Component-based Authoring Tool for Haptic Navigation

Takehiko Yamaguchi[2], Kazuhiro Oshima[1], Yuto Hirano[2], Akiyo Makishima[2],
Testsuya Harada[2] and Paul Richard[3]
[1]*Faculty of Industrial Science and Technology Applied Electronics, Tokyo University of Science, Niijuku6-3-1, Katsushikaku, Tokyo, Japan*
[2]*The Department of Appliced Electronics, Tokyo University of Science, Niijuku6-3-1, Katsushikaku, Tokyo, Japan*
[3]*Laboratoire Angevin de Recherche en Ingénierie des Systèmes (laris ea 7315), Université d'Angers, Angers, France*

Keywords: Haptic Authoring Tool, Haptic Navigation, Component-based IDE, Unity3D.

Abstract: Nowadays, providing a uniform development environment for haptic applications is considered as one of the critical issues in haptic technologies. Thus, to date, we have developed a component-based haptic authoring framework using a component capability extension mechanisms supported by Unity3D. Our aim is to develop a newly revised haptic component to extend our previous framework. The proposed component enables a user to design haptic interface such as haptic navigation, as well as to help simulate/investigate the effect of the haptic interaction. In this paper, we present a prototype system and its capabilities.

1 INTRODUCTION

Nowadays, the field of haptics have grown into numerous research activities, comprising four interdisciplinary research branches: human haptics, machine haptics, computer haptics, and the newly introduced multimedia haptics (Eid et al., 2007).

As the findings from these research fields, many haptic devices as well as applications have been developed for specific purpose such as rehabilitation (Broeren et al., 2004), training (Feygin et al., 2002), entertainment (Fyans et al., 2008), education (Bivall et al., 2011), and so on. In view of this background, providing a uniform development environment for haptic applications is considered as one of the critical issues (Eid et al., 2007).

In fact, having a framework that facilitate the development process of haptic applications enable to help novice users since the process remains a time consuming experience and requires programming expertise, for instance when adding haptic properties such as the stiffness, static or dynamic friction (Eid et al., 2008). Thus, to date, different authoring tools have been developed.

1.1 Haptic Authoring Tools

Unison is an extensible framework to standardize the development process of hapto-visual applications. With this framework, a user is required to develop a plug-in to install a component into the framework (El-Far et al., 2004). The Haptik Library is a software library which enables to provide uniform access to haptic devices (Pascale and Prattichizzo, 2007). The library does not support higher capabilities to simulate haptic rendering such as collision detection so that significant programming effort is required. Cha et al. (Cha et al., 2007) proposes an authoring tool that enables to haptically enhance broadcast contents as well as to provide viewers via streaming in a VOD context. This framework supports an only passive haptic interaction for a user. HAMLAT is an authoring tool for haptic application development (Eid et al., 2008).

This framework was designed using the Blender modeling tools (BlenderFoundation, 1995) as a main development platform, which enables to edit a 3D graphical model with haptic interaction. However, the system does not support an animation, namely the system implementation is limited to static scenes.

1.2 Component-based Haptic Authoring Tool

To date, we have developed a component-based

Yamaguchi T., Oshima K., Hirano Y., Makishima A., Harada T. and Richard P..
Component-based Authoring Tool for Haptic Navigation.
DOI: 10.5220/0005357804860491
In *Proceedings of the 10th International Conference on Computer Graphics Theory and Applications* (GRAPP-2015), pages 486-491
ISBN: 978-989-758-087-1

haptic authoring framework for Unity3D (Yamaguchi et al., 2014). This framework was designed as an extensible module using a component capability extension mechanisms supported by Unity3D (Unity Technologies, 2005) to accommodate the range of predictable variation that will be needed in different specific purpose of haptic interaction.

Unity3D is one of the most well designed component-based game development platform that enables rapid prototyping of high quality 2D/3D game. It runs on multiplatform such as Mac OS X, Windows, Linux, android, and even a web browser.

The proposed haptic component in the previous study (Yamaguchi et al., 2014) was integrated in the Unity Editor (Fig.1). We implemented five primitive haptic components as a haptic widget such as Vibration force, Collision response force, Viscosity force, Bump force, and Toggle force. These components could be dynamically added/removed on a game object in Unity game scene, enabling real time simulation of haptic effect with visual/audio effect.

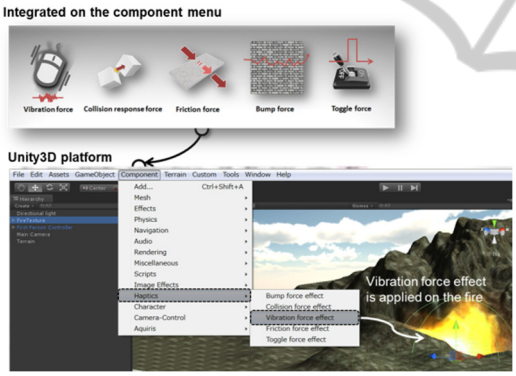

Figure 1: The integration of haptic components on the component menu of Unity3D IDE.

1.3 Newly Developed Haptic Component

Currently, haptically enhanced interaction mainly relies on magnetic effects or gradient force effects (Villard and Capobianco, 2009) in haptic navigation. These haptic effects enable to enhance user's spatial recognition as well as a sense of direction, however sometimes it decreases performance for example concerning task selection times (Oakley et al., 2001).

Although a number of researchers put forth efforts to overcome usability problem of haptic interaction in navigation, very little research has been conducted on the development of user interface design guidelines or user friendly features for haptic systems (Yamaguchi et al., 2009). In this perspective, there is a clear need for creating an authoring tool that allows to build as well as to help simulate/ investigate an effect of haptic interaction in navigation.

Our aim is to develop a haptic component module which enables to design a haptic user interface such as haptic navigation to extend our previous system. In this paper, we present a prototype system and its capabilities.

2 PROPOSED SYSTEM

2.1 System Framework

The system framework of the proposed system is shown in Figure 2. The framework consists of two modules: a "*Haptic Device Controller*" module and a "*Haptic Model*" module. Currently, we utilize a Novint Falcon™ which is a commercially available 3DoF force feedback device produced by Novint Technologies (Novint Technologies, 2006).

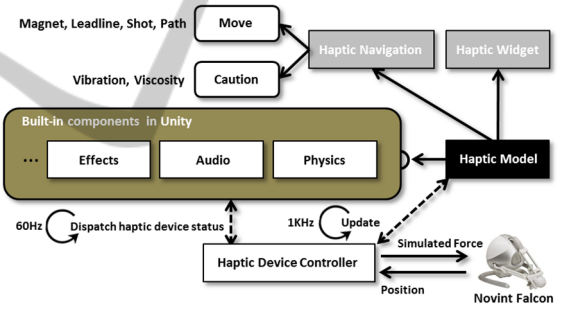

Figure 2: The system framework of the proposed system.

2.2 Haptic Device Controller Module

The "*Haptic Device Controller*" works as an interface to the connected haptic device and retrieves its status as well as dispatches the status to the other components including the "*Haptic Model*" module.

The haptic device status includes a grip position of the haptic device, a button status, and so on. These statuses are updated at 60Hz. A core thread of this module is implemented as a C++ based Dynamic Link Library (DLL) to access the API of the haptic device, as well as to manage a haptic rendering process at 1 kHz.

2.3 Haptic Model Module

The "Haptic Model" module enables to manage reaction force calculation logic, a database of the

haptic device status. The reaction force calculation process is running at 1 kHz to calculate a reaction force for an applied haptic component. The calculated force is sent to the ”Haptic Device Controller” module to display it via the connected haptic device. This module is listed on the Built-in components in Unity3D.

We categorized a common haptic interaction in navigation into two abstract components such as "Move", and "Caution". These abstract components are defined as sub modules of the “Haptic Model” module.

Each haptic component is associated with a visual object which represents a visually defined work area related to a haptic effect of the linked haptic component.

2.3.1 “Move” Component

The “*Move*” component consists of four primitive haptic components such as *Magnet*, *LeadLine*, *Shot*, and *Path*.

Magnet Component: This component is associated with a 3D sphere object which is a visual object that represents a work area of the *Magnet* Component. When one’s cursor, which is operated by a connected haptic device, is within the work area, for example, it will visually and haptically “*Snap*” with an attractive force to the center of the work area. The force of the *Magnet* Component: F_{magnet} was determined according to equation 1.

$$\boldsymbol{F_{magnet}} = m_{magnet}.\left(r - \left|\boldsymbol{P_{(t)}} - \boldsymbol{Q}\right|\right).\frac{\boldsymbol{Q}-\boldsymbol{P_{(t)}}}{|\boldsymbol{Q}-\boldsymbol{P_{(t)}}|} \quad (1)$$

where m_{magnet} is a spring constant, $\boldsymbol{P_{(t)}}$ is a position of user’s cursor, $\boldsymbol{Q}$ is a position of the *Magnet* Component, and r is a radius of the work area.

LeadLine Component: A set of primitive 3D visual objects such as cube, sphere, cylinder, and capsule area associated with this component to represent a work area of the *LeadLine* Component. When a user’s cursor is within the work area, it will visually and haptically move with a quick or an abrupt motion to a user defined direction. The force of the *LeadLine* Component: $F_{leadline}$ is expressed as equation 2.

$$\boldsymbol{F_{LeadLine}} = m_{leadline} \cdot \boldsymbol{n} \quad (2)$$

Where $m_{leadline}$ is a constant value which is related to a speed of the applied motion within the *LeadLine* Component area. $\boldsymbol{n}$ is the user defined direction.

Shot Component: A set of primitive 3D visual objects are associated with the Shot Component as a trigger area. When a user’s cursor is within the trigger area, it will visually and haptically move with a quick motion to a user defined terminal position. The terminal position could be repressed with a position of a 3D sphere. The force of the Shot Component: F_{Shot} is expressed as in equation 3.

$$\boldsymbol{F_{Shot}} = m_{Shot} \cdot \frac{\boldsymbol{G}-\boldsymbol{P_{(t)}}}{|\boldsymbol{G}-\boldsymbol{P_{(t)}}|} \quad (3)$$

Where, m_{Shot} is a constant value which represents the speed of the applied motion. $\boldsymbol{G}$ is the position of the terminal point.

Path Component: A set of primitive 3D visual objects are associated with the *Path* Component as a trigger area. This component is associated with a 3D path graph which represents a 3D line as a work area. This path is enabled to be edited by a user on the Unity3D Editor. When a cursor is within the trigger area, it will visually and haptically move on the 3D path with a quick motion. The force of the *Path* Component: $\boldsymbol{F_{Path}}$ is expressed as in equation 4.

$$\boldsymbol{F_{Path}} = m_{path} \cdot \frac{\boldsymbol{Q_{(t)}}-\boldsymbol{P_{(t)}}}{|\boldsymbol{Q_{(t)}}-\boldsymbol{P_{(t)}}|} \quad (4)$$

m_{path} is a constant value which represents the speed of the applied motion. $\boldsymbol{Q_{(t)}}$ is a pointer object which is jointed with a user’s cursor position: $\boldsymbol{P_{(t)}}$. The pointer object automatically moves on the 3D path.

2.3.2 “Caution” Component

The “*Caution*” component consists of two primitive haptic components such as Vibration, and Viscosity.

Vibration Component: This component is associated with a set of primitive 3D objects that represents a work area of the *Vibration* Component. When one’s cursor is within the work area, it will visually and haptically vibrate. The force of the *Vibration* Component: $\boldsymbol{F_{Vibration}}$ is expressed as in equation 5.

$$\boldsymbol{F_{Vibration}} = m_{Vibration} \cdot \boldsymbol{rand}() \quad (5)$$

$\boldsymbol{rand}()$ is an unit direction vector which represents a direction of the applied vibration haptic effect. The direction of the unit vector is randomly generated between 0 and 1 for each direction component: x, y, and z, and then it is normalized. $m_{Vibration}$ is a constant value which gives a magnitude of the applied vibration.

***Viscosity Component*:** A set of primitive 3D objects that represents a work area of the *Viscosity* Component. When a cursor moves within the work area, it will visually and haptically move with a viscous motion. The force of the Viscosity Component: $\boldsymbol{F_{Viscosity}}$ is expressed as in equation 6.

$$\boldsymbol{F_{Viscosity}} = -\, m_{Viscosity} \cdot (\boldsymbol{P_{(t)}} - \boldsymbol{Q_{(t)}}) \quad (6)$$

$m_{Viscosity}$ is a constant value which represents a coefficient of viscosity. $\boldsymbol{Q_{(t)}}$ is a 3D pointer which is connected to the cursor with a virtual fixed joint. The fixed joint restricts the movement of the $\boldsymbol{Q_{(t)}}$to be dependent upon the cursor position. The spring constant of the fixed joint is set at a high stiffness, e.g., 1000 N/m.

3 IMPLEMENTED RESULTS

A brief outline of the practical use of the proposed system is provided in this section. The development phase consists of three phases: (1) Definition of the work area which is visually represented, (2) applying a haptic component to the defined work area, and (3) preview of the designed haptic effect.

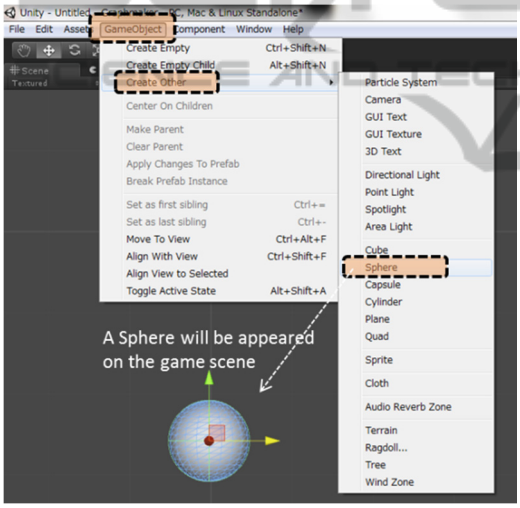

Figure 3: The process of the definition of the work/trigger area.

3.1 Definition of the Work Area

The author is required to define a work area or a trigger area for a haptic component. A set of primitive 3D objects built in the Unity3D Editor are used as the work/trigger area. In this example, a sphere is used as these areas.

The sphere object can be found on the "Create Other" menu under the "GameObject" menu which is on the main menu of the Unity3D Editor. When selected, a sphere appears on the game scene on the Unity3D Editor, allowing users to view, rotate, scale, and translate as well as to edit the parameter of the sphere (Figure 3).

If a haptic component which will be applied on the sphere is: a *Magnet*, a *Leadline*, a *Vibration*, or a *Viscosity* Component, the sphere will work as a work area. If it is a *Shot*, or a *Path* Component, the sphere will work as a trigger area.

3.2 Applying a Haptic Component

For the second phase, a haptic component can be applied to the defined work/trigger area. A set of haptic components can be found on the "Haptic" menu under the "Component" menu which is on the main menu of the Unity3D Editor. In this section, three examples are provided using *Magnet*, *Path*, and *Viscosity* Component.

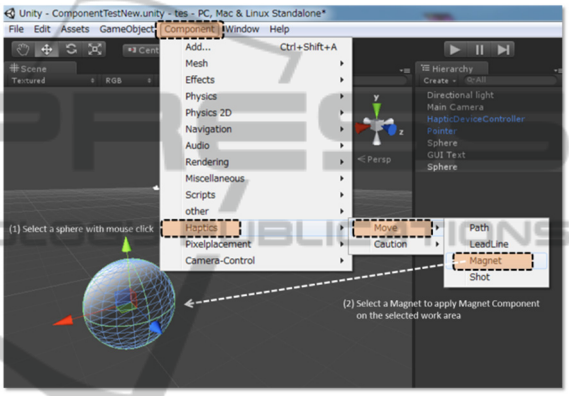

Figure 4: The applied Magnet Component on the sphere type work area.

Magnet Component: The defined sphere works as a work area. When the work area is selected by a mouse click, the *Magnet* Component can be applied to select the "*Magnet*" in the "*Move*" menu under the "Haptic" menu (Figure 4). The applied haptic effect works while a user's cursor is within the work area.

Path Component: The defined sphere works as a trigger area. When the work area is selected by mouse click, the *Path* Component can be applied to select the "*Path*" in the "*Move*" menu (Figure 5).

When the *Path* Component is applied on the work area, a 3D path graph will appear on the trigger area. The 3D path graph can be edited on the game scene. Furthermore, an additional path can be added to change a number of paths which is on the inspector menu. An applied haptic effect works when a user's cursor is within the trigger area.

Viscosity Component: The defined sphere works as a work area. When the work area is selected by mouse click, the *Viscosity* Component can be applied to select the "*Viscosity*" in the "*Caution*" menu. The applied haptic effect works while a user's cursor is within the work area.

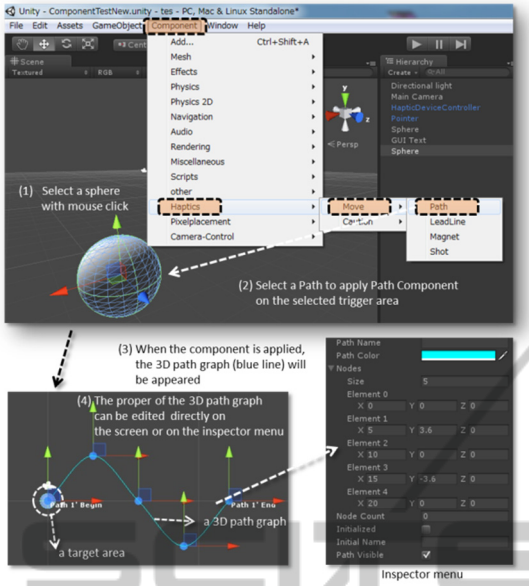

Figure 5: The applied Path Component on the sphere type trigger area.

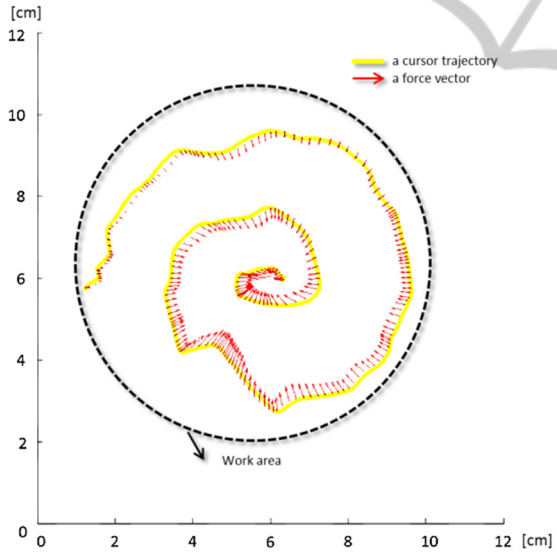

Figure 6: The cursor's trajectory with the applied force vector of the *Magnet* Component.

3.3 Results of the Applied Haptic Effect

For the third phase, the applied haptic effect can be previewed by pressing the play button on the Unity3D Editor. In this section, the results of the preview are shown concerning the three examples described in section 3.2.

Magnet Component: Figure 6 shows the previewed result of the applied haptic effect using the *Magnet* Component. The applied force vector with the cursor's trajectory is shown in Figure 7. The result shows that when the user's cursor is within the work area, it visually and haptically moves with an attractive force to the center of the work area.

Path Component: Figure 7 shows the previewed result of the applied haptic effect using the *Path* Component. The applied force vector with the cursor's trajectory is also shown in the figure. The result shows that when a cursor is within the trigger area, it visually and haptically moves on the 3D path with a quick motion.

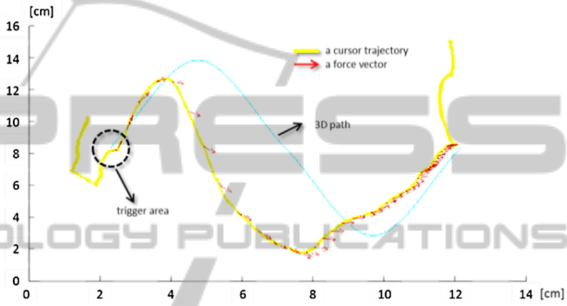

Figure 7: The cursor's trajectory with the applied force vector of the Path Component.

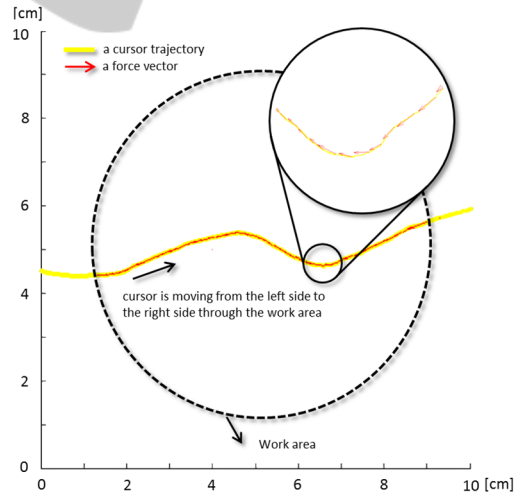

Figure 8. The cursor's trajectory with the applied force vector of the Viscosity Component.

Viscosity Component: Figure 8 shows the previewed result of the applied haptic effect using the *Viscosity* Component. The applied force vector with the cursor's trajectory is also shown in the figure. The result shows that when a cursor moves within the work area, it visually and haptically moves with a viscous motion.

4 CONCLUSION

In this paper, we proposed a component-based authoring tool for haptic navigation using Unity3D game engine as a main development platform. The proposed system was extended based on our previous work. The integrated haptic components enable a user to quickly develop a haptic application for navigation with no programming required.

We categorized a common haptic interaction in navigation into two abstract components such as "*Move*", and "*Caution*". The "*Move*" component consists of four primitive haptic components such as *Magnet*, *LeadLine*, *Shot*, and *Path*. The "*Caution*" components consists of two primitive haptic components such as *Vibration*, and *Viscosity*.

The results of the preview of the applied haptic effect: *Magnet*, *Path*, and *Viscosity* Component show how easily our framework can be used to built a haptic navigation environment as well as how the applied navigation works to see the cursor's trajectory with the applied force vector while the applied haptic effect is working.

As future work, we plan to extend the haptic component to support a variety of haptic interaction.

REFERENCES

Blender Foundation. Blender. (1995). http://www.blender.org/

Bivall, P., Ainsworth, S., & Tibell, L. A. (2011). Do haptic representations help complex molecular learning?. *Science Education*, *95*(4), 700-719.

Broeren, J., Rydmark, M., & Sunnerhagen, K. S. (2004). Virtual reality and haptics as a training device for movement rehabilitation after stroke: a single-case study. *Archives of physical medicine and rehabilitation*, *85*(8), 1247-1250.

Cha, J., Seo, Y., Kim, Y., & Ryu, J. (2007, March). An authoring/editing framework for haptic broadcasting: passive haptic interactions using MPEG-4 BIFS. In *Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (pp. 274-279). IEEE Computer Society.

Eid, M., Orozco, M., & El Saddik, A. (2007). A guided tour in haptic audio visual environments and applications. *International Journal of Advanced Media and Communication*, *1*(3), 265-297.

Eid, M., Andrews, S., Alamri, A., & El Saddik, A. (2008). HAMLAT: A HAML-based authoring tool for haptic application development. In *Haptics: Perception, Devices and Scenarios* (pp. 857-866). Springer Berlin Heidelberg.

El-Far, N. R., Shen, X., & Georganas, N. D. (2004, October). Applying Unison, a generic framework for hapto-visual application developments, to an e-commerce application. In *Haptic, Audio and Visual Environments and Their Applications, 2004. HAVE 2004. Proceedings. The 3rd IEEE International Workshop on* (pp. 93-98). IEEE.

Essert-Villard, C., & Capobianco, A. (2009, November). Hardborders: a new haptic approach for selection tasks in 3d menus. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology* (pp. 243-244). ACM.

Feygin, D., Keehner, M., & Tendick, F. (2002). Haptic guidance: Experimental evaluation of a haptic training method for a perceptual motor skill. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002. HAPTICS 2002. Proceedings. 10th Symposium on* (pp. 40-47). IEEE.

Fyans, A. C., & McAllister, G. (2008). Creating games with feeling. In *Proc. Int. Conf. on Computer Games: Artificial Intelligence and Mobile Systems, Las Vegas, NV* (pp. 94-98).

Novint Technologies. Novint Falcon. (2006). from http://www.novint.com/index.php/novintfalcon.

Oakley, I., Brewster, S., & Gray, P. (2001, March). Solving multi-target haptic problems in menu interaction. In *CHI'01 extended abstracts on Human factors in computing systems* (pp. 357-358), ACM.

Pascale, de M and Prattichizzo, D., (2007). The Haptic Library: A Component Based Architecture for Uniform Access to Haptic Devices. *IEEE Robotics & Automation Magazine*, *14 (4)*, 64-75.

Unity Technologies. Unity3D: Game development tool. (2005). from http://unity3d.com/

Yamaguchi, T., Richard, P., Oshima, K., & Kato, K. (2014). Component-Based Haptic Application Creation Framework for Unity3D. In *Proceedings of IWAIT 2014*.

Yamaguchi, T., Johnson, S., Kim, H. N., Li, Y., Nam, C. S., & Smith-Jackson, T. L. (2009). Haptic science learning system for students with visual impairments: A preliminary study. In *Universal Access in Human-Computer Interaction. Applications and Services* (pp. 157-166). Springer Berlin Heidelberg.