Virtual Worlds Assignment 2

**Class** : Virtual Worlds

**Submission Date** : 2024/05/10

**Student ID** : 2022742021

**Affiliation** : Kwangwoon University

**Full name** : Juho Kim

**Major** : Software Engineering

# Abstract

This report provides a comprehensive walkthrough of the development process for the Virtual Worlds Programming Assignment 2 implemented in Unity 3D. First, the fulfillment of each requirement is discussed with descriptive code snippets. Then the report concludes by addressing the challenges encountered during development and outlining the limitations of the current code, which presents opportunities for future improvement.

The game was developed mainly applying the knowledge of vectors, transformations and dynamic object spawning. For more information, please visit [my GitHub repository.](https://github.com/juho-creator/VirtualWorlds2/blob/main/README.md)

# Requirements Fulfillment

Following table shows the assignment requirements from the slides along with completion status.

|  |  |  |
| --- | --- | --- |
| Easy 1 | Vertical Player Movement | O |
| Easy 2 | Animals increasing proportionally | O |
| Easy 3 | The food no longer moves | O |
| Easy 4 | Camera switcher | O |
| Medium | Animal Hunger Bar | O |
| Hard | Move to food | O |
| Expert 1 | Move freely | O |
| Expert 2 | Collision avoidance | △ |

**Easy 1 (Vertical Player Movement)**

Step1 : Initializing variables

**A black background with white text

Description automatically generated**

To begin with, the following 2 variables at **PlayerController.cs** were made private to prevent it from being contaminated by the user :

* **verticalInput** : receive input of vertical position from user
* **zRange** : Set bound for vertical movement

Step2 : Translating player vertically with user input

**A black background with white text

Description automatically generated**

The value of total time user spent pressing the arrow key multiplied by the predefined speed was passed in the Translate function to translate the player vertically when user presses the side arrow keys.

Step3 : Setting vertical motion range of player

**A screenshot of a computer program

Description automatically generated**

Then the vertical motion range was set to prevent user from moving out of the bound.

**Easy 2 (Animals increasing proportionally to current population)**

Following functionalities were implemented in **SpawnManger.cs** for increasing animals proportional to the current existing animal population on the ground:

Step 1: Initializing variables

A screen shot of a computer program

Description automatically generated

* **creaturePrefabs** : References the 3 types of animals to be spawned
* **spawnRangeX** : X range for spawning animals
* **spawnRangeZ** : Z range for spawning animals
* **startDelay** : Initial delay before starting to spawn animals
* **spawnInterval** : Delay for spawning new animals
* **maxCreatures** : Maximum number of creatures that can exist on the ground
* **creatureList** : List for tracking all animals on the ground

Step 2: Limiting spawn rate to prevent overpopulation



To prevent overpopulation, the number of creatures spawned is decreased with more animals existing on the ground. As the value returned from **CurrentCreatureCount** increases, the value saved in **creaturesToSpawn** is decreased.

Step 3: Spawning animals randomly

A screen shot of a computer code

Description automatically generated

**spawnPos** and **spawnRotation** are each used to randomly set the spawn position and rotation of the animal. Then random **creatureIndex** is used to spawn random animals and added to **creatureList** to track all the animals currently on the ground.

Step 4 : Untracking destroyed animal and updating spawn interval

A screen shot of a computer code

Description automatically generated

To untrack destroyed animals, NULL animal objects are removed from **creatureList**. Also, **AdjustSpawnInterval** is called to update spawn interval in proportion to current animal population.

A computer screen shot of white text

Description automatically generated

As the population of animal increases, the **newSpawninterval** is decreased, hence effectively spawning animals more frequently with increased animal population.

**Easy 3 (The food no longer moves)**

A black and white text

Description automatically generated with medium confidence

To prevent the food from moving, Move Forward script has been disabled, but not removed from the bone object. The reason for doing so is because we might want to enable it in the future.

|  |  |
| --- | --- |
| A screen shot of a computer  Description automatically generated |  |

Also, inside **PlayerController.cs** the offset vector has been added to the current position of the player to spawn the bone slightly ahead of the player’s current position for better user experience.

**Easy 4 (Camera switcher)**

Step 1 : Initializing variable for setting perspective

A black background with text and colorful letters

Description automatically generated

A private variable toggleView is initialized to 3 for setting default view to third person perspective.

Step 2 : Changing toggleView with user input

A black screen with white and green text

Description automatically generated

toggleView alternatives between 1 and 3 whenever user presses the left or right shift key.

For example, when user presses shift on third perspective, the new toggleView is 4-3 = 1.

Step 3: Setting user perspective

A computer screen shot of a program

Description automatically generated

The user perspective is set based on the value saved in toggleView, with the projection of each third and first perspective set to orthographic and perspective respectively.

Additionally, the rotation and position of the camera has been adjusted for each perspective.

**Medium (Animal Hunger Bar)**

Step 1 : Creating the health bar

A screenshot of a computer

Description automatically generated

Health bar object was created with **Foreground** and **Background** each set as green and red color, and a slider was added to control the width of green color to give the illusion of functioning like a health bar. Finally, **HealthBar.cs** script was added to make the healthbar interactive.

Step 2: Linking healthbar with each animal

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | To follow each animals, **Healthbar** has been added under each animal objects. |

After setting up the heathbar object with all animals, the following were implemented in **Healthbar.cs** :

Step3 : Initializing variables

A black background with white text

Description automatically generated

* **Slider** : For controlling hunger bar
* **mainCamera** : Track perspective of the camera to change hunger bar rotation in the future
* **hungriness** : Rate at which hunger bar is decreasing

Step 4 : Initializing hunger bar and camera object

A screen shot of a computer error

Description automatically generated

Hunger bar is set to max, 100 and maincamera object is initialized.

Step 5 : Destroying animal when hunger bar is zero

A screen shot of a computer program

Description automatically generated

When hunger bar is zero, the parent object, its corresponding animal is accessed and destroyed.

Step 6: Decreasing hunger bar over time



Hunger bar value decreases with change in time and hungriness.

Step 7 : Displaying hunger bar view from third perspective

A black screen with white text

Description automatically generated

The orientation of the hunger bar changes depending on whether the game is viewed from a first-person or third-person perspective, facing either the Z-axis or the Y-axis.

Top of Form

Step 8 : Gaining health upon eating the bone

A black screen with white text

Description automatically generated

**GainHealth** was defined in **HealthBar.cs** to increase health upon eating the bone.

A computer screen shot of a code

Description automatically generated

Finally, **HealthPot.cs** was attached to the bone to manage the child object of the collided object (the hunger bar), utilizing the **GainHealth** method to augment the hunger bar's level. Additionally, the bone is removed from the scene when consumed by an animal.

**Hard (Move to food)**

Following code was implemented in **MoveForward.cs :**

Step 1 : Looking for closest bone



The animal looks for any bones nearby.

Step 2 : Calculating distance from the bone

A screen shot of a computer

Description automatically generated

The distance of the animal from the bone is calculated.

Step 3 : Animal moving towards the bone

A black screen with colorful text

Description automatically generated

If the animal detects the bone within the **detectionRange**, it moves toward the bone.

**Expert 1 (Move freely)**

Following code was implemented in **MoveForward.cs :**

A screen shot of a computer code

Description automatically generated

**Wander** rotates the animal for every **nextWanderTime** to make the animal freely.

**Expert 2 (Collision avoidance)**

Following code was implemented in **MoveForward.cs** :

A computer code with colorful text

Description automatically generated

Upon colliding with another animal, the animal pauses for 0.5 seconds and rotates to face other direction, effectively avoiding collisions.

# Conclusion

◼ **Key challenges tackled**

Some of the hardest parts were:

1. Creating health bar object



Initially, following the [tutorial from Kapp](https://www.youtube.com/watch?v=34cxdprbpaU) the hunger bar was made using one slider, however this caused the health bar to appear to be partially filled when hunger value is zero.

A red rectangular object with a blue background

Description automatically generated

To prevent partial filling of the hunger bar, two image objects were added under a slider.

1. Making health increase

I was not sure how to access the hungerbar from another script because it was a child object of an animal object. So it took some time to read the [Unity documentation](https://docs.unity3d.com/ScriptReference/Component.GetComponentInChildren.html) and understand **GetComponentInChildren.**

1. Displaying health bar at third person perspective

|  |  |
| --- | --- |
| A video game of a video game  Description automatically generated | Initially the hunger bar did not appear on third person perspective as shown in the picture.  So I attempted to create a 3d hunger bar, but the UI for the hunger bar fell short of expectations in terms of quality and effectiveness.  Therefore, in order to display hunger bar at third person perspective, code has been added to **HealthBar.cs** to rotate hunger bar towards Z axis on third person perspective |

◼ **Limitations**

Collision Avoidance may not occur when animal is spawned on another existing animal.

◼ **Future improvements**

Prevent animal from spawning at location where there is another existing animal.

References

* [Getting variable from another script in unity](https://www.youtube.com/watch?v=2pCkInvkwZ0)
* [Difference between Update(), LateUpdate(), FixedUpdate()](https://discussions.unity.com/t/what-is-the-difference-between-update-lateupdate-fixedupdate-and-when-i-should-use-them/26258)
* [HungerBar in Unity2020](https://www.youtube.com/watch?v=34cxdprbpaU)
* [How to make a HEALTHY BAR in unity](https://www.youtube.com/watch?v=BLfNP4Sc_iA)
* [Gain Health on Collision](https://www.youtube.com/watch?v=HOJHDhUWKk8)
* [Tags – Unity Official Tutorials](https://www.youtube.com/watch?v=EA54-vfLkUI)
* [Experimenting with Obstacle Avoidance in Unity3D](https://www.youtube.com/watch?v=SVazwHyfB7g)