Gesture Based URI Development

**Name of Game:** Viking Run

**Game Type:** 3D Endless Runner

**Gesture Technology used:** Microsoft Kinect V2, Voice control

**Group Members:** Aaron Hannon (G00347352), Matthew Sloyan (G00348036), Michael Coleman (G00347650)

**Github Link:** <https://github.com/aaronhannon/GestureBasesUIProject>

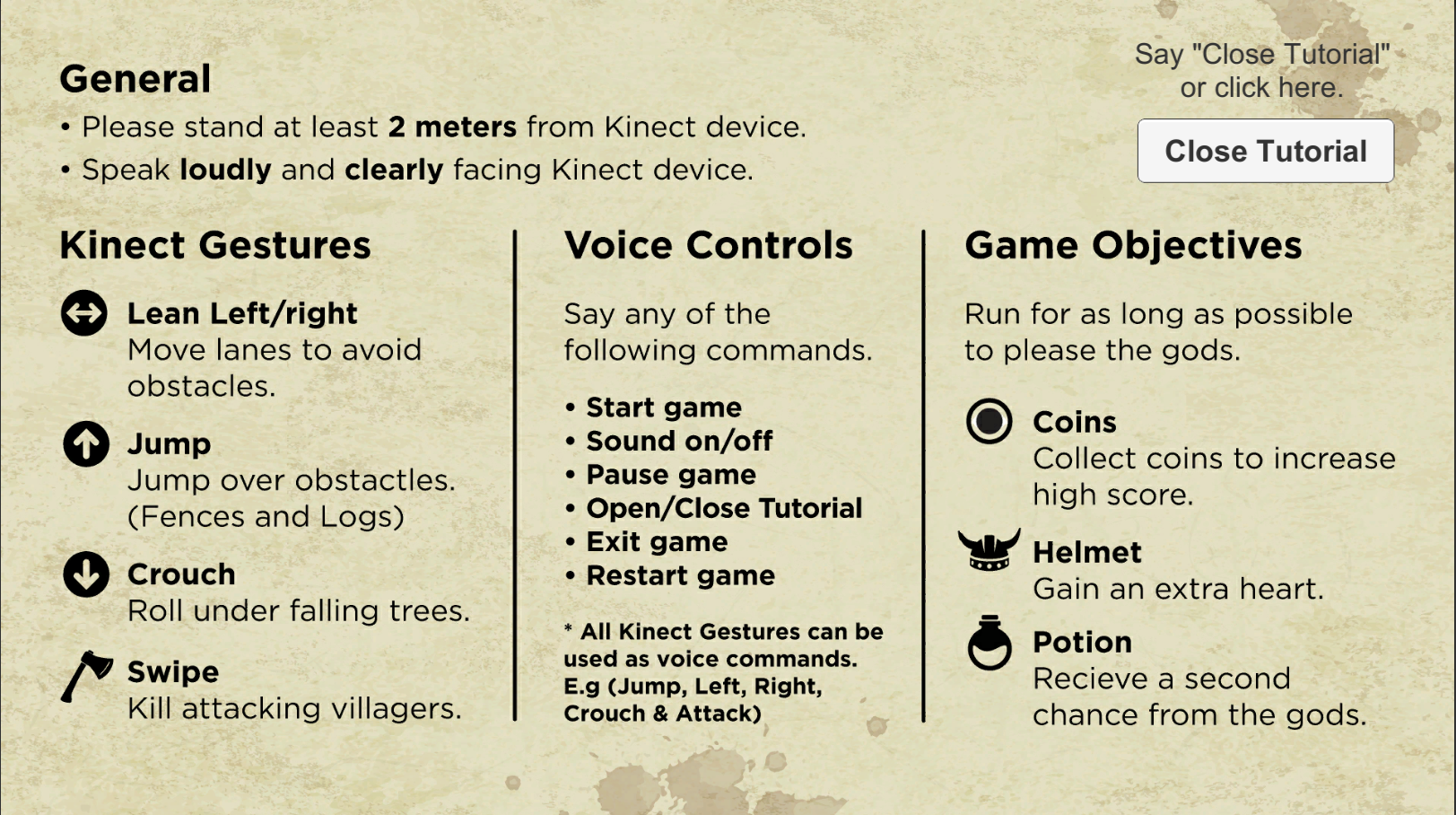
## **Purpose of the application:**

The purpose of this application is to produce a 3D endless runner in Unity which is controlled using the Microsoft Kinect V2. The game should allow the user to navigate through the game by controlling the character via a specified selection of gestures. The game should have the ability to control menu options via voice commands using Windows speech library within Unity. The UI should include a tutorial of the controls needed by the user. The reason we chose this application was to create a fun experience for the player and to improve our ability to work with hardware within game development. This is why we chose to incorporate two different methods of gestures rather than one. We also took this opportunity to work with Unity 3D as we only had experience creating games within a 2D workspace. This would also provide us with a greater challenge due to the more complex nature of 3D game development e.g. physics systems.

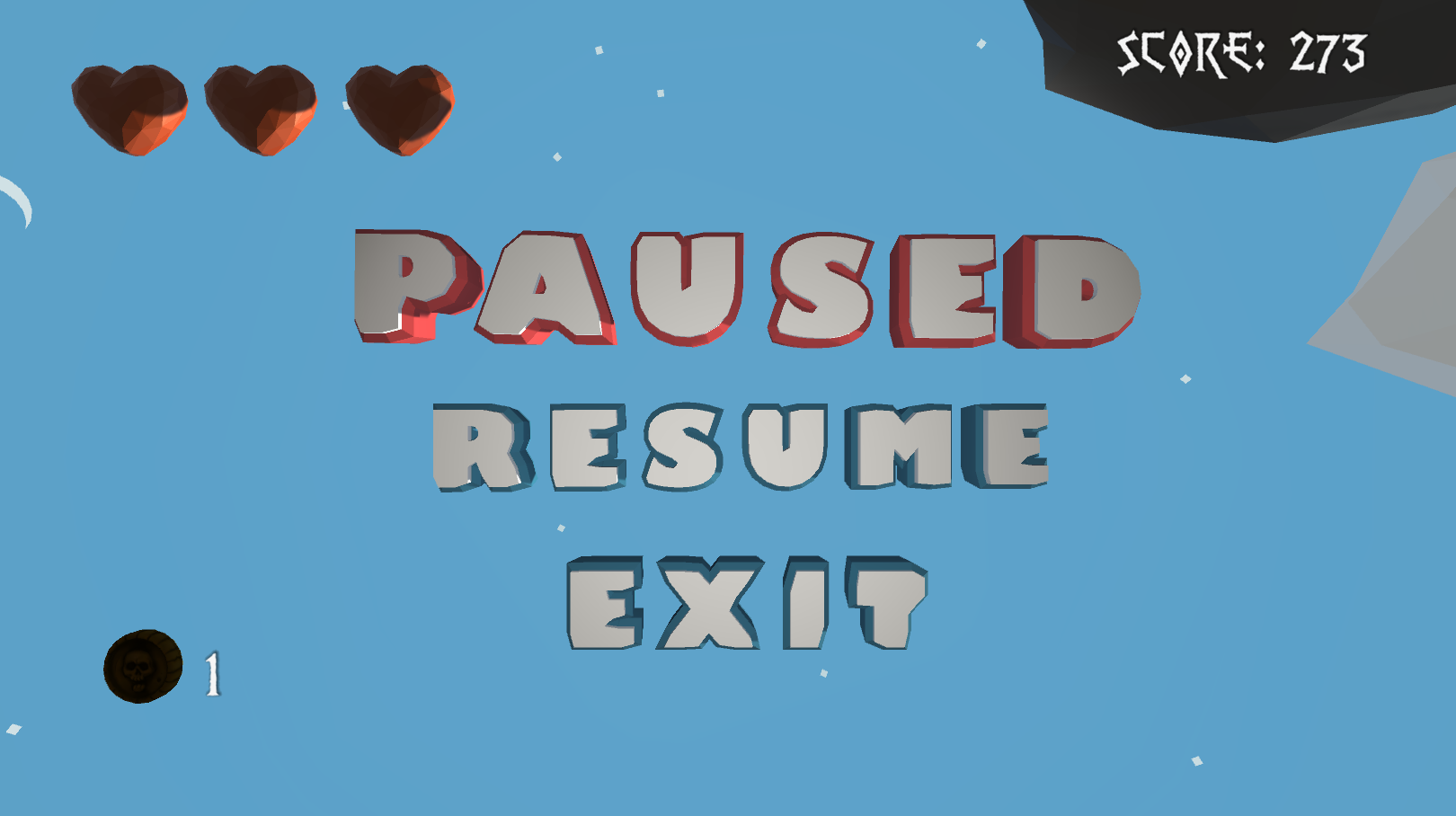
We decided that the UI should be fully voice controlled, that is removing all necessity for the user to have input controls. The user has full access to every feature in the game via voice commands. The controls for the voice commands to access the menu are detailed in the next section under the Voice Recognition heading. The user is thrown into the game where the menu is incorporated into the same scene as the game. This allows for a dynamic background making for an aesthetic view for the user.



The user is given prompts to say their name for saving their score and a prompt of how to access the tutorial with all other commands. The tutorial shows a pop up menu on screen giving the user an overview of all controls in the game.



The pause menu also can be accessed when the game is started via voice command. This pauses the character and NPC movement and pans the camera up into the sky with a dynamically moving pause menu which moves with the player.



When the player dies a dynamic death scene representative of Valhalla (Viking belief similar to heaven) is transitioned to. This also has a UI element which displays highscores, as well as the current user score.



At the beginning of the development process due to having to share technology with other groups we also fully incorporated controls using the keyboard. We decided to incorporate voice commands after we had tried and tested the kinect and ensured that they would be compatible with using the kinect. After ensuring they were compatible, we converted all the keyboards UI controls and game controls to voice commands. As a result, the game is controllable using the techniques: Kinect Gestures, Voice Control Gestures and Keyboard and Mouse Controls.

The physical Gameplay results in the player running infinitely in a direction with many obstacles getting in their way. There are 3 different chunks which spawn in the game. A village, a river and a forest. Each chunk provides different challenges for a user. A village has fences and moving NPC’s (Non playable characters), A river results in the player getting into a boat and moving at an increased speed. The river has rocks which they need to avoid. The forests contain logs on the ground as well as falling trees on either side which they need to avoid. Players have many different methods to avoid these obstacles which are listed in the gestures section. There are many collectables throughout the game, coins can be collected as a multiplier to the user score. Revives can be collected which allow the user to get a second chance when they lose all their hearts. They also can collect helmets which allow them to take an extra hit of damage without losing a heart. When the user dies their score is shown and compared to the highscores and added to the high score scoreboard if it's in the top 3 high scores.

## **Gestures identified as appropriate for this application:**

Another focus of this project was to develop something that incorporated gestures but also served the purpose of being fun and easy to play. In order to achieve this we needed to incorporate simple and intuitive gestures which would allow the player to prefer using gestures rather than a standard controller/keyboard. We initially took it upon us to research different areas of gesture based technology and hardware. Some of the topics we looked at included voice recognition, full body tracking (Kinect), simple hand gestures (Leap Motion Controllers) and arm gestures (Myo). A breakdown of the hardware and their benefits can be found in the section below. Based on our research we decided to choose full body tracking and voice recognition. Firstly due to the fact our game would be an endless runner full body tracking fits perfectly with the motions that are standard to a game of this genre, such as jumping, moving and crouching. Secondly it provides a means of exercise by gamifying it. We also chose voice recognition as it goes well with Kinect gesture tracking, as the player can easily enter commands while they are playing the game without the need to pick up a controller, so it allows a full hands free experience. This also increases the safety for those around. Below we will look at some of the gestures we have developed, our reasons for implementing them and our testing efforts for each.

### **Kinect**

With the Kinect set up with Unity we began to test the inbuilt gestures, however we felt it would be best to create our own unique to our game. We recorded all three of our bodies doing each gesture in multiple different ways to get a well rounded result. With this we had a full suite of the below gestures to incorporate into the game. We wanted to make sure they were all unique to each other, so that none would overlap. Also, we emphasised ease of play with simple gestures that anyone can complete.

1. **Jump**

To trigger this gesture the player must jump off the ground. We used a confidence level to achieve this as sometimes it could be seen as a crouch.

1. **Move Left & Right**

When the player leans left or right the correct gesture is triggered, which moves the player into the left or right lane to avoid obstacles.

1. **Crouch**

If the player physically ducks down or crouches the player will do a forward roll which doesn’t break momentum. It made more sense to do a roll in the game as the player is constantly running, but a roll wouldn’t have been a suitable gesture so instead we implemented a simple crouch.

1. **Attack**

The player can swing either their right or left arm to make the player swing their axe to kill villagers.

### **Voice Recognition**

As mentioned above to continue the hands free and ease of play experience we decided to incorporate voice recognition as our second gesture system, which merged nicely with the already developed Kinect gesture controls. We thought it would suit well to trivial tasks such as pausing the game, turning off the sound etc which could easily be spoken while jumping, attacking and leaning. From this we developed a dictionary of the most common phrases for each task increasing accessibility. All of these other options can be found below along with the task.

Also all voice commands are implemented for the all above Kinect gestures as another means of playing the game, so it is accessible to all players. For example if a player doesn’t want to or can’t jump/crouch then they can use voice commands or keyboard inputs. Example inputs are as follows (jump, left, right, crouch, roll, attack, swipe).

1. **Start Game**

Other variants - start game, play game, start, play, begin, begin game.

1. **Restart Game**

Other variants - reset game, restart game, reset, restart.

1. **Exit Game**

Other variants - exit game, close game, exit, close, quit game, quit.

1. **Pause Game**

Other variants - pause game, unpause game, pause, unpause, resume, resume

Game.

1. **Turn on/off sound**

Other variants - sound, sound on, sound off, turn on sound, turn off sound, turn sound on, turn sound off, volume on, volume off.

1. **Display Tutorial**

Other variants - open tutorial, close, tutorial, display tutorial, show tutorial, open guide, close guide, show guide, tutorial, guide, help.

1. **Name Input**

At the start of the game the user can input their name by voice command which will be saved to the high score at the end.

## **Hardware used in creating the application:**

### **Kinect**

### **Voice Recognition**

### **Other technologies we looked at:**

**Myo**

**Leap Motion Controllers**

## **Architecture for the solution:**

## 

## 

## **Conclusions & Recommendations:**

This was a great project as it tested our abilities in a new area which we had no familiarities with. As a collective group we had zero experience working with the Microsoft Kinect and any of its associated scripts. We also had no experience in 3D game development. Two members of our group had experience with voice recognition in unity but not using the windows speech library provided by unity.

This new combination of new technologies provided us with a great challenge as working with new hardware will always provide many issues which a good knowledge of the inner workings of the hardware are required to develop a solution. This was definitely evident when using the Kinect. Many challenging aspects were encountered and limitations to the technology needed to be adhered to when coming up with solutions for mechanics within the game. Areas such as training gestures required even more external tools and methods to incorporate our custom gestures in the game. This provided a challenge as many users carry out the same actions in different ways e.g. some people may jump with their legs bent while others may not, so it was crucial during training to carry out the actions in as many ways as possible.

The physical scripts associated with the Kinect for unity, took a while to get our heads around. Also tutorials and documentation for the Kinect is very limited. This was great in a sense that it forced us to get stuck into the code and try to decipher for ourselves how it works and how we could adapt it to carry out the tasks we needed.

The voice controls on the other hand was more simple to get a good grasp on. Unity provides a lot of good documentation for this and it works quite well. The two types of speech recognizers we used cannot be active at the same time. This was a software limitation on what we could do. Unity only allows users to have one active at a time. So in order to use another, the whole system that was active needs to be shut down. We worked around this quite well by ensuring the user gives their name first then other voice commands are enabled. One issue with voice commands in unity is that there is a slight processing delay so commands are not instant like they are with the Kinect. The phrase recognition system must take the user voice input and match it to a specific keyword. Although still quick in a fast paced game like this, small issues such as this could be crucial to the user living or dying.

The game development proved rather tricky as 3D game development adds a load more areas for things to go wrong in comparison with 2D games which we all had previous experience with. For example, 3D physics provides a much greater challenge to get correct due to the many different aspects added in a 3D world. Collisions provided issues as well as areas such as animation creating proved to be a time consuming process to get perfect. Nevertheless, through a lot of testing we found and fixed the majority of bugs that cropped up during development and provided a game of a pretty high standard.

If we were to start this project again, firstly choosing a hardware which we had more frequent access to would be a great idea as it may have allowed us to try incorporate more advanced methods of using the Kinect, perhaps areas such as multiple players could have been looked at. Another area to look at if we had more time to develop would be to add more gestures and obstacles to the game. Also perhaps with more time, we could have created winding paths rather than straight ones. It may be to have been possible to have added more hardware e.g. Playstation Move controllers to act as the players hands while the kinect could control the player movement. This would provide a variation on virtual reality without the need for the expensive headset.

We certainly would recommend this project to anyone with an interest in gesture and game development as we thoroughly enjoyed our experience developing this game as well as vastly improving our research ability into developing with many different areas of hardware. Not only did this project provide us with the knowledge of working with voice controls and Kinect gestures, we also learned about many other hardwares and how they contrast in their implementation. This project provides a great understanding into working with external hardware and the cohesion process which needs to be carried out in order to combine their functionalities.