Lehmann, Anke, et al. "Poster: Design and evaluation of 3D content with wind output." 3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium, (2009).
[8] Meehan, Michael, et al. "Effect of latency on presence in stressful virtual environments." Virtual Reality, 2003. Proceedings. IEEE., (2003).
[9] Miné, Mark. "Characterization of end-to-end delays in head-mounted display systems." The University of North Carolina at Chapel Hill, TR93-001 (1993).
[10] Moon, Taeyong, and Gerard J. Kim. "Design and evaluation of a wind display for virtual reality." Proceedings of the ACM symposium on Virtual reality software and technology. ACM, (2004).
[11] Okamura, Allison M., Mark R. Cutkosky, and Jack T. Dennerlein. "Reality-based models for vibration feedback in virtual environments." Mechatronics, IEEE/ASME Transactions on 6.3 (2001): 245-252., (2001).
[12] Pausch, Randy, et al. "Disney's Aladdin: first steps toward storytelling in virtual reality." Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. ACM, (1996).
[13] Taylor II, Russell M., et al. "VRPN: a device-independent, network-transparent VR peripheral system." Proceedings of the ACM symposium on Virtual reality software and technology. ACM, (2001).
[14] Unity 3D environment data, "Car Tutorial," Unity Asset Store, https://www.assetstore.unity3d.com/en/#!/content/10, (2014).
[15] Unity 3D environment data, "Tuscany," Oculus Developer Downloads, https://developer.oculus.com/downloads/, (2014).
[16] Yu, Jiang, et al. "Air Canon Design of Projection-Based Olfactory Display." *ICAT*., (2003).