The University of Hong Kong
Bachelor of Engineering
(Computer Science)

COMP 4801
Final Year Project

Final Report - Individual

Open World Virtual Reality
Role Playing Game

Supervisor
Dr. T.W. Chim

Submission Date:
16 April 2017

Lau Chui Shan
3035097325

Group FYP16025

Group Members
Wai Yip Yin Calvin 2008993564
Hui Sen Fung Felix 3035090327
Leung Ho Ki Tony 3035097210
ABSTRACT

Virtual reality (VR) game is a new kind of games that creates a boom in recent years because of the rising attention of VR technology from the public. Yet, the existing open world VR games may not provide enough satisfying gaming experience to players as they either impose logical limitation to the space of the game world, or require a large gaming space in reality.

At the same time, it is found that nowadays public is lacking skills for surviving in field. It is valuable to develop a game that can let players learn survival skills from the game. Therefore, this project is to develop an open world VR role playing game with topic of survival.

As a result, the team developed a open world VR role playing game with Kinect as movement input. To ensure safety of player, a platform was also constructed. This report will further explain the background, objectives, scope, methodology and results of the project.
ACKNOWLEDGEMENT

I would like to thank Dr. T. W. Chim for being the supervisor of our final year project. Dr. Chim gave us a lot of inspiring suggestions on the direction of our project. No matter what direction we chose, Dr. Chim offered help for us by giving suggestions on how to make the project better. It is my pleasure to have one more chance to be instructed by Dr. Chim.

I would also like to thank the Computer Science Department for providing the Kinect for development. Without the Kinect, we cannot proceed the project.
# Table of Content

1. Introduction 5  
2. Objectives 5  
3. Contribution 6  
4. Previous Works 6  
   4.1 VR Entertainment 6  
   4.2 VR Locomotion 7  
   4.3 VR Game 8  
   4.4 VR Training 8  
5. Scope 8  
   5.1 Game Development 8  
   5.2 Locomotion Method & Input Control 9  
6. Methodology 9  
   6.2 Game Development 9  
   6.3 Locomotion Method & Input Control 11  
7. Results 12  
   7.1 Game Development 12  
      7.1.1 Game Mechanics 12  
      7.1.2 Player 14  
      7.1.3 Environment 18  
   7.2 Locomotion Method & Input Control 23  
      7.2.1 Movement Posture Study 23  
      7.2.2 Input Method Study 23  
      7.2.3 Platform 24  
      7.2.4 Gesture Recognition 25  
8. Conclusion 26  
9. Future Development 27  
   9.1 Game Development 27  
   9.2 Locomotion Method and Input Control 27
## List of Figures

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Description</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Final model of the player character and its skeleton</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>Lower arm with cylinder of wound material and bandage material</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Interface for showing player survival parameter</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Interface for showing task instruction and player survival parameter</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Interface of inventory menu</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Whole view of the island</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>Intermediate prefabs of shelter</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>Baked navigation mesh</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>Navigation mesh with nav mesh obstacle exists</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>Structure of platform</td>
<td>24</td>
</tr>
</tbody>
</table>

## List of Tables

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Description</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project Contribution</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Tasks designed</td>
<td>13</td>
</tr>
</tbody>
</table>
1. Introduction

In recent years, virtual reality (VR) seems to have a rapid development because of the easy installation of VR headset with mobile phone. With VR implementation, users can have a virtual but realistic 3D sight produced by the VR application.

However, even the VR application produced a realistic 3D sight for the user, as limited by the physical gaming space and equipments other than the headset for gaming are not yet widespread, current VR games gives unnatural interaction between the game world to the players. Therefore, it does not provide an immersive VR gaming experience for players.

The team understands that public is expecting a realistic interaction with objects in the virtual world because the under the highly developed graphic rendering, objects are realistic as in the real world. As a result, the team developed a VR game that aims to give high immersive experience to the player. The game was developed to give natural interaction to players but at the same time the gaming space is not limited by the physical space. ‘Open World’ in the project title here means even the physical space that the player can move is small, the world that they can explore in the game world is opened and large. To achieve this, not only motion capture system is needed, but also a tool for keeping the player in the same position in real world is needed.

On the other hand, the team found that nowadays in a living place with abundant resources, general public is losing the skills to live in a harsh environment. Therefore, the team considered to choose survival as the game topic of the role playing game (RPG). While playing the game, players are able to learn skills to the live in field with limited resources.

2. Objectives

With the limitation to current VR game development and lack of survival skills by public, the team decided two aims for the project.

- To develop a VR game that allows players to explore the game world with neglect of physical space in real world.

- To educate players basic survival skills like making fire and making shelter in a safe environment
3. Contribution

Members in the team contribute to the project as follows.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>UID</th>
<th>Contribution</th>
<th>Sections Related</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wai Yip Yin Calvin</td>
<td>2008993564</td>
<td>- Game logic programming</td>
<td>7.1.1 (Survival Parameter), 7.1.2 (Gesture Handle), 7.1.3 (part of Game)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Code review</td>
<td>(part of Task Gesture Handler in Non-moving Objects), 7.2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Platform development</td>
<td></td>
</tr>
<tr>
<td>Hui Sen Fung Felix</td>
<td>3035090327</td>
<td>- Gesture recognition module development</td>
<td>7.1.2 (User Interface), 7.2.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Input control design</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Locomotion method implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- UI development</td>
<td></td>
</tr>
<tr>
<td>Leung Ho Ki</td>
<td>3035097210</td>
<td>- Background story writing</td>
<td>7.1.1, 7.2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Stage design</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Skill research</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Platform development</td>
<td></td>
</tr>
<tr>
<td>Lau Chui Shan</td>
<td>3035097325</td>
<td>- 3D modeling &amp; animation</td>
<td>7.1.1 (Detailed Tasks, Inventory), 7.1.2 (except Gesture Handler), 7.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Game world interaction scripting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- AI scripting</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Project Contribution.

4. Previous Works

4.1 VR Entertainment

VR is an environment generated by computer to show 3D objects in the virtual world as they in the real world. By separating view for left eye and right eye, when the user put on the VR headset, they can see objects like in the real world.

VR entertainment was first made available to domestics back in the mid 80’s [1] by giving player controls over the camera angle with head movements, enabling the player to look around in the digital world at a certain point or a preset routine. However at that time, player cannot control the view other than angle. Therefore, different locomotion method of VR were then introduced.
4.2 VR Locomotion

In order to give better user experience to VR users, different locomotions were done to simulate the gesture of users in the virtual world. Based on study done on these locomotion, it showed that each of them has certain strength and weakness. [2]

The first locomotion for VR is simply by the sensor of the heading set. Users can interact with plots in the virtual world by moving the headset, changing the value of sensors, and sometimes the headset is connected with a simple remote control which get users’ input of buttons. This locomotion is easy for implementation and just need small space for gaming. The users can also explore the game world quickly because the speed of action in the game world is not limited by the action in real world.

The second locomotion added movement detection to the first one. Using this locomotion, users move and stand in the virtual world by stepping and standing in the real world. The direction of movement is changed by the direction of a hand-holding remote. Same as the first locomotion, this locomotion only need small moving space in the reality since the users are merely encouraged to step but not move forward or backward. On the top of this, this locomotion is more immersive than the first one because the character movement is controlled by corresponding body movement in the real world. However, the user experience is still unrealistic as the movement in virtual world is simulated according to the remote and stepping movement, so the view in virtual world cannot show exactly what the users expect to see.

The third locomotion takes spacial detection into consideration. HTC Vive is an example using this locomotion. Users can move freely in a specified space. The position and direction of the users are detected by the system so that the system can show the sight with respect to the position of the user, which is much more immersive than the previous two. Besides, different body gesture such as squatting can also be detected, such that users are not easily to get motion sick. Yet, this moving space is limited as the area that can be detected.

The last locomotion is Virtuix Omni. Virtuix Omni includes a treadmill for user to walk on. The direction of the users is also detected by the machine. Therefore, this locomotion is highly immersive and is believed to be able to give great user experience. Nonetheless, Virtual Omni is a large scale machine which require lots of space. The cost of it is also high so this is not recommended for general public.
4.3 VR Game

Although open world VR games and VRRPGs are not the first time to come to the game market, the combination of these three is seems to be new to this market. Yet, some computer scientists tried to mod open world RPG with VR technology, producing open world VRRPGs. [3] Grand Theft Auto V is an example of this kind of open world VRRPGs. Grand Theft Auto V is originally an open world RPG. After installing the third-party mod that mods the game with VR, players can enjoy the realistic VR experience when playing the game. However, as third-party mod is used to achieve the VR effect, the game may lose performance, giving unsatisfying gaming experience in the aspect of smoothy.

4.4 VR Training

For some jobs with high risk like military and aviation, VR has been used for training purpose. By using VR, dangerous situations can be simulated and the trainee can practice how to cope with these situation under a safe environment. On the other hand, VR is used for training on the work which may cause harm to others. For instance, driving and doing surgery. Training by VR can ensure even trainee make fatal mistakes, nothing will be damaged or hurt. These show that VR training is worth doing.

5. Scope

5.1 Game Development

In the project, a Open World VRRPG should be developed. To let the players learn survival skills, players are required to perform those skills while they are being well instructed in the game. As the game is open world that players can explore game world without limit of space in real world, the map of the game world should be made large. To keep the playability of the game, besides instructing players to perform survival skills, the game also needs a story so the players have a ultimate goal in the game. This should be implemented together with a quest system. The final product of the game should be implemented with VR.

However, due to the one-year-time limit of the project, the team decided to only develop the tutorial tasks that let the players to learn basic survival skills. A smaller but still large enough map was decided to be used for the tutorial tasks.
5.2 Locomotion Method & Input Control

A motion capture system should be used to capture the movement of the player. The movement should be interpreted as different gestures such that they can call different handler to handle the gesture in the game. To raise the immersiveness of the game, the team should develop a locomotion system which can interpret the lower limbs movement of the player to navigation input. To keep the player safe and keep their moving area small in the real world, a platform was constructed. The whole system should be in small size, easy to install and affordable by general public.

6. Methodology

In this project, KinoVR was decided to used for generating VR views. KinoVR is a VR streaming application that connects the computer to mobile phone. It can translate the computer view into VR view automatically, while the movement of the mobile phone is detected to change the angle of view of the computer side application. Therefore, the team just focus on the game develop and locomotion method and input control.

6.2 Game Development

The game development was mainly divided into 4 phases, namely analysis, design, implementation and testing. They were done accordingly.

Analysis

In the analysis phase, identifying actions that can be captured by the chosen locomotion system, and identifying key elements for a survival game was done. Actions able to be captured had to be identified first because the results are important for designing the task. This was done after the choosing the locomotion system for the project. Simultaneously, the team decided the prime requirements of a human being for survival, such that the tasks progress based on these requirements.

Design

In the design phase, game mechanics and detailed specification of tutorial tasks were designed. Based on the findings of requirement for survival from the previous phrase, categories of status that used in the game was chosen as the survival parameters. The mechanics on how to make changes to these status was also decided. Besides, according to the actions identified, content of the tasks was determined. To ease later development, actions and objects involved were listed for each tasks.
Implementation

The game was developed using Unity 3D. The implementation of the game was further splitted into 3 parts, which were game mechanics, player and environment.

For the part of game mechanics, the planned mechanics were decided to be implemented directly by scripting.

For the part of player, it required building of the 3D model of the character by Blender and MakeHuman. Then the character needed to be scripted to handle the gesture returned by input, such that the character is interactive to the game world. Last, user interface had to be implemented for the player, such as showing the survival status and different kinds of menu.

For the part of environment, it represented the background environment for the character and every interactive objects for the players. To obtain all the object models needed, the best way is to find resources online and apply them directly to the game with permission, because they are not the main focus of the project and time can be saved if using existing models. As long as the models are small size, low polygons and suitable for the task, we can consider to use them in the game. As the background environment is static, no scripting was needed for it. Yet, for interactive objects, in order to let them interactive and behave how they should be, scripting was needed.

Testing

Before establishing locomotion method, the game was tested by a first person character which controlled by mouse and keyboard input. Objects interaction was tested by keyboard input to check the functionality of every public method. As the scripts are object-oriented, this was easily achieved. After establishing locomotion method and motion capture system, test on whole game flow was done.

Export

Finally, the game was exported as a computer application. With the help of KinoVR, the VR view of the game will be generated for the mobile application.
6.3 Locomotion Method & Input Control

Same as game development, development of locomotion method and input control was also divided into phases of analysis, design, implementation and testing.

Analysis

Two analysis were done before design the locomotion method and input control, namely movement posture study and motion capture system study.

For the movement posture study, postures of a human being when doing different action is captured. The data was used in designing the platform as the platform needs to ensure safety of players whenever they are doing any actions. When recognizing gesture from movement, these data was also used.

For the motion capture system study, the team investigated some kinds of sensors and compared them by functionalities and price. One of these systems was chosen as the motion capture system in this project.

Design

With regard to the data collected from the movement posture study, a platform was designed to keep the player safety. On the other hand, locomotion method algorithm was designed based on the motion capture system chosen and its limitation.

Implementation

For the platform, as this is a concrete object in real world. The team needed to purchase the materials for the platform and construct it as designed.

For the locomotion method and input control, implementation was done in two main directions, navigation direction and gesture recognition. Navigation direction means getting the direction of moving of the player. Gesture recognition represents interpreting movement of player into gesture in the game. This was done in Unity.

Testing

Last, testing on the platform focused on keeping the player safe. Testees were required to do different actions including fall down the test the reliability of the platform. Also, if the testees were kept at same position of the platform or not was checked while testing.

To test the locomotion method and input control, testee were required to do different gestures. If gestures were recognized, a debug log was output in the Unity console. The test check the response time and accuracy of recognition.
7. Results

7.1 Game Development

7.1.1 Game Mechanics

Game Story

Although the storyline was not applied to the game in this project because we only develop the tutorial tasks for the game, a story was designed to ensure the tutorial tasks suit it.

The story starts from a solar storm. After the solar storm, most of the power plant are destroyed. At this time, the upper-class dominant the power plant and only distribute small amount of electricity for the lower-class.

The player acts as one of the lower class in the game. Due to the limited amount of electricity, electrical devices can not be used and the player has to survive in this harsh environment. By the progress of the game, the player will be given a goal to fight for fair use of electricity.

Survival Parameter

The team decided to have 4 types of survival parameter for the player in the game, namely hunger, thirstiness, injury level and health point. Hunger and thirstiness drops per in-game day. They can only increased by eating and drinking respectively. Injury level is depends on the health point. There are 5 level of health, including healthy, 3 level of injury and dead. Normally the player stays in level of ‘healthy’. However if the player gets hurt and the level of hunger and thirstiness keep negative, health point decreases and leads to a worse level in injury level.

A script is attached to the player character to store and calculate the value of these status. When they change, it will also change the value shown in the user interface.
Detailed Tasks

There are in total 5 tutorial tasks. They were decided under the consideration of fulfilling the basic need of living. For each task, multiple missions have to be done by player accordingly. When the task is completed, the level of player status can be raised. The first tutorial task is for letting the player to know how to control the character so it requires the player to merely pick up objects and put down objects. The following tasks then further requires the player the build shelter, make fire, hunting and do first-aid. Different actions are required to be performed by the player.

<table>
<thead>
<tr>
<th>Tutorial No.</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Collecting Resources</td>
</tr>
<tr>
<td>2</td>
<td>Building Shelter</td>
</tr>
<tr>
<td>3</td>
<td>Making Fire</td>
</tr>
<tr>
<td>4</td>
<td>Hunting</td>
</tr>
<tr>
<td>5</td>
<td>Basic First-aid Treatment</td>
</tr>
</tbody>
</table>

Table 2. Tasks designed.

As for each task, the player is required to collect different kinds of objects, the environment setting may be different. Therefore, a static script was implemented to control the whole game flow. The script allows setting or resetting the environment. In order to save the initial state of the environment for resetting task, the environment of the current task is copied at the beginning of the task, and only the copy will be used. When the game progresses to the next task or the task is reset, the previous copy will be destroyed. For the objects that keep existing in the scene like shelter and fire, they are excluded in this process. Besides the environment, the inventory of the player also needs to be reset for every task. Hence, the script will call the inventory to clear itself. For tool objects that should be kept by player for every task such as knife and bucket, they are added to the inventory again after clearing the inventory.

Inventory

The player is allowed to save the pickable objects into their inventory. Therefore, a static script was used for the inventory so that it can be access by any other scripts. In the script, it holds an arraylist of Pickable objects. Each Pickable object store the data of the gameobject, the object name, amount it represents and its image. Arraylist is used because it can be dynamically extended and reduced, and also allows direct access. The script includes public functions that allow other script to add, remove or get objects from the arraylist. It also allows to clear the whole arraylist when the task is restart.
7.1.2 Player

3D Model

3D model of the player character was produced by using MakeHuman. In the ideal case, the model exported from MakeHuman can be directly used by Unity 3D. However, after testing the model with Kinect, the team found that the head of the model would obstruct the model from performing the captured motion. Therefore, the model was further modified by Blender to remove head part. As to control the orientation of the camera, the skeleton of the head was kept.

![Figure 1. Final model of the player character and its skeleton.](image)
Wound and Treatment

In the task of first-aid treatment, wound shows on the lower arm of the player character. To visualize this, a cylinder was externally added to the gameobject of the character. A cylinder rather than a plane was chosen because the cylinder can suit the curve of a lower arm. Also, it can be reused for showing result of treatment.

To show the wound, material with a wound texture was used. The material was set to be transparent such that only the part of wound in the image was shown. To show the treatment, material with bandage texture was used. The bandage texture filled up the curve surface of the cylinder.

A script was attached to the player character to show the wound and bandage. Since the cylinder gameobject is deep under the character gameobject hierarchy, the cylinder gameobject was set as a public variable such that the script can access the gameobject easier. Besides, the material of wound and bandage was set publicly. When the script start, the cylinder was set to be disabled because both the wound and bandage should only be shown in the task of first-aid treatment. When the script is called by game flow control script to show the wound, the script set the cylinder to use the wound material and enable the cylinder. When the script is called to show bandage, the script change the material of cylinder to the bandage material.

Figure 2. Lower arm with cylinder of wound material and bandage material.

Gesture Handle

When the gesture of player is recognized by the gesture recognition module, gesture handler is called. According to the gesture recognized, corresponding functions are called to respond to that kind of gesture. Besides, a public function that return the position of a specified joint is written in this script.
User Interface

Finally, user interface was added to the player such that player can know the player status and control the game. There are 3 kinds of user interface, including player status, game instruction and inventory menu.

For player status interface, it is always shown to the player. The value of survival parameter is shown.

![Figure 3. Interface for showing player survival parameter.](image)

For game instruction interface, it shows the instruction of each mission in the tutorial task. It shows another instruction when the mission is complete.

![Figure 4. Interface for showing task instruction and player survival parameter.](image)
The inventory menu shows the objects that picked by the player. It was made by the UI component of Unity 3D. The menu can show 3 objects at most at the same time, and it can be scrolled horizontally. For each object, a pre-captured image and the name of the object is shown. When the menu is opened, the player can also choose to hold or drop the objects.

To implement this, the inventory menu holds an independent script and the functions are called by the player script. The script holds the index of the current selected object and an array of the data of the 3 objects being shown. When it is called to open the menu, the data from the array is retrieved and sets the images and the name of the objects shown. Whenever it is called the turn the menu left or right, the script first shift the two objects that keep showing in the array and change the index of current selected object. Then it calls the inventory for the new showing object by its index. When the script is called to hold the object, it will first determine which hand should hold the object by the principle least recent use. If the hand has already hold something, that will be put back to the inventory. Otherwise, the script will call the object to be held. The inventory will not be changed no matter putting object to hand or storing in-hand object back to inventory because they are still owned by the player. If the object is dropped, it calls the inventory to remove that object by index, and call the object to be dropped.

Figure 5. Interface of inventory menu.
7.1.3 Environment

Background Environment

Based on the setting of game background and tasks, the background environment requires forests for tree log and river for clean water. As this is a survival game, the island is also needed to be undeveloped. Then, the team tried to search for model of a whole island that fulfills these requirements. However, there were no model that can fulfill all the requirements. Therefore, the team decided to build the island by the terrain function in Unity 3D.

Outside the terrain component, it is surrounded by water plane of Unity. Since this is just a tutorial, the moving area of the player was decided to be limited. To achieve this, the moving area is surrounded by cliff or slope, such that the player cannot go to those area. The land was also partially lowered for the river. The shape of the island and the forest can be made by terrain of Unity. In order to let the player feeling the island is large, tree farther was made smaller. Besides the terrain, models of huge rock were also added to the environment such that the player can stand behind the rock to escape from animals.

Figure 6. Whole view of the island.

Non-moving Objects

In order to save time of development, the team chose to use existing 3D models online for the objects involved in the game. Most of the objects have suitable 3D models online, but for some of the special objects like shelter, they needs modeling. Rather that doing modeling by Blender, the team combine the existing models to build those special objects as a new prefab, because the team had already got the models of their raw materials. Combining two existing models is easier that modeling a new model. Also, as models being combined are used for the game, reusing them spends less space than using a whole new model.
For each of the gameobject used, they were attached by rigidbody component and collider component in Unity. Rigidbody component letting the gameobject to have physical properties like in the real world. Collider component letting the gameobject to become concrete and they can not go through each other under natural contacting.

For shelter and firewood, as they are heavy and not supposed to be moved by player, their mass were set to be very large. Also, as they were attached with nav mesh obstacle component because they obstruct the animals to move. This will be explained in the next part of Animals.

In the part of scripting, the team defined an object to have two behaviours, which is pickable and can be combined when multiple objects are put together. These two behaviours were implemented by having two script for objects.

For the script controls the pickable behaviours, it was only attached to objects pickable, which is light and small. Most of the objects were attached by this script, except some unmovable objects like shelter and fire. This script can be called to pick up object, put down object, hold the object and put back the object into inventory. For each object, information such as name and image had to be inputted. When the object is called to be picked up by the player script, these information and the gameobject itself is added to inventory. The gameobject will then be disabled such that the player cannot see it. When the object is called to be dropped, the position of the gameobject will be set to in front of the camera. Then the gameobject is enabled such that if free fall to the ground. When the object is called to be hold, it is similar to being dropped, but the position is set to the hand of the player. Last when the object is called to back to inventory, the script simply disable the object again.
For the script that combining two objects, it was attached to a unique object of a list of required objects. For instance, to build the ridge pole of shelter, 2 Y-shaped tree branch and a long tree log is needed. The script should be attached to the long tree log and there should be only one long tree log in the environment. To use this script, name and amount of other components, as well as the output object have to be inputted. The script detects whether the gameobject it attached to collides with other gameobjects every game frame. If the gameobject collided is one of the required components, it will be added to count and save the gameobject to a list. Once all the required components are fulfilled, the script will destroy all gameobject in the list and instantiate the output object. Finally destroy the attached gameobject itself.

Some gameobjects may be attached with the third kind of scripts - task gesture handler. Task gesture handler is different from general gesture handler mentioned above. General gesture handler is called by the gesture recognition module once the module recognized. Yet task gesture handler executes every game frame to monitor the motion of player in order to finish the task. To do so, task gesture handler call the public function of general gesture handler for the position of joint involved. As it is task based, the script is only run when the corresponding task start. By doing so, not much burden will be brought to the general gesture handler.

An example of task gesture handler is fire handler used in the task of making fire. Fire handler was attached to fire board. The script starts running when the fireboard is put into the scene. The fire handler monitors the position of the arm of the player and heat the fireboard based on the value. If the fireboard get enough heat, it calls for smoke and fire in sequence.

**Fire and Smoke**

Fire and smoke are a special type of non-moving object because they are volumeless and their shape keeps changing. As a result, they should not be represented by a concrete 3D model. In this case, particle system in Unity was used for the fire and smoke.

In order to control the intensity of the fire and smoke, a script is attached to it and its functions can be called by the player. The player can call the script to make smoke or make fire. For making smoke, only smoke is shown. The script get the parameter of position of its fuels from its caller, because the position of the fire and smoke should be depends on its fuels. Then, the value of start lifetime and start size of the particle system of the smoke is enlarged from time to time. As fire is not exist at this state so the size of smoke was kept small and upper limit of these values is implemented. For making fire, the script increase the size of both fire and smoke. To do so, start lifetime, start speed, start size, maximum particle number and radius of both the smoke and fire is increased. The initial and final values can be set publicly.
Animals

The team decided to have two kinds of animals in the task of hunting. Wolf and rat were chosen to be the animal that hunt the player and being hunted respectively. Wolf is required to have the animation of walking, running and biting, while rat is required to have walking, running and death. Finally, the animated 3D model was found from the Unity asset store.

Since the island is full of obstacles like trees and rocks, nav mesh agent component was attach to the animal gameobject. By using nav mesh agent, the animals are able to find path automatically to the destination while all obstacles are detoured. To have this function, first the island was set to be static because baking only consider the collider and slope of the static gameobjects. Then, bake the scene for navigation mesh. However, for shelter and firewood, as they are instantiated in runtime, they cannot be set as static. Thus, they were attached with nav mesh obstacle component. By doing so, even they does not exist when baking, when the animals are find path, these obstacles will also be considered and be detoured.

Figure. Baked navigation mesh. (Water Plane is hided) Animals can walk through the blue areas.
In the script of wolf and rat, they were first assigned a destination so that they walk through the island. Then, ray is casted from the animals to the player character every game frame, in order to check if the animals can see the player character directly. However, they behave differently when they see the player, so different script was used to handle this situation.

For the wolf, it is in the side of hunting. Hence, once it see the player character, the destination will be changed to the player character. The nav mesh agent component attached will direct the wolf to the player character. At the same time, the animation of the wolf will be changed from walking to running so it goes to the player in a higher speed. Otherwise, the wolf will keep walking to the original destination.

For the rat, it is in the side of being hunting. When it see the player character, the destination will be changed to a position behind the rat, which means escaping from the player. The script also changes the animation of it from walking to running. Once the distance between the player is larger that the preset safe distance, the rat will choose another destination that is farthest from the player from a list of preset destinations. The animation will be switched back to walking. If the rat collided with a spear which is the player’s weapon, the rat will dead and have the animation of death. Otherwise, like the wolf, it will go to the original destination.
7.2 Locomotion Method & Input Control

7.2.1 Movement Posture Study

The team did a study on movement posture to collect data for enhancing the design of platform and as the reference of gesture recognition algorithm.

For the first part that measurement on waist height, waist height when standing, crouching and sitting on floor were measured. We found that when standing and crouching, waist height varied by about 20cm. When sitting on floor, the waist height did not vary much. These set of data shows the waist height may have big different because of different posture which the platform should take this into account.

For the second part of the experiment, the posture of actions were captured as video. Same as the previous part, the postures have big differences between testees, especially the actions of punching, kicking and picking objects from the ground.

7.2.2 Input Method Study

To choose a motion capture system that is suitable for the project, the team did an investigation of several motion capture systems, including Xsens, Perception Neuron and Microsoft Kinect.

Xsens is a powerful sensor kit. It can sense the full movement of the player. It also provides library for getting data from the sensor. However, it costs HKD $100K for the sensor and extra HKD $100K for the library, which is out of the project budget.

Perception Neuron from Noitom is another sensor kit. It is less powerful than Xsens but still widely used. Yet, no library is provided with the sensor kit. And it costs USD $799, which still over the project budget.

Last, we turned to Microsoft Kinect. Kinect is a widespread entertainment equipment for general public. It is also available from the Computer Science Department which can provide the device for the team to develop. However, Kinect is limited by the direction of the player. Kinect can only detect the movement correctly when the player is facing the Kinect sensor. Therefore, this should be considered when designing locomotion method.
7.2.3 Platform

The platform was designed to contain 4 parts, which are body supporting pole, kinect holding pole, platform base and base ring track. They are detachable for each other and easy to set so that the platform can be easily store when it is not used.

![Structure of platform](image)

The platform base is a smooth surface with diameter of 1.5 m. Players can stand on the platform base to play the game. When the player is moving on the platform base, as it is smooth, player is kept in the same position because of the unmoved body supporting pole.

The body supporting pole was connected to the waist of player at back of player. The pole is reliable such that it can hold the player even the player fall down.

The platform base and the body supporting pole was connected by the base ring track. With the base ring track, the body supporting pole can rotate around the platform base. Such design lets the body supporting pole keeps at the back of the player even the player turns around.

Last, as Kinect was chosen as the motion capture system, the limitation that Kinect only able to sense movement accurately when the player face the sensor had to be considered. Therefore, the team decided to attach the Kinect sensor to the platform such that the sensor can be kept in front of the player. However, according to the information from Microsoft, the minimal distance between the player and sensor is 80 cm. Therefore, the Kinect holding pole has an external part to increase the distance between them.
7.2.4 Gesture Recognition

Based on the data returned from the Kinect sensor, gesture recognition algorithm was applied to classify the gesture performed by the player. When designing the algorithms, data collected by the movement posture study was used as reference.

Pick Up Gesture

The algorithm only consider the y-coordinate of the player’s hand. Players are not required to do the action by putting hand from the ground. The gesture can be recognized once the player put their hand in a lower position close to the ground. However, if the player is crouching, the algorithm also classify the action as picking up objects, but the fact is not. Therefore, mouse click is also required for picking up object.

Walking Gesture

The algorithm monitors the position of the knee joints. To perform the walking gesture, the players need to pull up left leg, put down left leg, and repeat by right leg. Different sequence of left leg and right leg can also trigger the walking gesture. Yet, sometimes the player may step forwards one leg for kicking, crouching or actions other than walking. Hence, the algorithm only classify the player as walking when at least two steps are continuously performed by the player.

Menu Manipulation Gesture

This is a special gesture because this is not for interacting with the game world objects but for controlling the inventory menu. When the player swipe left or right, the inventory menu show the object on left or right correspondingly. When the player swipe up, the inventory menu tells to hold the object, the inventory menu tell to discard the object if the player swipe down. The gesture is detected by monitoring the position of two hand. One hand keeps putting up and the other hand swipe.
8. Conclusion

Under the rapid development of virtual reality, general public are easier to have a taste of VR game. However, as the equipments for VR game are either expensive, or not providing enough immersive game experience to player, the team decided to develop a VR game with higher immersiveness and affordable cost.

Besides, the team found that in recent year, public especially youngster is losing skills to live in field. Therefore, it is important for them to learn or practise the survival skills.

As a result, the team decided to develop a open world VR RPG about survival. By the game, players can learn survival skills in a safe environment and they can explore the virtual world unlimited by the actual gaming space in real world.

The project is mainly divided into two part, namely game development, and locomotion method and input control. For the game develop, the team only developed the tutorial tasks that let players learn basic survival skills due to the one-year-time limitation. For the locomotion method and input control, the team developed a gesture recognition algorithm detecting the gestures performed by player and return it to the game. To keep the player safe and from the danger of hitting obstacles in the real world, a easy-install platform was constructed.

In the part of game development, it was further divided into game mechanics, player and environment 3 parts. For game mechanics, survival parameters, game flow, inventory were implemented. For the player, 3D model of player character was made. It was also added with gesture handler to handle gesture returned from the input, and added with user interface to show the player status, instruction and inventory. For the environment, background environment was built by Terrain. 3D models of non-moving objects were used in the game with scripts controlling the behaviour of the objects. Last, animated animals were put into the game. They achieved the AI functions by scripting and navigation mesh.

In the part of locomotion method and input control, the platform and gesture recognition algorithm were developed separately. For the platform, the team construct a platform with 4 detachable parts. The platform can hold the player when the player fall down in order to keep their safety. Also, the platform can keep the Kinect sensor in front of the player to let the sensor maintain accurate detection.
9. Future Development

9.1 Game Development

As currently, the team only developed the tutorial tasks for the game. For future development, a complete game can be developed. As mentioned before, the game should have a storyline and implement with a quest system.

Besides the game mechanics, the environment of the game also needs modification. The present island for the tutorial tasks is just a smaller scale of map. To have a large scale map, algorithm should be used to show the part of map that the player can see only. Otherwise, if keep the whole large scale map enabled at the same time, performance of the game may be affected. Therefore, the map should be break down to smaller part. Only the parts near the player should be enabled.

For the objects involved in the game, more variation of objects can be used. For example, at later in the game, the player may need to fight with enemies. Therefore more types of weapon may be needed. Like gun, it is not is non-moving object which only need to be pick up. Scripts may be need for emitting bullets.

Last, as the game progress, the player may interact with other characters. Human behaviour is not as simple as animal. Hence, more complicated AI scripting may be need for future development.

9.2 Locomotion Method and Input Control

Currently the Kinect sensor is attached to the platform. Although this design is to keep the Kinect sensor in front of the player, when the sensor is rotating about the platform, there is probability that the sensor hit something or even someone. To solve this problem completely, alternative solution may be considered. For example, using multiple Kinect sensor to sensor the movement of player in different direction. However, further study is needed.

From the angle of gesture recognition, the current implementation can be modified to provide higher immersiveness to player because mouse input was used for gesture recognition. To improve this, position of more joints can be compared. Yet this may lead to worse performance of game and more study on movement postures are needed.
References

