School of Engineering and ICT

UTAS_CMYK_colour_Domestic_Stacked_Cent

ICT Discipline

# Electronic Submission Assignment Cover Sheet

If you are required to submit your assignment work via MyLO or if you are required to do an electronic submission only (no paper submission), then you must submit a copy of this cover sheet using the same method as you are required to submit the rest of the assignment.

Student ID: 222671 Login Name: bhayes1

Family Name: Hayes Given Name: Brad

Unit Code: KIT208 Unit Name: Mixed and Virtual Reality

Tutorial Day/Time: Friday 3pm

Assignment Title/Number: Assignment 3: Gesture Shell

**By submitting this coversheet electronically I declare that all material in this assignment is my own work except where there is clear acknowledgement or reference to the work of others and I have complied and agreed to the University statement on Plagiarism and Academic Integrity on the University website at http://www.utas.edu.au/plagiarism or in the Student Information Handbook. I am aware that my assignment may be submitted to plagiarism detection software, and might be retained on its database.**

### Statement on Plagiarism and Academic Integrity

Plagiarism is a form of cheating. It is taking and using someone else's thoughts, writings or inventions and representing them as your own, for example:

* using an author's words without putting them in quotation marks and citing the source;
* using an author's ideas without proper acknowledgment and citation; or
* copying another student's work.

If you have any doubts about how to refer to the work of others in your assignments, please consult your lecturer or tutor for relevant referencing guidelines, and the academic integrity resources on the web at http://www.utas.edu.au/tl/supporting/academicintegrity/index.html.

The intentional copying of someone else’s work as one’s own is a serious offence punishable by penalties that may range from a fine or deduction/cancellation of marks and, in the most serious of cases, to exclusion from a unit, a course or the University. Details of penalties that can be imposed are available in the Ordinance of Student Discipline – Part 3 Academic Misconduct, see http://www.utas.edu.au/universitycouncil/legislation/ord9.pdf

argued for an intuitive input system that sort of combines standard PC/Laptop interactions, with that of Tablet Actions. So by combining the swipe, pinch, grab and zoom functions that we are so used to with smart devices.

To improve this aspect of the design process. I was responsible for the storyboarding and translating that into the interactions that we saw in the final project videos. Brad however was responsible for the technical deisgn.

## Presentation:

Whilst Brad was busy developing the technical side, I was building the Powerpoint presentation for which to present to the class. This included incorporating demonstration videos, as well as building each slide and running through the majority of the presentation with the class.

Overall I would say that the work was roughly 50/50, Brad managed the majority of the technical aspects, whilst I handled the presentation.

-Tim Anderson 180218

# Individual Report- Bradley Hayes

**1 – Introduction.**

Project name: Gesture Shell  
Developers:

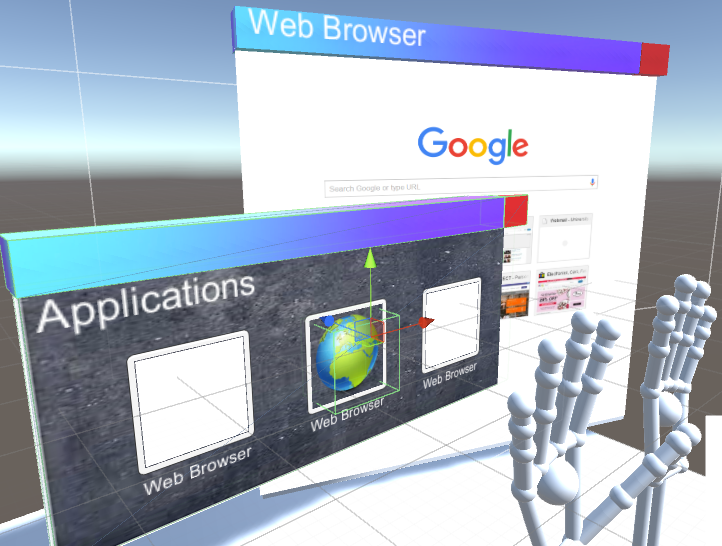
* Bradley Hayes : 222671
* Tim Anderson : 180218

Figure 1: Scene view in Unity of concept model

# Introduction:

Gesture Shell drew inspiration from a variety of sources. The need to develop an intuitive interface that doesn’t stray too far from the desktops that users are familiar with, yet still needing to be interactive and engaging for the user. Gesture Shell also draws interfaces from work people have done with Graphical User Interfaces (GUIs) for use inside of games and other programs. A prime example of this has been HocGaming’s Sword Art Online GUI that has been shared via YouTube (HocGaming, 2015).

Gesture Shell is a replacement for typical operating system interface to incorporate VR headsets and hand tracking. Built to resemble the familiar Windows Explorer Shell Interface. As seen in Figure 1 the applications window would operate similar to a start menu in windows listing all available applications that can run in the interface. Allowing the user to select them with a finger press using hand tracking hardware such as the Leap motion. (<https://www.leapmotion.com/>)

Combined with a VR headset such as the Oculus Rift (<https://www.oculus.com/en-us/>) users can have a desktop environment with potentially unlimited screen real-estate. Allowing the user to instantiate application windows and place them anywhere in 3d space with their hands.

With the growing number of power users, developers and workers running a great numbers of applications simultaneously, the communication bandwidth between the user and their computer is too limited for the level of productivity they demand from themselves and their working environment. Eventually, buying more computer screens and an expensive keyboard is not going to be enough.

Many applications have been developed for VR and AR headsets with hand tracking and special controller’s. These applications have shown dramatic improvements in productivity, such VR Clay (<http://vrclay.com/>) which has enabled developers to build highly detailed 3d models with unparalleled speed and accuracy, reducing days of work into minutes. This power needs to be brought to all users not just 3d development industries. We need to bring the everyday user, the multitasking power user, the web designers, programmers and the Gesture Shell concept may be the stepping stone into bringing these users up to speed.

Virtual Reality, hand tracking and other forms of human interface hardware need to become common place in order to address this ‘interaction bandwidth’ problem on a global scale.

# 4.0 User Needs:

## 4.1 Screen Real-estate:

It is becoming very common practice to use 2 or 3 displays to increase the desktop real-estate and be able to see several application windows without having to frequently switch between them. They are increasing resolutions, buying 4k screens, some going as far as having multiple large 4k monitors to fit all their applications into one viewable area. This will be addressed enabling the user to look around in a 3d environment and position windows anywhere they want without being limited by the size and number of screens they have.

## 4.2 Intuitive and familiar interaction mechanics:

In order for VR and hand tracking to become common place the majority needs to be able to easily adapt to the new environment. To address this, we have designed the interface to resemble the popular Windows Explorer shell. Applications have a title bar with a close button (with plans to implement maximize and minimize also) that can be grabbed and moved around the environment. Essentially the user would feel as if each application window is a virtual tablet device.

## 4.3 Fun and Exciting & Chosen Interface Technology:

To make the user feel excited and enjoy the experience we had many ideas to implement visual feedback from their interaction. We have tested ideas just as making the buttons 3d objects that push in and spring back. Close buttons that fling the application window off into endless space.

The interface technology that we have chosen to work with is the Leap Motion controller. The Leap Motion is a USB peripheral that contains two Infrared cameras. These infrared cameras can detect hands that are placed above the camera. Software bundled with the Leap Motion then translate the data into usable actions and data points in space. The Structures of the hands are set up in an anatomically correct manner and the bones of the hand models are named as such, These points can then be used to interact with buttons for example in Unity.

# 5 - Technical Details

## 5.1 Interaction

Indented features of the interface:

* Global gestures. For binging up menu for the Gesture Shell such as list of available applications or the virtual keyboard.
* Virtual Keyboard. Appearing as needed for text inputs or on demand with a global gesture.
* Tablet/Mobile device Gestures such as swiping, finger scrolling etc.
* 3d Task bar ring that applications can be minimized to.
* User options to customize global gestures.
* User options to customize application gestures.
* Ability to replace the standard windows shell:
  + Giving user access to all their native windows applications.
  + Launching windows applications and streaming their windows as a texture in the Gesture Shell.
* Resizing of windows. Two handed rotation and resize can be simultaneously executed but was problematic during tests so was disabled.

## 5.1.2 Interaction explained in detail

Gestures such as tapping, swiping and finger scrolling are required as we will be removing the mouse as an input device. Right clicking could be replaced with 2 finger or 3 finger tap. These interactions could be modified with settings to resemble trackpad gestures if that is what the user is more familiar with such as 2 finger scrolling, 3 and 4 finger swiping.

All of these touchscreen/trackpad style gestures would be applied to the application and can be directly translated to an application window running outside the environment. As mentioned in the feature list we would like to be able to replace the windows shell interface. We have done some research and discovered it is indeed possible to capture the 2d render stream of an application window. With this we could make all native windows applications appear in the Gesture Shell and send all interaction interpretations to the applications themselves. Translating a tap’s relative x & y coordinates on the texture stream to be a mouse click in the applications window.

Resizing of windows. The resizing of application space and not the objects that make the window models is required. And this window resize needs to be sent to any windows native application that is being streamed into the shell with corresponding adjustments to the texture streaming the window.

Virtual Keyboards are required to fully interact with standard applications unless the user opts for voice inputs. Although we have not attempted to implement our own we have tested others such as one created by Joshua Corvinus (see references) that can function fairly well. A lot of work goes into counteracting the lack of hand tracking accuracy. Keyboard layouts could be fully customized however and we envisage that virtual keyboards would become a development area of its own. Keyboard input would be much slower at first but with future developments in hardware and software virtual keyboard input could surpass the traditional method.

Assisting factors for virtual keyboards:

* Text prediction
* Voice recognition
* Hand writing recognition (possible using a gesture or grabbing a virtual pen to trigger the input)
* Community is likely to design new layouts for 3d virtual keyboards to become more viable.

## 5.2 Technical Development

We have managed to implement grabbing/positioning and rotating of mock applications windows in the 3d environment. We have toyed with single hand rotation but found using 2 hands to rotate a window to be more intuitive. We have also implemented a close button for the windows. The script for the close button is designed so that it will find the parent of its parent and hide it. This design enables the close button to be part of a title bar prefab that can be instantiated for each application that the user desires to open. The close button is a 3d object that detects collision. The script is only triggered by a collision with the most distal bone model of the index finger of either hand contained within the Leap Motion hand control Models. Resizing of objects using two handed pinch was tested and became problematic so it was disabled in all of the test scenes. A great deal of time would need to put into scripting a new resizing script in order to make the application space change in size and not just scale the objects the make up the application window as a whole.

The most time consuming faze was getting the Leap Motion interaction to work in unity to start with. By including prefabs from the Leap Motion standard assets collection in the Orion SK. And importing custom assets form the concept design scenes and attaching pinch detection scripts from additional Leap Motion assets we managed to get the global environment interaction to work.

## 5.3 3d Models and assets.

### Custom Assets

* Application Window Prefab: A simple collection of cubes cube to be instantiated and acts as the parent to any application UI elements for mock application construct. Has a rigid body component and a pinch detection script from the Orion Self-Development-Kit. Anything that is included as a child of this object can be manipulated with global gestures.
* Browser\_UI Prefab: Early version of the mock browser window with an input box for the address bar.
* Show\_Browser Prefab: An invisible cube that hides the applications list and shows the mock browser when a collision is detected.
* Close Button Prefab: A simple 3d cube with accustom script as described in section 5.2.
* UI Canvas: Set with static 3d world positioning to enable UI elements such as input boxes and text to be coupled with 3d objects.

### Assets from outside sources

* Screenshot was taken of google chromes new tab page to use for the mock browser window.
* Leap Motion Assets.
  + Leap Motion Controller prefab with bone style hand models.
  + Pinch Detection Scripts from additional Leap Motion developers.
* A world image taken from findicons.com

### Test Scenes (in the tests folder of the unity project)

**NOTE:** These scenes are designed with head mounted orientation for use with a VR headset. When trying out these scenes position the leap motion controller in the head mounted position.

* + Single Hand Rotation: Was found to be much harder to control.
  + UI Tests: The basic idea but uses the mouse for interaction.
  + Two Handed Rotation:
    - Much easier to rotate and position windows.
    - Browser Button and close button works.
  + 2 Blanks: Just 2 blank windows with 3d button that switch between them.

# User Feedback

The main concern brought up in the presentation of the concept was the accuracy of control. Weather hand gestures could accidentally trigger events in other windows, and if application windows could end up out of reach in the 3d environment.

To combat this lack in hand tracking accuracy we would need to recognize hand positions as gestures that enable/disable the ability to interact in certain ways. We have managed to make buttons only detect collision with specific index finger / pointer bone section of the hand models only to prevent accidents.

# Conclusion

This kind of interface really requires an advancement in the hardware technologies to become a reality. Increased accuracy in hand tracking and higher resolution displays in VR headsets. False pinches are often detected mainly due to inaccurate tracking by the leap device. Trying to click a button and the hand model all of sudden resembles the pinch position despite the user’s hands not being in that position at all. Also a lot of the development in unity is hardware targeted. Making an interface that can use any hand tracking or VR hardware would require specific assets, and tailored scenes.

Capturing a 2d rendering stream from windows application is possible, however many of the developers that have achieved such task in their own applications have kept their methods a secret and exactly how to go about achieving this is still unknown to us.

An application like this will be incredibly difficult to implement and may require a large development team with specialized skill sets.

# References

FindIcons, 2016, *Globe Icons*, Accessed 12/05/2016, <http://findicons.com/icon/66568/globe>

Google, 2016, *Google Chrome*, Accessed 12/05/2016  
<https://www.google.com.au/intl/en/chrome/browser/desktop/index.html>

Leap Motion, 2016, *Leap Motion Orion* SDK,Accessed 15/05/2016 <https://developer.leapmotion.com/orion>

HocGaming, 2015, *Sword Art Online GUI DevLog 1-5*, Accessed 12/5/2016, <https://www.youtube.com/watch?v=jLp3W1gbhRk>

Virtual Keyboard by Joshua Corvinus

Available at https://github.com/jcorvinus/VRKeyboard