Creating 3D games with Unity3D

1st Boris Fuchs
Mobile Computing
University of Applied Sciences Upper Austria
Hagenberg, Austria
boris.fuchs@students.fh-hagenberg.at

2nd David Mitterlehner
Mobile Computing
University of Applied Sciences Upper Austria
Hagenberg, Austria
david.mitterlehner@students.fh-hagenberg.at

3rd Anika Seibezeder
Mobile Computing
University of Applied Sciences Upper Austria
Hagenberg, Austria
anika.seibezeder@students.fh-hagenberg.at

***Abstract*—Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Nowadays creating realistic 3D games is getting a lot easier, due to various tools like Unity3D or the Unreal Engine. Game developers need little to no programming skills to make games in a minimum amount of time, because a great amount of premade assets and scripts are available for a very low cost. Most of the developing work can be done in the graphical editor, like creating 3D figures and adding properties, which can include self made scripts or physical properties. The majority of the existing game editors provide developing on multiple platforms including Smartphones and the most common game consoles. However Unity3D is better suited for devices with low end graphics cards, because it is more lightweight than the Unreal Engine. Compared to it, the available assets of Unity3D are not as high quality, but more economical on memory and graphics.

The main points of this paper are: outlining the workflow of the game development process in Unity3D which consists of creating an environment, making a Third-Person-Character and some enemies, and comparing the basic functionality and simplicity of Unity3D and the Unreal Engine. The advantages and disadvantages of both editors are examined and demonstrated.
The first point of the game development process in Unity3D is to create an environment, with a small forest, some mountains and a few cottages. Then some textures are added to the components. Secondly the Third-Person-Character is linked with a script, so it can move around in the environment. The third point is to create some enemies, that are chasing the player. Lastly animations for both, the character and the enemies, are provided and added to the game.

The goal of this paper is to demonstrate the simplicity of Unity3D by creating a small 3D game in Unity3D.To keep the costs low, only freely available assets are used. The focus is not on scripting, but on the features of the graphical editor provided by Unity3D and the Unreal Engine.

II. Creating an Environment

A. Subsection 1

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing.

III. Character Controls

Character controls are a must have when developing a game. Otherwise the user would not be able to interact with a given character. No matter how hard the user is pressing a key or moving the mouse, the character still would not move at all. With Unity3D the user interaction can be easily made by implementing scripts and adding them to the editor via Drag and Drop. The scripts can be written in C#, JavaScript and Boo [1]. In the editor itself there is no opportunity to write the code, but there are several supported editors such as the recommended editor MonoDevelop by Unity3D or VisualStudio by Microsoft [1].

A. Adding a Character

In Unity3D adding a character is easily done by dragging a Prefab or an object into the middle screen to the wanted position. There is also another possibility by dragging the Prefab or object to the left side of the screen, where all used objects are shown. A Prefab is a pre-built object combined of one to many components [2]. When downloading the Unity3D editor some prefabs are already given and can be used as is. To see and edit features of the Prefab or object, a left click on the wanted object on the left side of the screen is needed.

As seen in figure 1, a character has components like a Rigidbody, which gives the character a mass, a capsule collider, which triggers interrupts when colliding with other objects and scripts. Scripts can have some changeable variables like shown with the 2nd script, or no changeable variable.

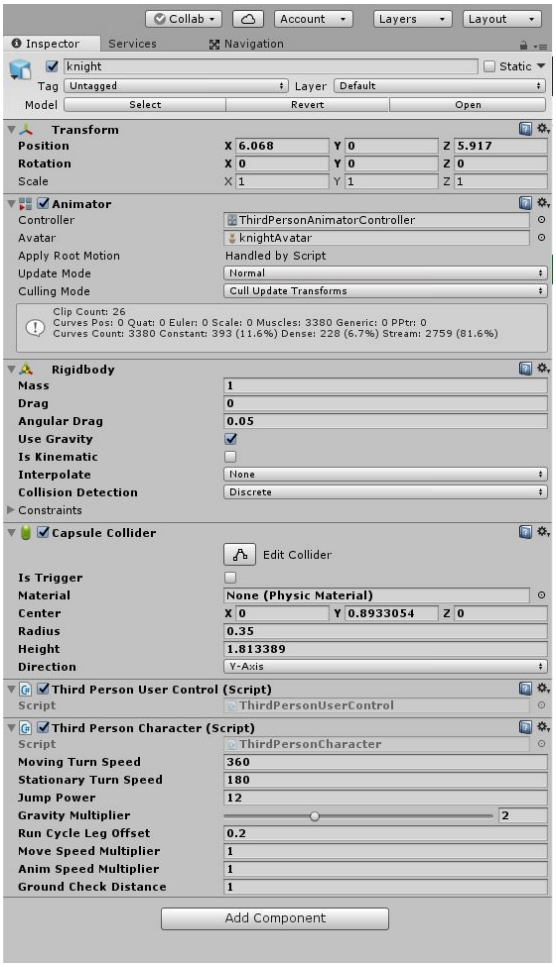

Fig. 1. Character Components

B. Scripting in VisualStudio

A script is simply added by clicking the button ”Add Component”, choosing ”New Script” and selecting the wanted language. After adding the script it can be written by double clicking the component and opening the wanted script editor. The first thing to know about writing a script for Unity3D is, that the class is extending from the `class MonoBehaviour` and is ipmlementing the functions `void Start()`, `void Update()` and `void FixedUpdate()`. The start function is used for variable initialization and one time execution. The update function is used for continuous events, like checking which key is pressed, and is updated every frame [1]. As the user may press other keys while playing the game, the statements for checking the key should be implemented in the update function. Both functions are not necessarily needed but can help building the logic of the game. Another important function is `void FixedUpdate()`. Like the `void Update()` function, it also is updated every frame and it is mainly used when dealing with the Physics of the Object [1].
When adding variables there is just a few thing to know: all `public` variables are later shown in the component screen and can be edited there. It is also possible to import other components from the object. They are added to by using `GetComponent<>()`

C. Cross Platform Input Manager

The Cross Platform Input Manager can be used to determine how keys or actions should be interpreted. For example there can be an entry considering jumping with a character, which defines, that a jump on a Desktop or Laptop is made with pressing the spacebar, while on the smartphone a touch gesture activates the jump. After defining the inputs, they can be used with `CrossPlatformInputManager.GetButtonDown("Jump")`.

Listing 1. Example Implementation for Determining a Jump

```
1 private bool isJumping;
2
3 private void Update()
4 {
5    if (!isJumping)
6    {
7       isJumping = CrossPlatformInputManager.
            GetButtonDown("Jump");
8    }
9 }
```

The code in Listing 1 shows how to implement the Cross Platform Input Manager in the `void Update()` function which is updated every frame.

IV. Adding Animations

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

V. Comparing Unity3D with the Unreal Engine

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

References

[1] Ji Won Oak and Jae-Hwan Bae, "Development of Smart Multiplatform Game App using UNITY3D Engine for CPR Education", vol. 9, no. 7, pp. 263–268, 2014.

[2] Unity Technologies. "Manuel:Prefabs", Publication: 2018.1-002B, 2018.