UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Character Creator Tool

Adrián Mirabel Martínez

Director: Roger Montserrat

Grau: Grau en Disseny i Desenvolupament de Videojocs (Barcelona)

Curs: 2021-22

Universitat: Center de la Imatge i la Tecnologia Multimedia (CITM)

Index

Abstract

People want to be identified in some form while playing video games, customization is a game element to accomplish that challenge.

This project will analyse diverse customization methods in video games, then identify best practices/features and reproduce them using Unity and Unreal Engine 4.

Those techniques will be mainly analysed from the programmer's point of view.

Later on, combining those techniques to create two final character creators one in 2D done in Unity and the other in 3D Unreal Engine 4.

Key Words

Customization, Character Creator, Game Engine, Unity, Unreal Engine 4 (UE4), Graphics, Blend shapes, Morph targets, Bones, Pixels

Links

Follow the development of the project on GitHub.

GitHub: https://github.com/M1R4B3L/Character-Creator-Tool

Releases of the tool will be done to take track of tool versions.

Releases: https://github.com/M1R4B3L/Character-Creator-Tool/releases

Tables Index

Figures Index

Glossary

Triple-A or AAA games: Video game industry informal classification to signify high-budget, high-profile games that are typically produced and distributed by large, well-known studios.

Double-A games are those developed typically outside of the large first-party studios.

Unity: Video game engine developed by Unity Technologies.

Pixel: The smallest unit of programable colour on a computer display or image.

Shader: Program that executes in GPU using algorithms and math calculations to compute per frame the result colour of each pixel.

UE4: Unreal Engine 4. Video game engine developed by Epic Games.

Blueprint Class: Asset that allows content creators to easily add functionality on top of existing gameplay classes, UE4 only.

Snapshot: Instantaneous picture or memory state copy saved the moment it's created.

Blend Shape/Morph Target: Snapshot of vertex locations for a specific mesh that has been deformed in some way.

Dynamic Material: Material/texture of a 3D object that can change in run time, UE4 only.

1. Introduction

Character creators are a customization technique designed as a tool to help developers create hundreds of characters in seconds instead of creating each character by hand, resulting in a massive reduction of production resources; a new tribe/race/family was created in days instead of months.

Later on, these tools started getting added to the video game, now the players were able to change the aspect of their in-game character. But programmers needed to add limitations for a better game experience and game story cohesion.

For this project the art point of view will be moved to the side, focusing all the efforts of the development on the design and programming part. To help with the creation of the tools I will be using Unreal Engine 4 and Unity. Although, this project could be done in other game engines like Godot, Cryengine, etc, or even in your custom engine.

Unreal Engine 4 will be used to replicate 3D techniques because most of the engine features are designed and adapted for 3D assets; leaving the replication of 2D techniques to be developed in Unity, an easier and more beginner-friendly game engine.

Before developing anything a study must be done to obtain desired and optimal results, that's why an analysis of a minimum of two games per rendering option 2D or 3D will be done accordingly.

To end this introduction, the final part of the project will be reproducing an in-game character creator using the techniques previously created, analysing some approaches done in the industry.

Fig 1

1.1 Motivation

First game I played in my life was Sims 3, I spend more time in the character creator section of the game than playing the actual game, I couldn't understand the menus; since then, I'm the type of guy who spends hours customising their character before playing the game, so I wanted to understand what was behind this game element.

Moreover, recent games are adding character creators to help users' immersion in the game, giving them autonomy to represent themselves and their image on it, I wanted to demonstrate that adding this feature into games is not as expensive and difficult as it seems from the outside.

That's why I want to analyse different techniques experts in the industry use to achieve the final result of a character creator. However, I am much more interested in the design and programming aspects behind that game feature.

1.2 Problem Formulation

Game customization is a must feature in nowadays video game industry.

Why do you ask? Most recent games are being designed and developed taking into account multiplayer gameplay, that's why players adopt an online image to represent them in front of the player base; in-game skins are a clear example of it. People can recognise popular players based on specific skin usage.

Excluding multiplayer games, each month new double-A or AAA games appear with their unique style of character customization, spending a lot of game pre-production, production and post-production resources (research, design, time, money, marketing…) on it. Beginner developers usually are not able to spend their development time adding customization, let alone multiplayer into the game.

Taking the previous problem into account companies have started to create tools to help developers. However, those tools are expensive or their usage is limited to the tool application, meaning that cannot be integrated into the game.

1.3 General Objectives

This project's main objective is to research and study multiple character customization practices in the industry, for later on displaying some of those techniques analysed and developed inside a game engine Unity, UE4.

Focusing on these objectives:

- To analyse 2D and 3D game customization techniques.
- To develop a UE4 tool.
- To develop a Unity tool.

1.4 Specific Objectives

As per specific objectives of this project, are as follows:

Character Creation Analysis:

- Study of a minimum of two 2D games that include a character creator and a 2D game/project that uses a customization method.
- Study at least three 3D games that include a character creator and a 3D game/project that uses a customization method.

Unreal Engine 4 development:

- Design of the character creator, combining previously analysed techniques.
- Design and development of the UI.
- Development of the techniques.

Unity development:

- Design of the character creator, combining previously analysed techniques.
- Design and development of the UI.
- Development of the techniques.

1.5 Project Scope

Character Creators are seen as tools for Triple-A or Double-A studios, but is an upswing feature in the video game industry. The main practice of this feature is outside of the game, used by designers and artists to create NPCs[1] to enrich the world and the story behind it.

Although is an incredible feature for all game developers, beginner or indie studios will lack the necessary knowledge to implement it correctly.

That's why this thesis is beneficial for them, working as a fast, easy-to-understand/use example.

[1] NPC: Non-playable character, players can usually interact with them.

2. State of the Art

2.1 Context

Character creators can be seen as modelling tools, video game features or even educational helpers. In this project, character creators will be seen as a tool to design an avatar, that later on will be used as a character inside a video game.

Character creators used in video games have a lot of features in common, from the layout of the creation screen to the techniques for modification of the avatar; what makes them special is the style of the game, design of the UI, optimization techniques, deployment platforms and the most important one **graphics**. Video game graphics define the core of the video game; they describe how the video game is going to be played and seen.

Depending on the video game graphics we can classify character creators into two big groups 2D and 3D.

2.2 Character Creators 2D

Graphics are what defines the game, in this case, 2D.

Taking into account the art style of 2D video games we can divide them into Vector or Raster graphics (Raster art is more commonly known as pixel art).

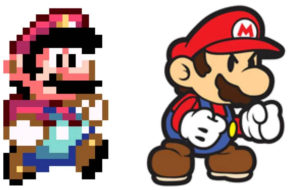

Fig 2.1

Vector Graphics: This is a form of computer graphics where visual images are created from geometric shapes defined in a plane, such as points, lines, curves and polygons.

Raster Graphics: This is a form of computer graphics where visual images are represented as a two-dimensional grid of square pixels.

These two computer graphics techniques need to be taken into account when creating a character creator.

2.2.1 Techniques for 2D games

To explain the techniques in a more practical way, they are going to be described using video games as a reference.

Sprite swap:

To explain this technique, we need to understand that a sprite is a two-dimensional image usually used for animations independently manipulated by the game. Usually, sprites are found in a sprite sheet for optimization reasons.

Pool of Radiance *(Fig 2.2)* used this technique to swap between portraits' body parts creating multiple options for your character. In this case, you were only able to change the head or body of your avatar.

Newer games like Don't Starve together *(Fig 2.3)* apply this technique to change character skins.

Fig 2.2

Fig 2.3

Unity is working in a new feature only for versions 2021.1 with 2D Animation 6.0 and over. This new feature enables users to change Game Object's renderer sprite at run time.

Sprite painting:

Painting an image usually is done in drawing software, creating the final sprite sheet for the character of the game. This technique creates and adds a new layer on top of the character sprite to modify body parts, add details or even change the sprite completely.

This technique has drawbacks, creating new textures in run time reduces game performance and texture size is limited.

Animal Crossing is the best example of it, creating patterns for your clothes.

Fig 2.4

2.3 Character Creators 3D

Character creators with three dimensions graphics use a three-dimensional representation of geometric data. This geometry is what forms later on game models.

Similar to 2D in 3D graphics we have two options to design our game; Geometry or Voxels.

Geometry: Most 3D games are created using models, what is a model? A model is a coordinate-based representation of any surface of an object in three dimensions.

Voxel: Represents a value in a regular grid in three dimensions. Similar to pixels in 2D.

2.3.1 Techniques for 3D games

Customization on 3D games usually are done through in-game skins, skins are just modifications of the base model, usually done in a 3D modelling software.

Accessories Changes: Similar to sprite swapping, 3D model accessories changes depend on the selected ones.

Apart from accessories, you are able to change skin colour, hairstyle…

This is usually done with a default character as the base and adding slots to hadle each accessory.

Fig 2.5

Blend Shapes/Morph targets: This technique is related to animation and usually created in 3D modelling software, morph targets are related to small modifications of the base model to create gestures and body deformations. They are really useful because artists are able to add thousands of new details just by modifying the model a handled amount.

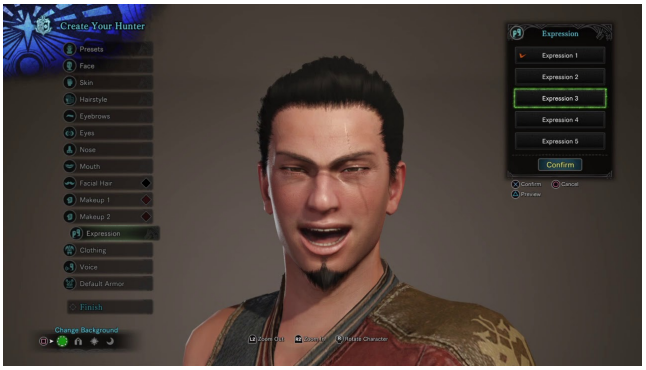

Fig 2.6

2.1 Market Study

While doing a market study we need to focus on three main points: The target, platform and sector.

In our case, target and platform are interrelated; most beginner and indie studios will start their development using Unity Engine, it offers developers adaptability for most common platforms Pc, Mobile even for new platforms like AR and VR. That's why indie studios, mostly localised in mobile gaming, have a better starting point in Unity than in many other game engines. Nevertheless, other platforms are not falling behind as we can observe in Unity's Gaming Report of 2022 (*Unity Gaming Report 2022*, 2022) multiplatform games released using Unity Engine increased by 200% in the last five years.

Unreal Engine has had exponential growth during the current years, with more and more middle and large-size studios switching to Epic Games' engine, thanks to the royalty policies of the company (*Unity Gaming Report 2022*, 2022) and the popularity of the engine in different industries, like the automobile or virtual production for filmmaking.

The last point we need to cover is the sector, our product enters the category of game customization. Personalization in video games is where companies/studios get most of their income, mostly from skins and cosmetic purchases.

Riot Games, in November 2021, included into their shooter game VALORANT a Champion bundle for the upcoming world championship, this bundle revenue surpasses $18 million. "Developer Riot Games earned $18.72 million in revenue from VALORANT Champions skins,…" (*Sapaz, 2022*).

Figure 2

To finish the market study, we take a look at similar products already released or in development. Fortunately for us, most of the products are tools for creating characters outside the game focusing on middle and big corporations, completely opposite to our objective.

(*3D Character Maker | Character Creator*, n.d.)

Figure 3

Only small tools found on the Unity and UE4 marketplace accomplish our goals, these tools are centred exclusively on one technique and don't offer a fully working character creator.

3. Project Management

3.1 Procedures and Monitoring Tools

For better development of the task, some tools will be used to manage the process in a clear and efficient way.

These are the following tools needed:

3.1.1 GANTT (Initial Plan)

GANTT tool will be used for the initial plan of the development process; for better clarification of the task I divided them into three parts: Green for **Research & Desing**, Blue for **Development** and Yellow for **Writing the Memory.**

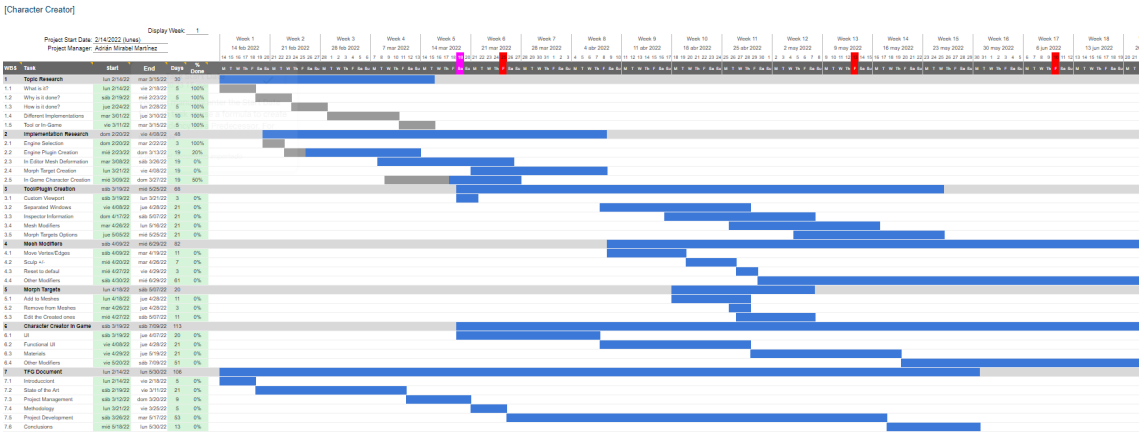

Table 3.1 - Initial Gantt

For better readability there is a link to the Gantt: [Gantt Chart Character Creator.xlsx]

3.1.2 Revision for September Delivery

During the development of the project, some issues were encountered, and implementation of the techniques was postponed to later days, not being able to achieve the proposed planning; the project was postponed to delivery in September.

Some changes were presented to decrease development time and accomplish the goals established:

Change the game engine: Unity will be used for 2D character creator, reducing the production time because the initial learning curve has already been surpassed.

Remove Editor Tools: We removed the in-editor features, following the risk and contingency plan, to focus exclusively on developing the techniques to accomplish the proposed objectives.

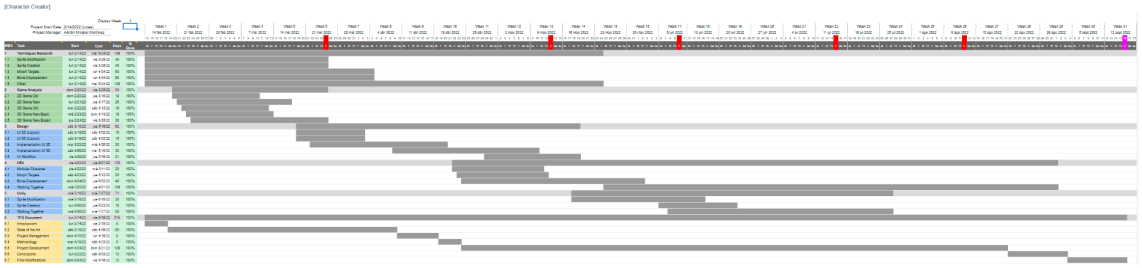

Table 3.2 - Revision Gantt

3.1.3 GitHub repository, version control tool

Github repository will be used as a version control tool, following the methodology of the project explained, later on, releasing new versions of the project to GitHub when a step of the methodology is reached.

3.1.4 Google Drive

Drive will be used as the document editor and file saver, every document related to the project will be uploaded into a folder to, later on, being shared with the director (if wanted).

3.2 Validation Tools

To validate the techniques analysed and implemented in this project, two simple character creators will be created by combing multiple techniques one in 2D and another in 3D.

If a technique is not accurately analysed (not being able to obtain the exact implementation as used in the studied feature), it will be validated if accomplishes a similar result as in the analysed game and the process is correctly recorded and documented.

3.3 SWOT

	Positive	Negative
Internal	**Strengths** - Previous experience with engines and the subject. - Knowledge of the workflow of character creation from the artist's point of view - Experience with Unity - Hundreds of hours spent with several in-game character creators.	**Weaknesses** - Little knowledge of the character creation workflow. - No knowledge of UE4. - No knowledge of character creators outside games.
External	**Opportunities** - Adaptability: Each developer will be able to adapt the tool effortlessly into their game. - Increase the popularity of the games that have a character creator/customization. - Free to use by all developers	**Threats** - Existing tools are done by big companies. - Popularity is causing bigger companies to start researching the topic.

Table 3.3. - SWOT

3.4 Risk and Contingency Plans

The most important part while developing a project is detecting problems and having a solution for them with the biggest preparation time possible.

This is the list of the most important problems that could appear during the implementation of the tool.

Risk	Solution
Time mismanagement: It's most probable to happen. The principal cause is not being able to follow the schedule proposed in the initial GANTT.	High Risk 1. The project will be adjusted accordingly to the remaining time. 2. UE4 editor implementation will be removed to focus on the main objectives.
Lack of knowledge 1: Not having previously worked in UE4 could cause a risk to the development.	High Risk 1. UE4 editor implementation will be removed to focus on the main objectives. 2. The project will be changed to adapt to this difficulty, changing UE4 to Unity.
Lack of knowledge 2: While analysing a game not being able to understand the complexity of the technique.	Medium Risk 1. As explained in the validation department the technique will be validated if accomplished a similar outcome. 2. No outcome is accomplished that technique will only be explained and not implemented.
Lack of knowledge 3: Unity Engine could cause a problem when switching.	Medium Risk 1. If Unity fails technique will be explained, but not implemented, and a simpler version will be replicated.
Validation Final Projects 2D & 3D: If some techniques are not combined and the final product is not accomplished.	Low Risk 1. Non-combined techniques will be explained and shown individually documenting how the combination process could have been done.

Table 3.4 Risk & Contingency

3.5 Initial Cost Analysis

Due to the usage of free software for students, the only cost for this project is labour cost and maintenance of the developer.

To make an estimation of the cost of this project, the developer will be earning 20€/h as a junior developer, with a total length of 6 working months.

Total Working Hours	6 month	300 h	
Description	Payment	Cost	Total Cost
Salary	per Hour	14,0 €	4.200,0 €
Equipment	Unique	1.235,0 €	1.235,0 €
Water	Month	15,8 €	15,8 €
Electricity	Month	40,5 €	40,5 €
TOTAL			5.435,0 €

Table 3.5 Initial Cost Analysis

4. Methodology

This project is a software project focusing on the implementation of different techniques based on previous game analysis. The most common methodology to represent software development in a real work environment is the Agile method; for this project, we will do one step more and specify the Agile process the development is gonna work on, a more specific Agile method and the one used in this project is called Feature-Driven Development (FDD).

4.1 Feature-Driven Development (FDD)

FDD is a framework in the Agile methodology for software development, following Agile methodology it divides the work into smaller increments. Later on, add those increments in an iteration process. (*What Is FDD in Agile?* n.d.).

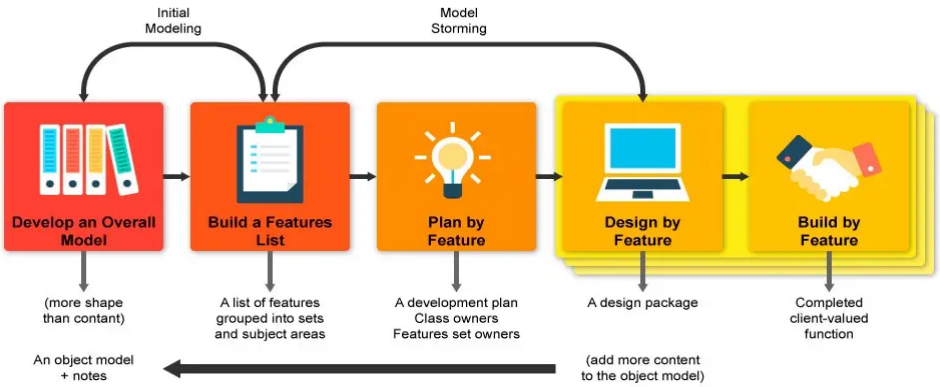

Figure 3

4.2 Applying FDD

FDD methodology is divided into 5 steps as seen in (Fig.3):

1. Develop the overall model: This step has been explained during the proposition of Project Objectives and then subdivided into Specific Objectives.

2. Build the features list: The feature list is composed of all the techniques analysed during the Development section.

3. Plan by feature: Individual features will be documented in the Development section after their analysis, separating them between both character creator options.

4. Design by feature: Design and explanation of the feature.

5. Build by feature: Implementation and documentation of the process.

When the process finishes an iteration happens to combine all the features together.

5. Development

This project will consist of the analysis and application of different techniques to develop a character creator system, those techniques will be explained during the development of the project separating them into two different applications. Each application will focus on one main technique; separating them will reduce development time and increase the production quality. I will not focus any effort on combining them together.

The first application consist of a 2D character creator, focusing on sprite modification, layering order, shaders and solving the problems the previous techniques could create.

The second application consist of a 3D character creator, focusing on accessories, camerawork, blend shapes, bone transformation and material modification.

5.1 Character Creator 2D

Character Creator 2D is an application developed in Unity Engine version 2020.3.22f1, it is one of the official releases that include LTS[2]. We will be using Unity because it provides better functionality for 2D applications and the techniques executed will be more comfortable to showcase and explain. Moreover, I have more experience developing applications in the Unity game engine.

The first step in the development is defining the art style of the application and the necessary sprites to achieve our desired outcome.

The application will be created using the 2D template the engine offers, we use the template because it provides preselected settings based on common best practices for different types of Projects. These settings are optimized for 2D projects across the full range of platforms that Unity supports.

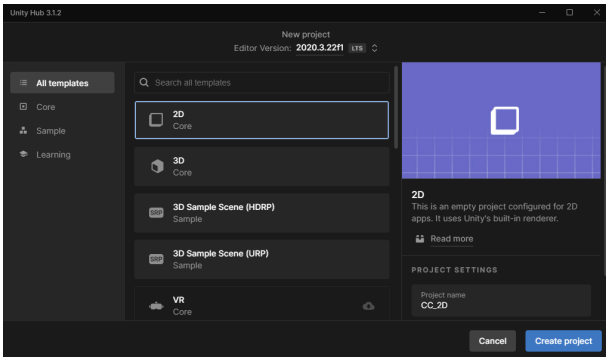

Figure 3 - Unity Template

[2] Long Term Support

5.1.1 Sprite Creation

For this application we are going to use a simple pixel art style, all the assets can be done in any 2D software that uses a pixel rendering for drawing images, for example, Aseprite, Adobe Photoshop, GIMP, etc.

Then it will be imported to the Unity project and added to the screen.

When importing a new sprite you will need to take into account the import settings of your sprite; adjusting them to accomplish the desired visualization of the sprite.

The most important settings are:

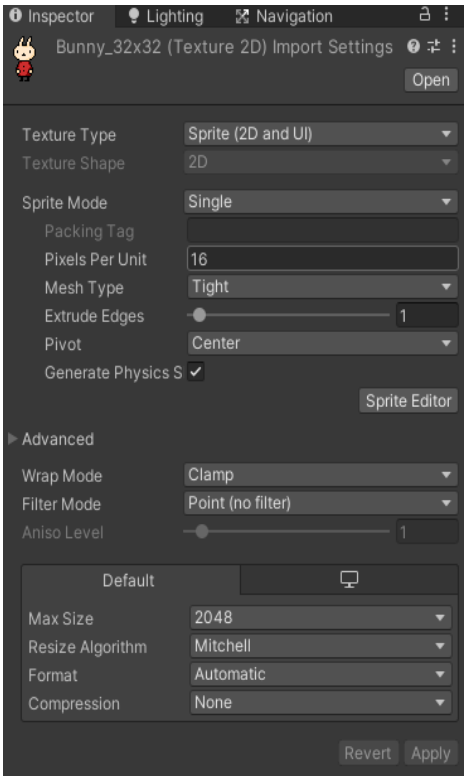

Figure 4 - Import Settings

- Sprite Mode: Specifies how the image is going to be used, in this case as we only have 1 character per sprite we use the single mode; if we had character animations in a sprite sheet we use multiple mode, it will handle each frame separately.

- Pixel Per Unit (PPU): It specifies the number of pixels corresponding to 1 unit of the Scene. In the picture on the right, we can see that it corresponds to a 32x32 texture with a 16 PPU meaning that all the sprite corresponds to 2 scene units.

- Filter Mode: It's a technique to improve texture quality by changing texture colours based on pixel proximity, for pixel art, it's better to remove it completely and have base colours instead of interpolated ones.

Figure 5 - Bad vs Good Usage

For this project, we are going to create 32x32 or 16x16 sprites for the character and the accessories.

We are using power of 2 textures sizes for GPU optimization, it's not necessary for new GPU cards but we are going to use it for easier calculations from sprite to scene size.

5.1.2 Sprite Order/Sorting Layers

Once we have our base character added to the scene we need to add some accessories for better customization, when adding something on a character we need to take into account the rendering order of the scene. In the image below we can see how affects the render order of a sprite, as we can see from the characters on the left and middle both of them have a layer sorting problem.

Figure 6 - Sprite Order

How do we solve this?

- The most straightforward solution is dividing the hat sprite into two parts, front and back, then adjusting the Z distance (in Unity engine Z axis gives us the depth of and object) of the game object[3] transform and adapting them ourselves. As we can see in the image on the right.

Figure 7 - Z distance

- Another solution will be accomplished if we separate the hat into two parts like before, but we add sorting layers to each sprite part, which will give us more control over the sprite without needing to change transform positions.

[3] Game Object: is the scene representation of any item in Unity, like characters, props, etc.

5.1.3 Sprite Modification

Changing a sprite could be seen from two points of view:

- Total Freedom: the user is able to draw new pixels and remove the previous ones created by ourselves wasting our time and effort on something not necessary.

- Limitation: we limit the user's actions to specific item/colour combinations; making him feel in a freedom state, but having some kind of control to reduce problems during the gameplay time.

For this project, I'm focusing on giving the users some item/colour alternatives; item variation will be done in a 2D pixel software, but colours for those items will be controlled inside the engine using shaders.

To create the shader we have two options use a programming language called HLSL or a shader graph. A shader created using a graph will be easier to implement and more visually appealing.

How to start? Firstly we create our shader graph, we reference the texture we want to change in this case our character and we convert it to a Sample Texture 2D to obtain the colours for each pixel on the texture.

Then, we replace the colours of each pixel we are interested in using a colour picker, outputting them in a final texture.

5.2 Character Creator 3D (Basic)

Character Creator 3D is an application developed in Unreal Engine 4 version 4.26.2. In this application, I want to focus on bone transformations, blend shapes and changing accessories and materials. UE4 has the fastest implementation pipeline to quickly implement the necessities of this application.

5.2.1 Default Character

The default character will be obtained from the UE4 marketplace called Stylized Character Kit: Casual 01 created by ROCKETARTS (Polo, n.d.).

The character we obtained is composed of separated skinned mesh objects corresponding to each of the character's body parts, we need a way to merge all of them together.

First, we create a new Blueprint Class Character that will be our character for the application.

To accomplish our final character we are using a node called Set Master Pose Component, this node will be placed inside the construction script of our third person character actor, the node is composed of 2 main inputs:

Target: Meshes that will be merged together in the same skeleton.

New Master Bone Component: The main skeleton where all the other skeleton meshes will take as reference.

Using this technique instead of a C++ script to merge all the meshes will help us in future development, giving us the opportunity to use morph targets from each mesh.

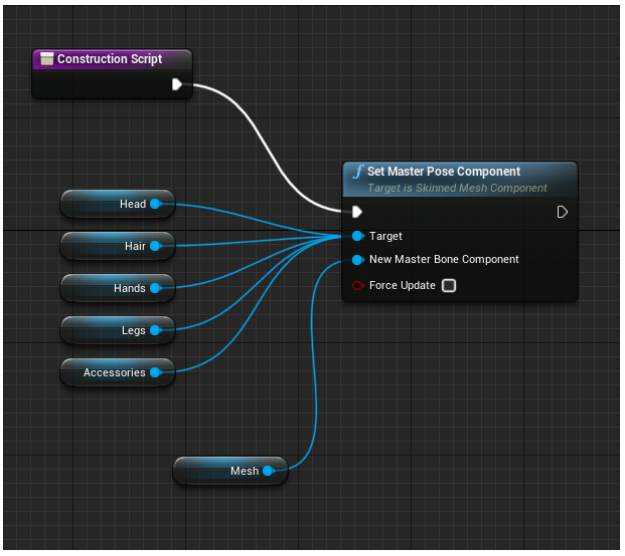

Figure 8 - Master Pose Component

5.2.2 Accessories

Once we finally have our base character we need to change its meshes to add different accessories.

First, we are going to create a Blueprint Enumerator to identify the different body parts the character is composed of. In my case Hair, Head, Torso, Arms, Legs and Accessories.

Subsequently, we create a character struct to identify and save the character parts, to be used later on in the UI to change them at run time.

5.2.3 Camera

Camera it's really important to emphasise what the player is currently changing; we need to create some camera animations based on which bone the player is currently selecting.

Those camera animations are created inside the camera actor that will work as our first-person character.

First, we obtain the mouse position of the user, we convert it to world position to identify which bone/skeletal mesh is selected and we do a zoom animation focusing on that body part; it will do the same animation when the body part button is clicked in the UI.

5.2.4 Blend Shapes

When the character is finally merged we can access the morph targets of the imported skeletal meshes, from the character blueprint we can get and set the current value of the blend shapes using a node called Set the value of Blend Shape, what we need to do is change that value using a slider in the UI of the application calling that node and changing its value in run time.

5.2.5 Bones

Similar to the Blend Shapes we could obtain and modify the transformation of the selected bone, in this case, we want to use the bone transforms to modify the width and height of our character. Modifying them individually will create body dysphoria, that's why we need to scale our bones depending on different factors. These factors are included as variables in the UI of the application.

5.2.6 Materials

Using the UI and some custom variables inside the materials we are able to modify the body parts' colours. This implementation will not work at run time that's why we need to create Dynamic Material Instances to change the materials at run time.

6. Conclusion

This project is a combination of good practices and bad time management.

During the last days of development, problems appeared making the final delivery impossible to accomplish, I worked alone during the project (missing the Director).

7. Bibliography

- Polo, A. (n.d.). *Content by ROCKETARTS - UE Marketplace*. Unreal Engine. Retrieved May 13, 2022, from https://www.unrealengine.com/marketplace/en-US/profile/ROCKETARTS?count=20&sortBy=effectiveDate&sortDir=DESC&start=0

- *Release and Royalty Tracking Guidelines*. (n.d.). Unreal Engine. Retrieved February 26, 2022, from https://www.unrealengine.com/en-US/release

- *Unity Gaming Report 2022*. (2022). Unity Gaming. Retrieved February 26, 2022, from https://create.unity.com/gaming-report-2022

- Sapaz, A. (2022, March 3). *Sources: VALORANT Champions skins revenue breaks $18.7 million*. Upcomer. Retrieved February 26, 2022, from https://upcomer.com/sources-valorant-champions-skins-revenue-breaks-18-7-million

- *3D Character Maker | Character Creator*. (n.d.). Reallusion. Retrieved February 27, 2022, from https://www.reallusion.com/character-creator/

- *Unity Gaming Report 2022*. (2022). Unity Gaming. Retrieved February 26, 2022, from https://create.unity.com/gaming-report-2022

- *What is FDD in Agile?* (n.d.). Wrike. Retrieved March 12, 2022, from https://www.wrike.com/agile-guide/faq/what-is-fdd-in-agile/