I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any

other award.

Robert Klivis

Report

Working with Game Engines

Name: Robert Klivis

Module Number: M21622940

Course: Computer Games (Software Development)

GitHub Link:

Contents

[Scene 1 3](#_Toc7134484)

[Audio 3](#_Toc7134485)

[Inventory 3](#_Toc7134486)

[Sorting 4](#_Toc7134487)

[XML 4](#_Toc7134488)

[Scene 2: Part 1 5](#_Toc7134489)

[Player 5](#_Toc7134490)

[Camera Shake 6](#_Toc7134491)

[Camera Follow 6](#_Toc7134492)

[Player Controller 6](#_Toc7134493)

[Player Movement 7](#_Toc7134494)

[Scene 2: Part 2 7](#_Toc7134495)

[Editor Window 7](#_Toc7134496)

[XML 7](#_Toc7134497)

[Dialogue 8](#_Toc7134498)

[Sources Used 9](#_Toc7134499)

# Scene 1

## Audio

For creating the audio different events are being used. There are separate events for whenever block type is changed. For example, “OnEventBlockDestroyed” is being used for when block is destroyed while other events are used either of four types of blocks.

“SetBlock” function is used to know which of the events use and when.

These functions are done in the Voxel Chunk script.

Also, in audio manager script public “AudioClip” variables are added by using previously described event functions.

To play the current audio files “GetComponent” function is used so that the necessary audio files can be added.

Then they are added through “OnEnable” function to set events that are called in the Voxel Chunk script.

They are also removed by using “OnDisable” function by removing these set audios from the set events.

## Inventory

Inventory is displayed on the screen by using Canvas option in Unity.

It consists of a main panel that includes different inventory items and their names and amounts.

When set item is placed or picked up then it adds to the inventory by adding or subtracting from the amount displayed.

It is done by having a function that adds to float of the amount in the Player Script.

Also, there is a function in this set script that makes sure that the player has more than zero of the amounts for the set block that player wants to place. If the player does not have any blocks, then they cannot be placed until more are picked up.

More blocks can be picked up whenever a block is destroyed and smaller clone of the removed block spawns on the ground for the player to pick up.

Item names and amount are display by using Text function that includes the set strings.

Necessary floats are transformed to string to be displayed as a text.

## Sorting

For the sorting, merge sort is used to sort between different amounts of blocks and types of blocks.

Inventory is sorted through Inventory Management script.

It gets different types of blocks by using “GetComponent” function and accessing the Player Script to tell the amount and the name of each block.

Each block type is then added to the list for sorting.

For Loop is used to sort set names and amounts.

“DisplayListInOrder” function is then used to display the list in correct order on the screen for the player to see.

## XML

Voxel Chunk is loaded from the provided XML file for this assessment.

To do this TestDataClass, XMLVoxelFileWriter, XMLSerializeScript scripts are being used.

TestDataClass scrip consists of public string functions for name and description.

It also consists of “TestDataClass” function to set necessary name and all the necessary description for set xml files.

XMLVoxelFileWriter script consists of “SaveChunkToXMLFile” function to save the set chunk to xml file.

It is done by using XmlWriterSettings option to set necessary settings and XmlWriter to create the xml file with the set name and description.

For Loop is used to get the necessary variables and other information and then it is transformed to a string that can be used in the set xml file.

This script also consists of “LoadChunkFromXMLFile” function to load the information from the set xml file.

It gets the “voxelArray” variable and it also uses XmlReader to read the set xml file and its description.

This function uses while loop to read the values of x, y and z and sets it to “voxelArray” values.

It also has a function called “fileExists” that makes sure if an xml file already exists and if it does that it debugs a “File Exists!” text.

XMLSerializeScript is also being used for the set xml files.

In consists of two separate function for when the “F1” or “F2” buttons are pressed in the update method.

If “F1” button is pressed, then it uses the TestDataClass Script to set the name and description of set xml file.

It also uses XmlSerializer to get this set information.

And it also uses FileStream function to create a new file or save to existing file.

In this case the set file is “AssessmentChunk2.xml” which has been provided for this assessment.

However, if “F2” button is pressed, then it reads the information that is already there.

It is also done by using TestDataClass but in this case to read the name and description that is already there.

It also uses XmlSerializer to get the set information.

And it uses FileStream function to read from the set xml file which in this case is provided.

It also Deserializes the file from TestDataClass script.

# Scene 2: Part 1

## Player

The Player consists of a first-person controller object to have a set audio when player is moving around and also to have a camera for the first person.

This has been sourced from Unity by importing set assets.

Player also consists of PlayerScript that sets amounts for different blocks that are used in the inventory.

It also consists of a prefab for the clone of set block that has been destroyed.

As well as an inventory object for setting up the inventory.

It also uses RaycastHit function to make sure that the block can not be placed when the player is too far away from the surface as well as the same functions are used for destroying blocks so that it works properly and as intended.

## Camera Shake

*Can be found in Scripts2 Folder*

To make the camera shake possible, variables like set camera transforms, and different floats of “shakeDuration”, “shakeAmount” and “decreaseFactor” as well as vector 3 for original position were used, as well as bool for the shake itself.

Original position was set to position of the camera at the start of the scene.

Also, on the start function “shakeDuration” was set to zero and shake Boolean was set to false to make sure it is not activated at the start of the scene.

In the update function, set if statements were made, for example if statement when shake duration was more than 0 then then camera position was moved by using random number inside the unit sphere that was added to the original position and multiplied by amount of the shake.

If the set shake was not active, then shake duration was set back to zero and camera position was set to original position.

Also, if shake Boolean was true then shake duration was set to a set number and if not then it was reset to zero.

## Camera Follow

*Can be found in Scripts2 Folder*

For the camera follow, game object and vector 3 variables were set. These were used for the player and the offset position.

In the start function offset is set to its transform position minus player transform position.

Also, in the update transform position is set to player transform position plus the offset position.

## Player Controller

For the player controller different events and variables are being used like jump and movement events as well as set timers, camera and necessary dialogue variables. Set button are used for whenever player is moving or jumping. Also, camera shake as well as follow is activated through here to make sure that it works as intended. As well as activation for the dialogue whenever player comes close enough to the NPC to activate it and start the dialogue. This is set here so that player could only talk to the NPC when it is in close range to the NPC.

## Player Movement

Player movement consists of different necessary variables for speed, gravity, impulse and other things that would affect the general movement of the player. There is a state method used for when the player is on the ground or in the air as different functions are used for each one. There are also states for dashing so that the player would be able to do so when necessary. For the dash function, timer is also set so that it would work as intended and not dash for too long of a time. Another state included is a “disabled” state that makes sure that the previous state is disabled before next one is activated.

There also is a script used for resetting the player position whenever the player falls of the platform and has to be respawned. It uses a basic trigger detection that activates when player collides with the set trigger collider for the ground.

# Scene 2: Part 2

## Editor Window

*Can be found in Editor folder*

To create an editor window that can be for the set Unity project, “UnityEditor” option was used. Static Editor Window event was also set and new menu item to display this editor window was also set.

“OnGUI” function was used to create all the necessary buttons and texts for this editor window. It consists of set label fields, buttons and text areas to display necessary dialogue texts that can be changed and customized by the user of the Unity project.

When set dialogue are selected then they are saved to the inputs for the player and NPC, which is later saved and accessed through XML functions.

This window can be used to make it easier for the developer to change and save different dialogue options that are later used in the game itself.

## XML

To save and load certain dialogues both for the player and NPC, they are saved to the dialogue xml file. This done by having a dialogue data class script that includes string for the name of the xml file and different inputs that are the players and NPC inputs.

To save all the necessary dialogue information, “DialogueFile.xml” file is created. Same file can later be accessed to display necessary dialogue and so that it could also be altered in any way desired by the developer.

## Dialogue

To create the dialogue few variables were set like game objects for camera, dialogue and buttons. Also, there were set variables used for the texts for all the buttons and dialogue texts as well as Booleans and float for timer.

To make sure that dialogue worked as intended, there were few functions created for each button clicks which consisted of setting up the text and activate or disabling buttons themselves.

Also, to make sure that dialogue option only starts when player reaches the NPC, “OnCollisonEnter2D” function is used. This function checks if player collides with the NPC and when it does then dialogue, and buttons are set active as well as necessary text is displayed. However, when player leaves the area close to the NPC then these set dialogue buttons are disabled again.

# Sources Used

* Labs 1 to 8, provided by the Lecturer
* <https://docs.unity3d.com/ScriptReference/Component.GetComponent.html>
* <https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html>
* <https://docs.unity3d.com/ScriptReference/Random.Range.html>
* <https://docs.unity3d.com/ScriptReference/UI.Button-onClick.html>