A 3D Character Customization Tool using Unity Game Engine

Parth Patel | 013705718
Dennis Dang | 012444874

Table of Contents

Abstract

Character Customization is a well-known functionality in the gaming industry where users can change certain aspects of the character to suit their creative needs in various games. The proposed project consists of creating a basic working prototype of a character customization creator. Some planned features for this project include the capability to change the color of the hair, eyes and features of the body such as height and body size. The following project will also include an animation option to apply different animations onto the character while inside the customization mode.

1. Introduction

Most video games are oriented around the player. Video game makers will often highlight the features of their game as being centered around the player or allowing the player to craft his or her story and adventure. This is apparent in many video game press conferences and showcases.

RPGs (role playing games) champion this idea. Examples include Fallout 4 and Assassin's Creed: Unity shown below in Figure 1 and Figure 2, respectively. In RPGs, players take control of a character and adapt a certain role to accomplish goals to get to the end of the story within the game. Since many RPG titles have multiple ways of reaching the ending, players can adapt different roles for every new playthrough. Thus, to give players a sense of a different adventure each time they begin a new journey, almost all RPG titles allow players to create and customize a character. Character creation always happens in the beginning of the game before any story is introduced.

Fig. 1: Character creator for Fallout 4.

Fig. 2: Character creator for Assassin's Creed: Unity.

Character creators from multiple RPGs have overlapping customization elements. Most will allow players to modify features such as hair style, hair color, facial expression, clothing and equipment. Other character creators may allow for more exotic customizations such as positioning of each eyebrow, size of lips, how much the character's ears and nose protrude, length of arms and feet size.

For this project, we aim at creating our own character customization system. Having a good looking and easy-to-use UI (user interface) was one of the main goals for this project. This is because having an unintuitive and bad looking UI can draw the user away from the experience. We wanted the user to have an easy time creating a character as well as being immersed into the experience.

One feature that we added in our character creator was the option to preview the character using different animations. While the user is changing certain features of their character, the user can choose to have the character perform a certain action. This allows the user to preview ahead and see if the way the character looks while performing that action is

satisfactory to the user. This is a feature found in few video games where the user is restricted to just customizing their character without having the opportunity to preview their character with animations.

To make our character creator, we used the Unity game development platform. Unity provides a framework complete with the tools needed to create entire video game, let alone a character creator tool. We used Git as our version control software. Unity does provide their own version control but their software imposes a file size restriction. Since our project files were relatively large, we were compelled to use other tools that did not restrict overall project file size.

2. Functional Design

The design process for this project started off with a sketch where the parts of the design idea were taken from great games such as Code Vein and Destiny 2. Due to the fast growing design ideas in the industry, the UI design of a 3D game has to meet those standards to improve the user experience. Therefore, taking all these requirements into account, we designed a particular UI as shown in Figure 3. The ideas that went into this were from visualizing the application as a game running on a console. Video game consoles usually require games to have their button icons hence, this would be added to the black overlay transparent bar at the bottom of the design.

The rest of the design process following the conventional design concepts and the previous scenes. As shown in Figure 3, Figure 4 and Figure 5, these are the main three scenes used during the project. The final output of the application turned out to be very similar because of the design that we followed throughout. There were some elements which we removed due to unnecessary requirement and lack of space. For example, the “Load Character” button shown in Figure 1 required us to implement the saving feature for the application however, this was not proposed in the project goal therefore, we decided to remove the button overall.

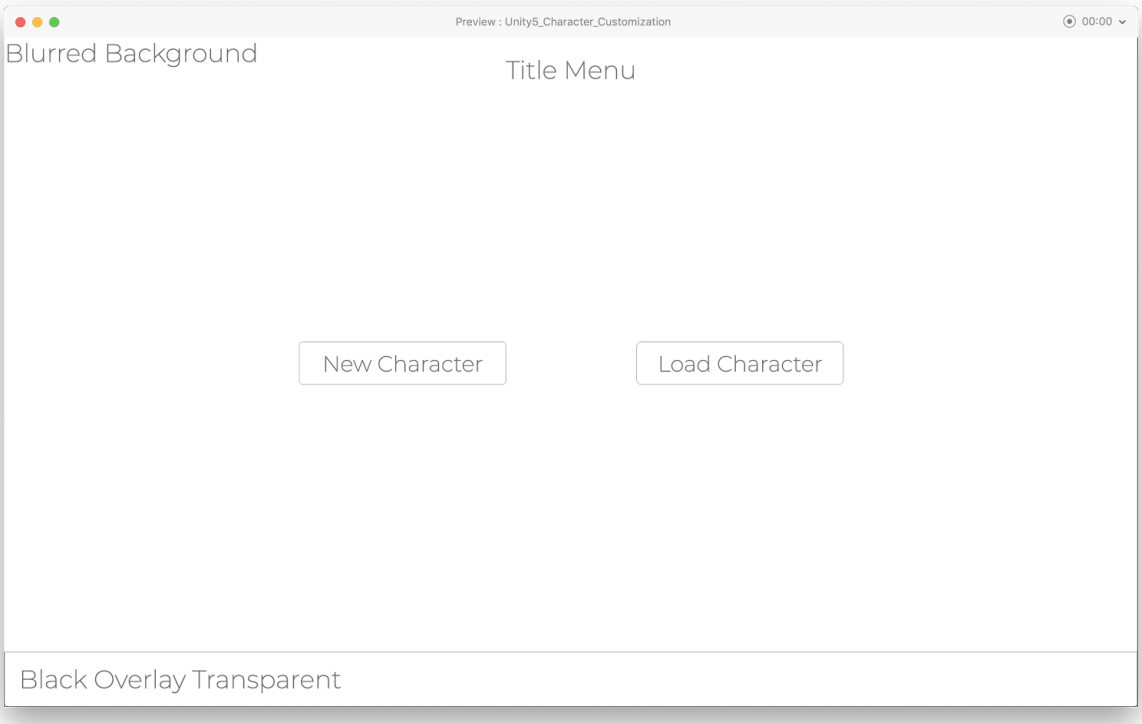

Fig. 3: First scene design of the application in Adobe XD.

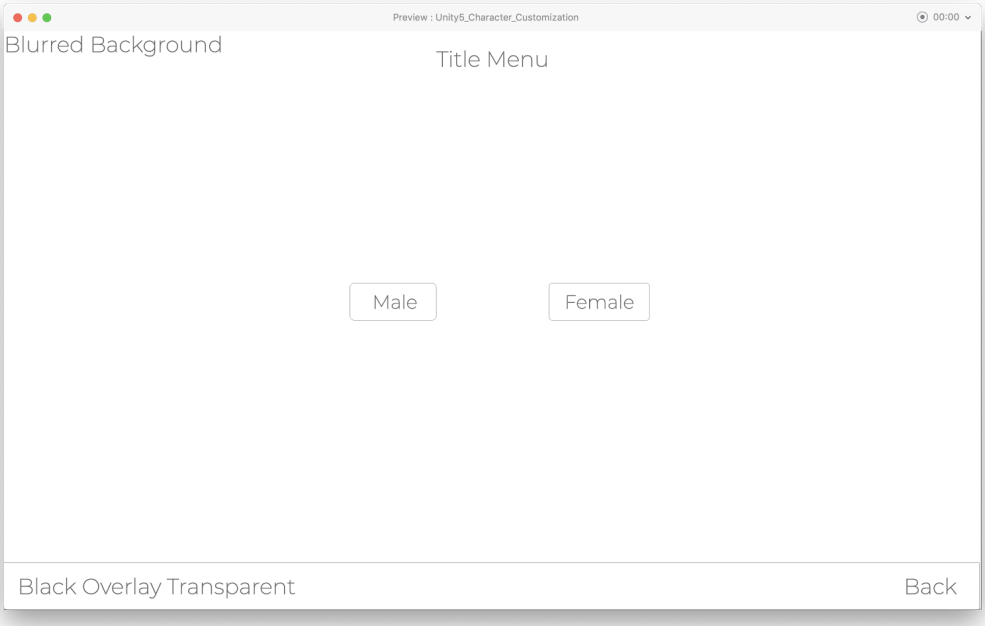

Fig. 4: Second scene design of the application in Adobe XD.

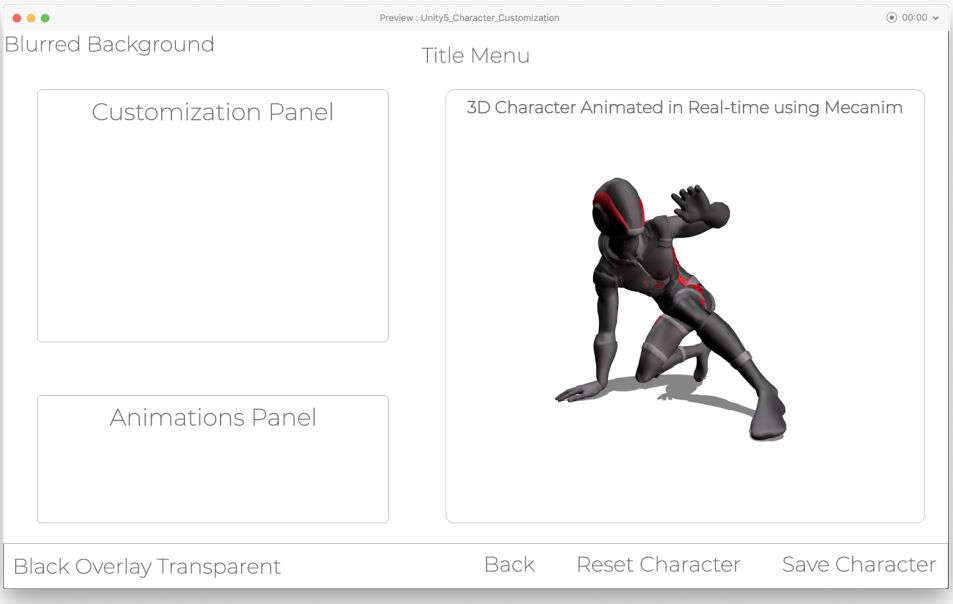

Fig. 5: Main scene design of the application in Adobe XD.

The inspiration gathered before attempting this project was from games known as Destiny 2 and Code Vein (as shown in Figure 6 and Figure 7 respectively). These games have a common design pattern because of their 3D games and quality of the design. The design quality is a trait that would enable any user to play the game even more and become interested. To implement the project from carefully visualizing the design, it was essential to have a character with a great quality to mesmerise the players and help them customize the character further by the accessories that we provided in the application. Although, Code Vein provides a great amount of accessories and options, Destiny 2 keeps it simple by only having a few attributes to customize. However, Destiny 2 shows a better functionable UI in terms of graphics and user preferences such as having buttons at a reasonable amount and reaching to a particular option without needing to click several buttons.

Fig. 6: Code Vein Character Customization design.

The design pattern gathered from both of these games allowed us to create a design that is suitable for any type of player whether it is a common player who likes more functionality or a player who requires a great UI. This design can be further seen in Figure 9 onwards.

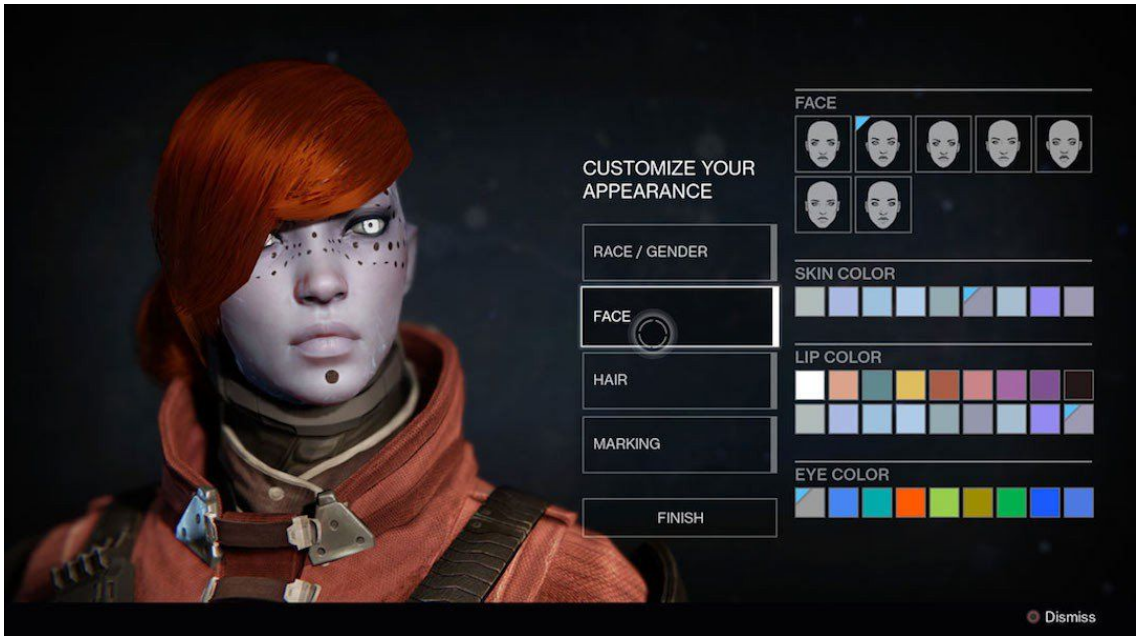

Fig.7: Destiny 2 Character Customization design.

3. System Design

The system design that was used in this project turned out to be smooth because of the decreased amount of button clicks required to move between scenes. All of the customization attributes are closer to each other, making it easier to navigate between the panels of the buttons. The below Figure 8 shows the workflow of the whole system architecture. It portrays each functionable element and the relationship to the functions that it provides. The first two scenes are very lightweight due to the scenes not requiring any other functionality other than moving between scenes. As the scene level goes up, the functionalities increase and the character customization attributes are visible to the player. The reason we chose these main attributes (hair, face, body and clothing) is because these are required from what we researched from previous games. These attributes allow the player to customize to their needs.

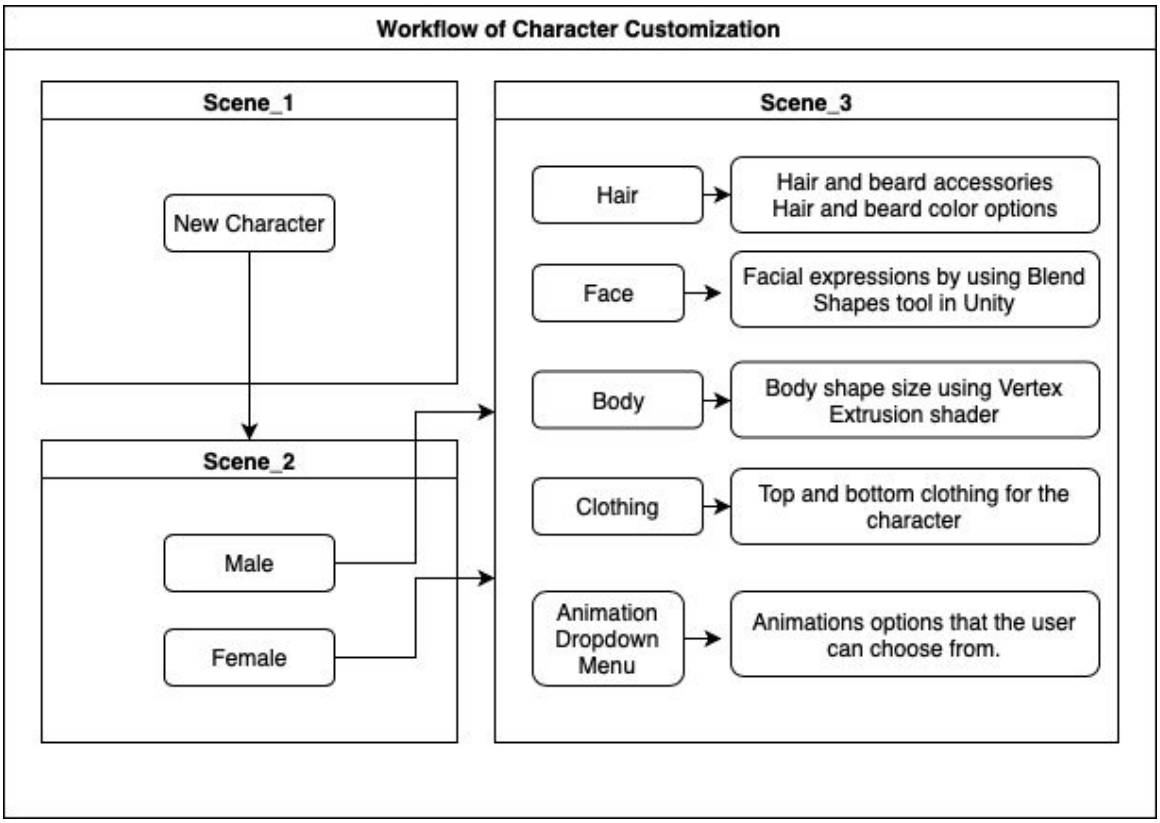

Fig. 8: Workflow of the system architecture.

4. Implementation

Since the goal of this project was to build a character creator with a nice looking UI, a considerable amount of time was spent trying to find free models and assets to work with. Designing our own models and clothing would have been beyond the scope for this project. As such, we used Unity's blacksmith character model featured at GDC 2015 [3]**.** Models and assets were found on Unity's Asset Store.

Scripts were to accommodate how aspects of the model (such as hair style and clothing) are changed dynamically. To swap out hair, we replaced the model's given hair with new hair. Within the script to handle this behavior, the previous hair was deleted and in its place was the new hair style. For changing hair color and clothing style, we located the corresponding body part within the character's hierarchy and modified the game object's textures.

In our prototype, we allow the player to customize four main features: hair, face, body and clothing shown in Figure 9. Within each feature we allow for further modifications.

Fig. 9: Main menu for our character customization application.

4.1. Adjusting Hair

Figure 10 shows the first submenu. Users are given a preview of different hair and beard styles. Users are able to change the model's hair and beard and the color of both separately by clicking on a particular style and color.

Fig. 10: Hair submenu

4.2. Facial Expressions using BlendShapes

Figure 11 shows the face submenu. Here, users are given a variety of options to change the character's face as well as skin color. Changes to the face as controlled with various sliders. Clicking and dragging the sliders from one end to the other will cause changes to the model's face. For example, the first slider will control how angry the model looks. The second slider changes how much the model's lips will protrude. The third slider raises and lowers the eyebrows changing how scared the model will look. The fourth slider adjusts how angry the model will look in the model's upper portion of the face. The fifth slider also adjusts the extent to which how angry the model looks in the model's lower portion of the face. Finally, skin color can be adjusted by clicking on any of the colored squares.

This animation along the character's face was created using BlendShapes [2] that is a feature of the Unity Game Engine. It allowed us

to control the expressions of the character which is essentially a feature in most games.

Fig. 11: Face submenu

4.3. Controlling Body Shapes by Shaders

Figure 12 shows the body submenu that allows players to control how large their character is. This submenu features 3 sliders that allows adjustment to the three main portions of the model's body: the head, torso and legs. Moving the sliders to the right causes their corresponding body parts to become thicker and bulkier while moving the sliders to the left causes body parts to become slimmer and leaner looking. This method was implemented using a vertex extrusion shader to control the facial structure of the character. This type of shader can also be used for procedural animation. This surface shader would move the vertices along their normals by the amount specified in the material of the character's body parts.

Fig 12: Body submenu

4.4. Changing Clothes

Figure 13 shows the clothing submenu. This submenu allows users to adjust the model's clothing style. For this project, we divided the model in two allowing the player to only adjust the tops and bottoms of the model. Users can adjust the tops and bottoms style independently by clicking on a preview style that they like.

Fig. 13: Clothing submenu

4.5. Animating Character using Mecanim and Rotation

Figure 14 shows how the model's stance is changed from idle to walking. With the idle stance, the model mostly moves from side to side. With the walking animation, the model can be seen walking forward with his sword in an attack position. Figure 15 shows how the model can be rotated. By clicking on the model and dragging with the mouse in either the left or right directions, the model can be rotated left or right to give the user a preview of how their character will look with all of the customization that has been done. These animations were created from Mocap data and we managed to use them using the Mecanim [1] feature to apply onto the character with ease. Mecanim allowed us to map the bone structure of the mocap animation onto the bone structure of the character which essentially makes the character animatable.

Fig. 14: Changing the model's animation from idle to walking

Fig. 15: Able to rotate the model by clicking and dragging the model

5. System Testing

Since our application was a character creator and was focused around how the character looks, our application was visually and manually tested by playing the application and clicking on every button and using every slider. We compared how the animations and model were being changed to how we expected them to change. If the model was found to not be in the correct position during play mode, we would preview the change while in play mode, update the camera position within the scripts and then apply the changes to our application.

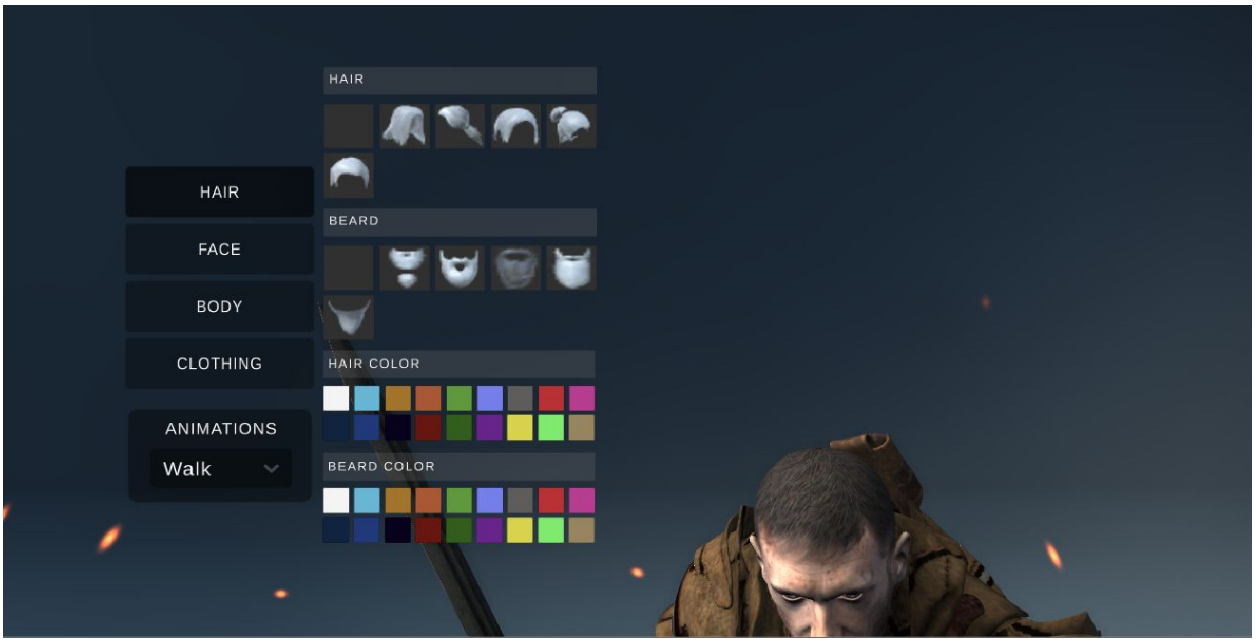

Fig. 16: Latest test shows that the model was placed too low. The camera angle needed to be adjusted.

For example in our latest test, we found that the camera angle was too high for the walking animation. This was because most of the application was tested using the idle stance where the model stands up straighter without hunching his back. The process that we went through to fix this issue is shown in Figure 17. While the application was being played, we adjusted the camera's position. After finding a suitable position, the coordinates were updated within the script files. This is the process that we went through for all tests involving position of game objects.

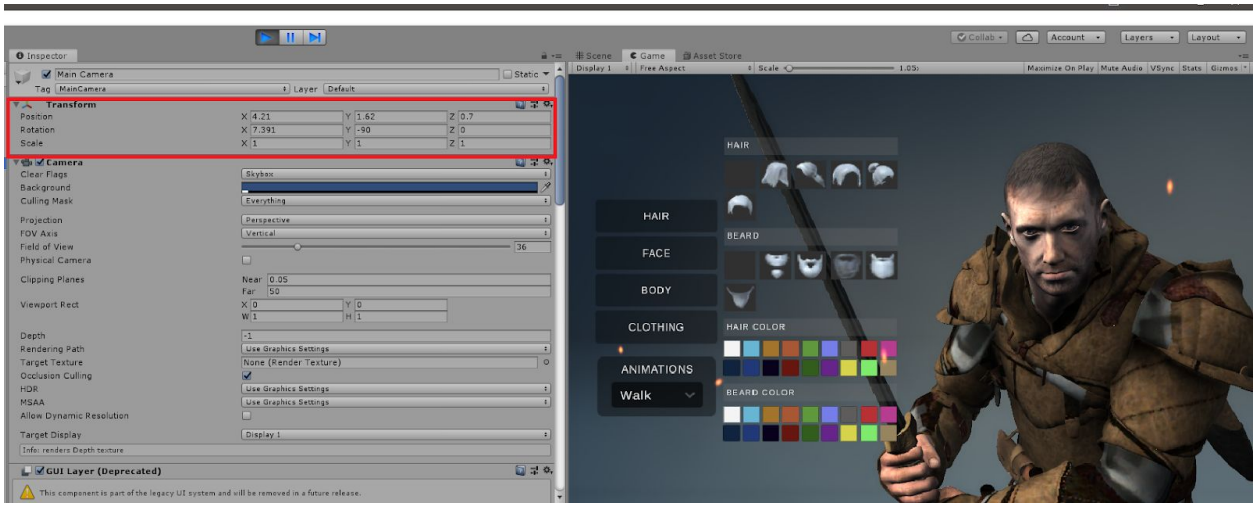

Fig. 17: To find a suitable camera angle, the the camera angle was adjusted while the application was being played. The main camera's transformation position was adjusted. After finding a suitable position, the (x,y,z) coordinates were updated within the scripts.

In addition to visual testing, we also extensively utilized the console log in our tests. This was particularly useful for hunting down and fixing unexpected errors and bugs. The console log was also helpful in finding unassigned reference exceptions shown in Figure 18. Since a lot of the buttons in our application directly reference different game objects within the scene, the console log was helpful in finding buttons that had missing references. Other errors such as missing or unparsable files due to incompatible Unity versions were displayed here which helped us find the files that was causing these errors.

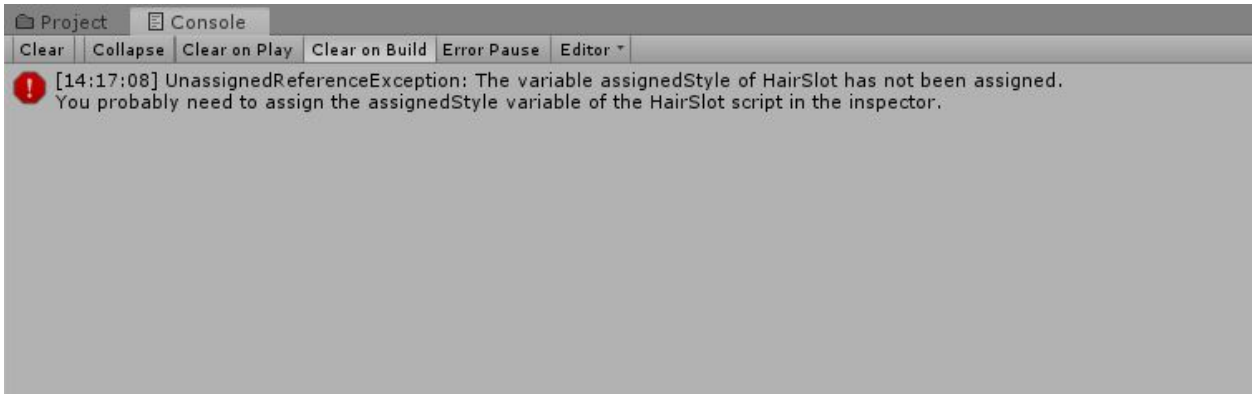

Fig. 18: Console log displaying an unassigned reference exception.

Despite all of our tests,there were some problems that we were not able to solve. In the beginning stages of our project, we encountered an error where the textures and materials for the model on Windows computers were not being correctly loaded. This problem was not found on Mac computers despite having the same exact files. We were not able to fix this error through extensive debugging. But miraculously, this problem seemed to fix itself toward the end of the project when we were focusing on model customization.

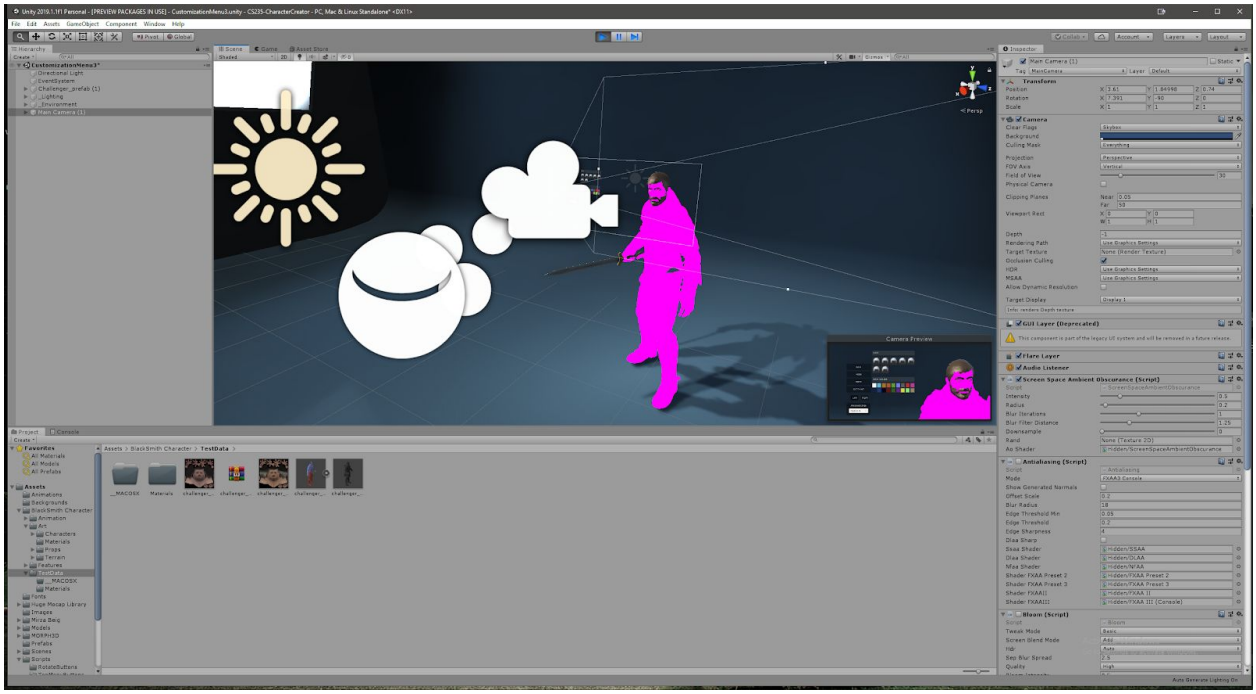

Fig. 19: The materials for the model was not being correctly loaded on Windows computers during the start of the development period of the application.

6. Conclusion

We were successfully able to build a basic prototype character creator tool using the Unity Game Engine. Emphasis was placed into creating an easy-to-use and nice looking UI. We drew inspiration from existing character creators from various video games. In particular, we thought that Code Vein's and Destiny's character creator were very good looking and so we emulated their UI into our design. Assets such as hair, clothing and textures were obtained for free from Unity's Asset Store.

Since this was just a prototype, we focused only on the most basic features for customization. Therefore, the future work that we will continue to work on would contain functionalities such as eye color, female character customization, better hair shader to look more appealing to the user, better lighting, clothing material and physics, more accessories/options to customize etc.

References

[1] U. Technologies, "Unity - Manual: Animation System Overview", *Docs.unity3d.com*, 2019. [Online]. Available: https://docs.unity3d.com/Manual/AnimationOverview.html. [Accessed: 17- May- 2019].

[2] U. Technologies, "Unity - Manual: Animation Blend Shapes", *Docs.unity3d.com*, 2019. [Online]. Available: https://docs.unity3d.com/Manual/BlendShapes.html. [Accessed: 17- May- 2019].

[3] "Pages - The Blacksmith - Unity", *Unity*, 2019. [Online]. Available: https://unity3d.com/pages/the-blacksmith. [Accessed: 17- May- 2019].