Bachelor's Thesis

Information and Communications Technology

2024

Sami Tikkanen

Developing a 3D Mesh Deformation Shader in Unreal Engine 5

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2024 | 37

Sami Tikkanen

Developing a 3D Mesh Deformation Shader in Unreal Engine 5

The thesis examined the Unreal Engine 5 game engine (UE5), its purpose and use cases. The goal was to develop a deformation shader system for 3D models using Unreal Engine. The development process aimed at making the system reusable and easily modifiable for various projects in the gaming and film industries.

The choice of Unreal Engine 5 for this thesis was based on the author’s familiarity with the platform and its reputation for innovative technologies widely embraced in both gaming and film industries.

Deformation is a mechanism that adds realism and dynamism to virtual worlds. It enables real-time shape transformation in objects in response to external forces, such as collisions or pressure. Deformation can enhance immersion and user engagement in the gaming environment.

The thesis resulted in the development of a functional Render Target and a vertex-based deformation shader. These tools are designed to be easily customized and integrated into various projects. Users can adjust deformation, stiffness, and durability parameters using pre-created customizable components, allowing for versatile application across different projects.

Keywords:

Unreal Engine, Deformation, Shader

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2024 | 37

Sami Tikkanen

3D-mallin deformaatiovarjostimen kehittäminen Unreal Engine 5 pelimoottorilla

Opinnäytetyössä tarkasteltiin Unreal Engine 5 -pelimoottoria (UE5) sekä sen käyttötarkoitusta ja toimintamalleja. Tavoitteena oli kehittää Unreal Enginen avulla 3D-mallin deformaatiovarjostinjärjestelmä. Kehittämisprosessissa tavoiteltiin järjestelmän uudelleenkäytettävyyttä ja helposti muokattavuutta erilaisiin peli- ja elokuvateollisuuden projekteihin.

Unreal Engine 5:n valinta tässä opinnäytetyössä perustui tekijän aiempaan kokemukseen alustan kanssa sekä sen asemaan edelläkävijänä teknologiassa, jota käytetään laajalti niin peli- kuin elokuvateollisuudessakin.

Deformaatio on mekaniikka, joka tuo realismia ja dynaamisuutta virtuaalimaailmoihin. Se mahdollistaa objektien muodonmuutokset reaaliajassa vastauksena ulkoisiin voimiin, kuten törmäyksiin tai paineeseen. Deformaation avulla voidaan lisätä immersiota ja käyttäjän eläytymistä peliympäristöön.

Opinnäytetyön tuloksena kehitettiin toimiva Render Target ja vertex-pohjainen deformaatiovarjostin. Nämä työkalut on suunniteltu helposti mukautettaviksi ja integroitaviksi erilaisiin projekteihin. Käyttäjät voivat säätää deformaation, jäykkyyden ja kestävyyden parametreja ennalta luotujen komponenttien avulla, mikä mahdollistaa monipuolisen soveltamisen eri projekteissa.

Asiasanat:

Unreal Engine, Deformaatio, Varjostin

Contents

Figures

List of Abbrevations and Symbols

3D	Three-dimensional, a format that provides the perception of depth in computer graphics, simulations and visual media.
Actor	Any object that can be placed or spawned within a game world.
Blueprint Scripting	Unreal Engine’s own visual scripting system.
C++	Programming language.
Epic Games	American video game and software developer company.
GPU	Graphics Processing Unit.
Hitscan	Refers to an instant-hit detection method where the game checks for a target along a straight line.
Line Trace	Technique used to detect objects along a specified line, often utilized for simulating ray-casting or hitscan mechanics.
Material	Set of properties that define the appearance of a surface, including its color, texture and reflectivity.
Material Function	Reusable snippet of shader code used within Materials to encapsulate shading operations.
Material Instance	A version of master Material enabling property adjustments without creating separate Materials.
Pawn	Type of Actor that can be controlled by the player or the game’s AI, serving as base class for all characters.
Unreal Engine	Game engine created and developed by Epic Games.

1 Introduction

Deformation occurs when physical stress is applied to an object, resulting in a change in its shape. (DuBose 2023). Various elements in a game necessitate mesh deformation, such as grass swaying in the wind, character interactions, waves in the water, and terrain features like footprints in snow. Consequently, it is evident that mesh deformation plays a crucial role in a wide array of games and genres. (Merzlikin 2022.)

Over the past few decades, the gaming industry has experienced significant evolution in gaming hardware, revolutionizing how people play and experience games. From the early gaming consoles to the emergence of high-end PC gaming, technological advancements have been crucial in shaping the gaming landscape and fueling the industry's growth. (FasterCapital 2024.)

The continuous advancement of processors, graphics cards and memory has broadened the possibilities for developers, resulting in more complex and visually impressive games. PC gaming has became innovative, continually pushing the boundaries of what is possible in gaming industry (Devereaux 2023). Deforming the vertices within a mesh until recently, was simply too expensive to consider in real-time. With the rise in processor speeds, soft-object animation is becoming more accessible and efficient. (Ferrier 1999.) Real-time methods demand less design time and can yield more realistic results. However, they often compromise performance. (Thomas & Zhang 2023.)

This thesis outlines the tools and techniques used in creating the mesh deformation shader system, with a focus on shaders, Render Targets, UV mapping and the utilization of Unreal Engine 5, detailing the information of deformation shader, covering its fundamental principles and how it is integrated within the game engine. Also, the thesis will go into the background of Unreal Engine and its use cases for games, films and other applications. Turku University of Applied Sciences served as the commissioner for this thesis.

2 Background of Unreal Engine

The first-generation Unreal Engine was developed by Tim Sweeney, the founder of Epic Games. (Sweeney 2005.) Unreal Engine is a game engine developed by Epic Games, first showcased in the 1998 first-person shooter game “Unreal”. Initially developed for PC first-person shooter, it has been successfully used in a variety of other genres, including platformers, fighting games, MMORPGs, and other RPGs as well.

Written in C++, Unreal Engine features a high degree of portability, supporting a wide range of platforms. Unreal Engine is a complete suite of creation tools for developing from independent hits to blockbuster franchises. (Raturi 2023.)

2.1 Games

Unreal Engine was originally designed to be used as the underlying technology for video games. It has been predominantly utilized in creating first-person shooters, expansive open-world adventures and complex strategy games usually in 3D environments. Unreal Engine tends more towards being used for higher-end projects, for example “ARK: Survival Evolved”, “Borderlands 3”, “MORDHAU”, “PlayerUnknown’s Battlegrounds”, “Sea of Thieves” and “XCOM 2” (Doucet & Pecorella 2021). However, the engine has gained popularity also among independent developers, as it offers robust features, exemplified by visual scripting (Blueprint Scripting) simplifying complex mechanics. With its extensive asset collection, The Unreal Engine Marketplace simplifies production while the supportive community offers valuable guidance. (Phuc 2023.)

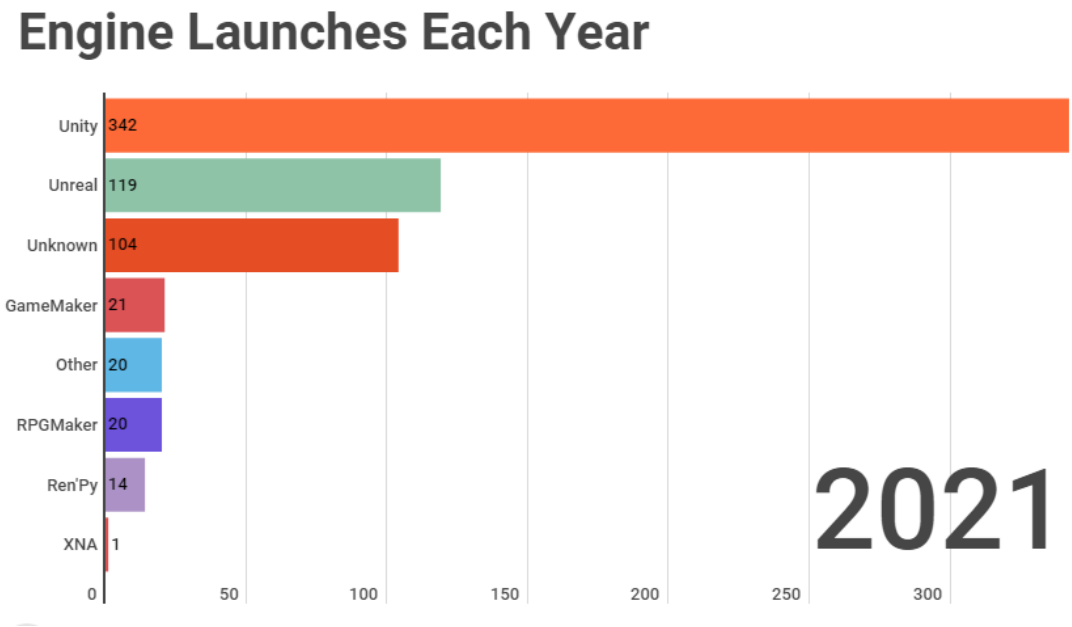

Figure 1. Annual releases on Steam, priced at minimum of $4.99 and having at least 50 reviews (Pecorella 2021a).

Unreal Engine games launch seemingly at the highest prices. Supporting the view that Unreal is preferred for larger-scale projects, 25 % of Unreal games launch at a price of $29.99 or higher, compared to about 6 % of Unity Games (Figure 2.). Games priced at $49.99 or more are almost exclusively developed using Unreal or custom engines – meanwhile, 85 % of all games are released at a price below $19.99. (Doucet & Pecorella 2021.)

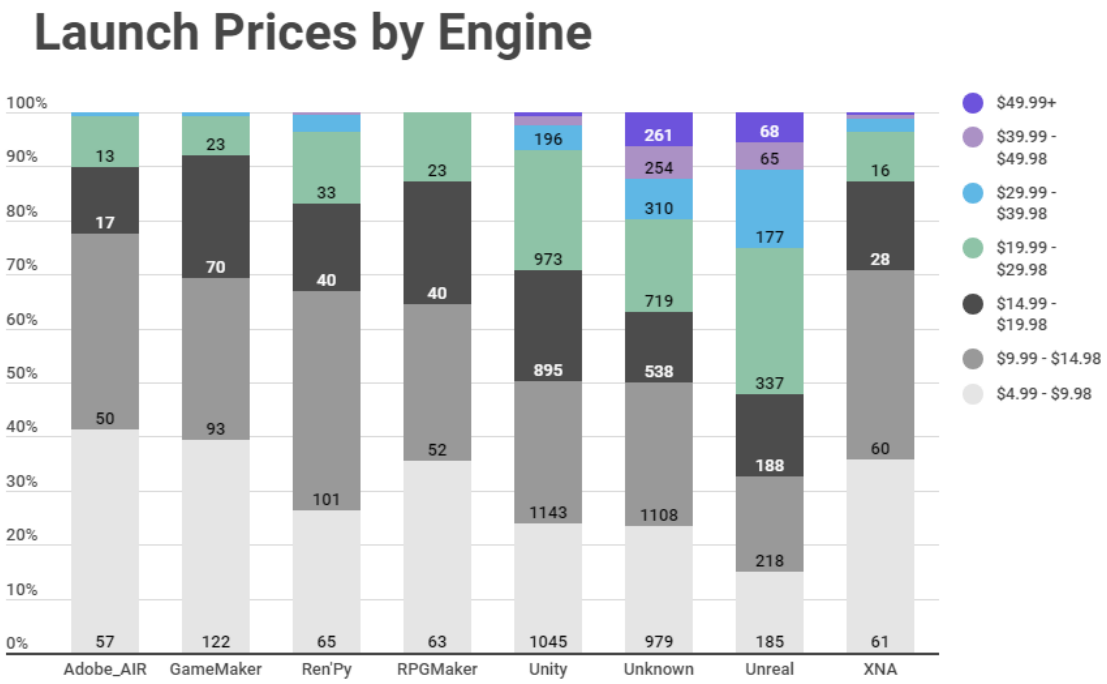

Figure 2. Engines with over 100 games, each having 50 or more ratings, released in 2016 or later, and priced at $4.99 or higher. (Pecorella 2021b)

It is also essential to recognize that Epic Games operates their own digital store, the Epic Games Store, where a significant portion of the available game utilize most-likely the Unreal Engine. However, the store supports games made with any engine. The Epic Games store is open to games developed with any engine, featuring initial titles from Unreal, Unity and internal engines. (Sweeney 2018.)

2.2 Films

Unreal Engine, known for its innovative features and adaptability, has emerged as a resource for filmmakers who want to expand creative limits and realize their imaginative concepts. As the worlds of video game graphics and movie visual effects blend, Unreal Engine is ideally suited to help all creative professionals achieve their visions, including those in filmmaking. (Dean 2023.) The engine's unparalleled real-time rendering capabilities have significantly influenced the film and television industry, revolutionizing visual effects and animation.

Unreal Engine's pivotal role in the development of virtual production techniques is exemplified by "The Mandalorian". This groundbreaking series utilized Unreal Engine to render photorealistic environments in real-time on LED screens surrounding the actors, significantly reducing the dependency on post-production CGI. This approach has transformed the filmmaking process, allowing for dynamic adjustments to lighting and backgrounds on set, thus enhancing the creative process with a level of flexibility previously unattainable with traditional CGI or green screens. (Farris 2020.)

Unreal Engine's advanced real-time rendering capabilities have revolutionized the pre-visualization (*previs*) process in filmmaking. By enabling high-fidelity, real-time visuals, Unreal Engine allows filmmakers to explore and refine visual concepts with unprecedented flexibility and efficiency long before the cameras roll. This marks a significant shift in the pre-production phase, fostering a more dynamic and collaborative environment where creative visions can be iterated upon, adjusted, and perfected in real-time. (Pohl 2017.)

The integration of Unreal Engine into previs workflows, as demonstrated by HALON Entertainment's innovative use of the engine for film projects, exemplifies how filmmakers can leverage game engine technology to push the boundaries of visual storytelling.

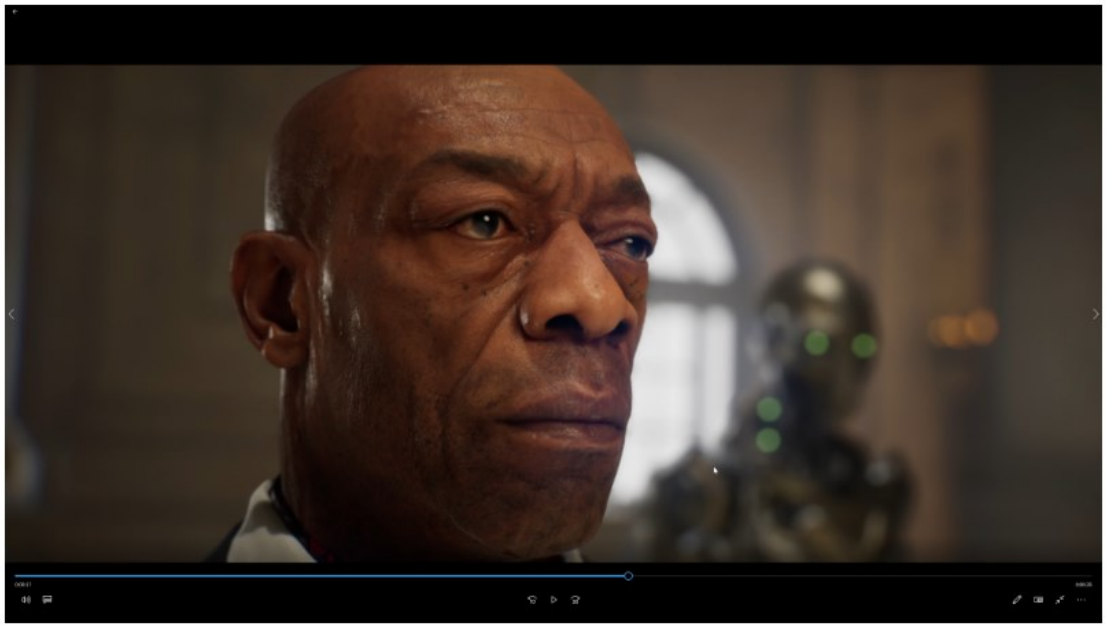

Figure 3. A metahuman created by the artist Aaron Sims using Unreal Engine. (Aaron Sims Creative 2023)

2.3 Other uses

Unreal Engine has also found applications in non-creative fields due to its accessibility and extensive features. It has served as the foundation for a virtual reality tool to study pharmaceutical drug molecules in collaboration with researchers, as a virtual environment for designing new buildings and automobiles, and for real-time graphics on cable news networks. (Yee 2018.) In March 2012, Epic Games revealed a collaboration with Virtual Heroes of Applied Research Associates to establish the Unreal Government Network, a program designed to oversee Unreal Engine licenses for government agencies. (Robertson 2012.)

This support agreement has led to the creation of several projects -- such as an anesthesiology training program for U.S. Army doctors, a multiplayer crime scene simulation for the FBI Academy, and various tools for the Intelligence Advanced Research Projects Activity. The latest mentioned was designed to help intelligence analysts identify and reduce cognitive biases in their work. (Brightman 2012.)

3 Tools and Techniques

3.1 Material Editor

The Material Editor is a node-based graphical interface. It allows you to design shaders for your geometry, including Static and Skeletal Meshes, and can also be integrated with systems like Cascade and Niagara to develop unique materials. (Unreal Engine Documentation 2024a.) In this thesis project, the Material Editor acts as a backbone in implementing the vertex deformation shader.

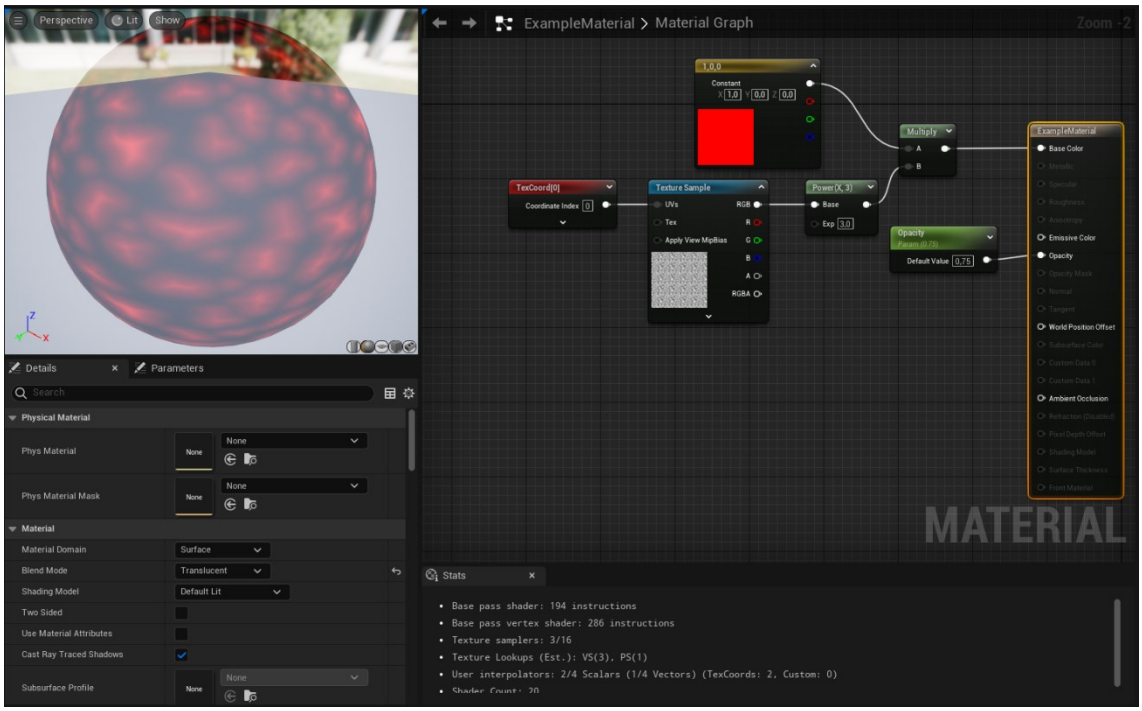

Figure 4. Material Editor in Unreal Engine 5.

Additionally, there is also a way to generate shader code by using “High-Level Shader Language” (HLSL), which offers a more direct, traditional and flexible approach to shader programming. This thesis project will use the node-based Material Editor (Figure 4) instead of HLSL.

3.2 Shaders

In computer graphics, a shader is a computer program that determines the correct levels of light, darkness, and color when rendering a 3D scene. (Ponnam 2023). Shaders are compact programs intended to run on a GPU. Due to the high speed of this hardware, these programs are minimal, with straightforward instructions, and are designed for parallel execution across numerous elements. Typically, these elements are geometry vertices and pixels that need to be rendered on the screen. However, some shaders, known as compute shaders, are more versatile and can process raw data that is not intended for direct display. (VFXDoc 2024.)

Vertex shaders can be utilized in video games to create various effects – character animation, object deformation, and camera movement. They enable real-time manipulation of vertex positions, allowing for dynamic interactions and realistic movements. Utilizing vertex shaders allows developers to craft visually compelling and immersive gaming experiences with precise control over vertex transformations and animations. (Schouzib I. 2023.)

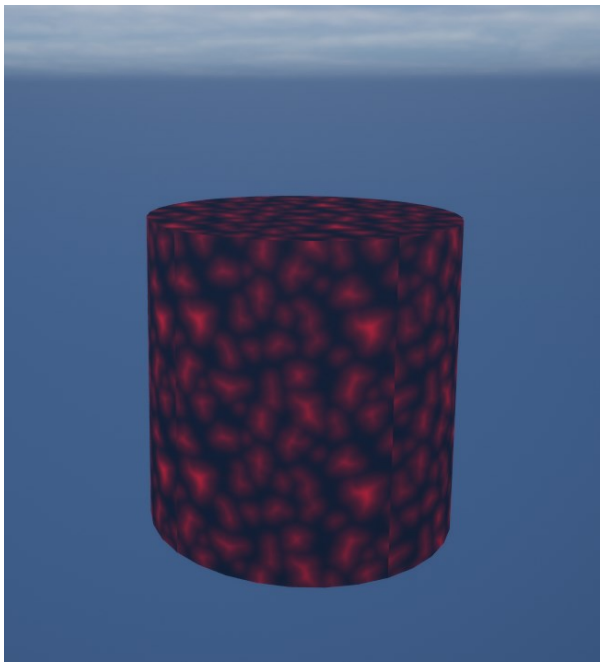

Figure 5. Simple shader applied to a cylinder 3D mesh.

3.3 3D Mesh

A 3D mesh forms the structural foundation of a three-dimensional model, composed of polygons. These meshes use reference points along the X, Y, and Z axes to define shapes with height, width, and depth. A 3D mesh model represents an object in three dimensions, consisting of a collection of faces, edges, and vertices that outline the object's shape and structure realistically. Each edge connects two adjacent vertices, which are points in 3D space. The faces, or polygons, make up the object's surface by enclosing the edges. Polygons in a 3D mesh are usually quadrangles or triangles, and these shapes can be broken down further into vertices defined by X, Y, and Z coordinates and lines. (Awati 2024.)

Unreal Engine uses “Static Meshes” for objects that do not change or move, such as furniture and buildings. Static Mesh assets are the fundamental units for constructing world geometry in levels created with Unreal Engine. These 3D models are designed in external modeling software such as 3dsMax, Maya, or Blender, and then imported into the Unreal Editor. Most levels created in Unreal Engine will predominantly consist of Static Meshes, typically used as Static Mesh Actors. (Unreal Engine Documentation 2024d.)

“Skeletal Meshes” are the foundational asset for characters in Unreal Engine. They contain the character’s visual mesh, or geometric model render of the character, and the character’s skeleton that contains bone data, which is used to animate the character. (Unreal Engine Documentation 2024e.)

High polygon counts, while providing detailed and visually appealing models, can strain game engines, leading to slower rendering times and potentially impacting overall game performance. Conversely, models with too few polygons might lack the necessary detail for a realistic and immersive gaming experience. (3D-Ace 2024.) However, the mesh deformation shader is vertex-based solution, meaning that the meshes should possess a higher vertex count than typically required.

3.4 UV Mapping

UV mapping is a technique for surface parameterization that converts a 3D surface into a 2D image, effectively creating a 2D coordinate system known as a UV map for a polygonal mesh. This is done by explicitly mapping UV coordinates to every vertex of the mesh. (Yu, Dai, Li, Ma, Liu & Qi 2023.) Usually the UV map serves as a bridge between a 3D model and a 2D texture (Robinson n.d).

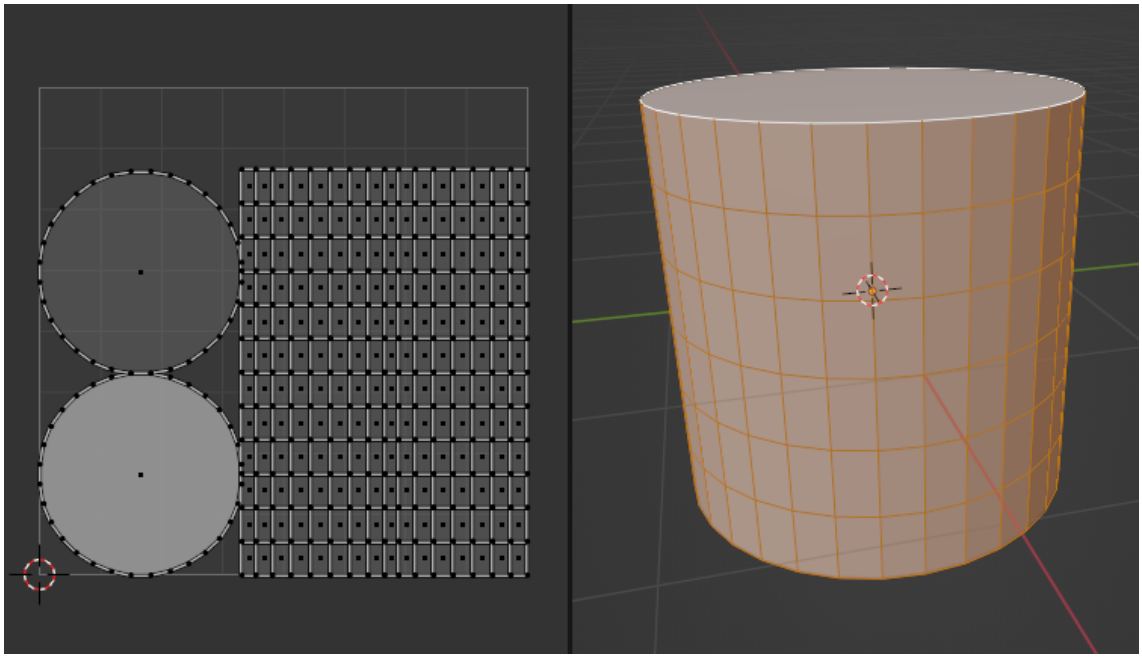

Figure 6. UV map of a simple subdivided 3D cylinder mesh.

The Mesh deformation shader uses UV maps to display the exact coordinates, where the deformation on the 3D mesh is intended to be displayed. While a 3D model mesh can have up to seven different UV Channels (Unreal Engine Documentation 2024f.), it is important to note that the deformation shader employs only a single UV map (Figure 6). Multiple UV maps or overlapping UV islands would complicate and interfere with the shader's deformation projection.

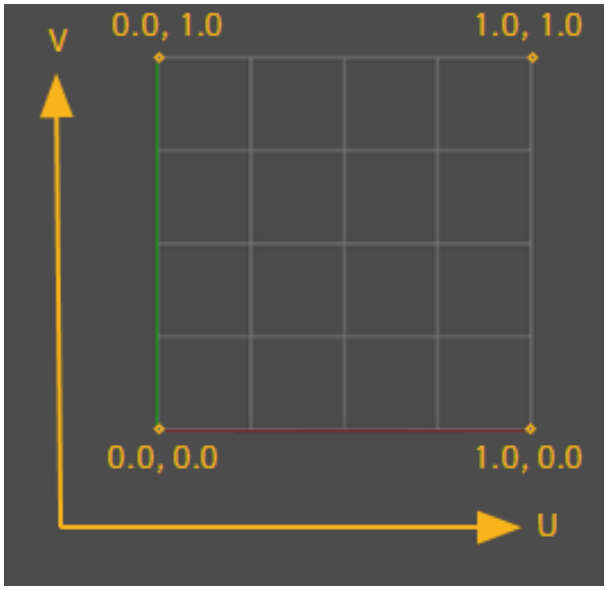

Figure 7. 2D coordinates of UV map. (Foundry Learn 2024)

3.5 Render Target

In computer graphics programming, a render target is a section of computer memory where the next frame to be displayed is created. Primarily used to boost rendering speed, the render target typically resides in dedicated memory on the graphics card near the GPU. In 3D graphics programming, render targets can optimize the rendering of objects that utilize images for their surface textures. (Eugene P. 2024.)

The thesis project uses Render Target for a specific purpose: capturing impact positions within the UV map. It captures 2D impact position coordinates and writes them runtime into a Render Target, which can be thought as an empty black texture. By utilizing this technique, it can precisely pinpoint the areas affected by collisions or other dynamic events. Once these impact positions are captured, they serve as data points for the mesh deformation shader.

Important consideration is the memory consumption associated with Render Targets, especially at higher resolutions. Render Targets can vary significantly in memory usage depending on their resolution. Higher values will increase image quality but at the cost of more video memory (Kodeco 2018). High-quality

resolutions, such as 2048x2048 or 4096x4096, while offering great detail and precision, are substantially more memory-intensive.

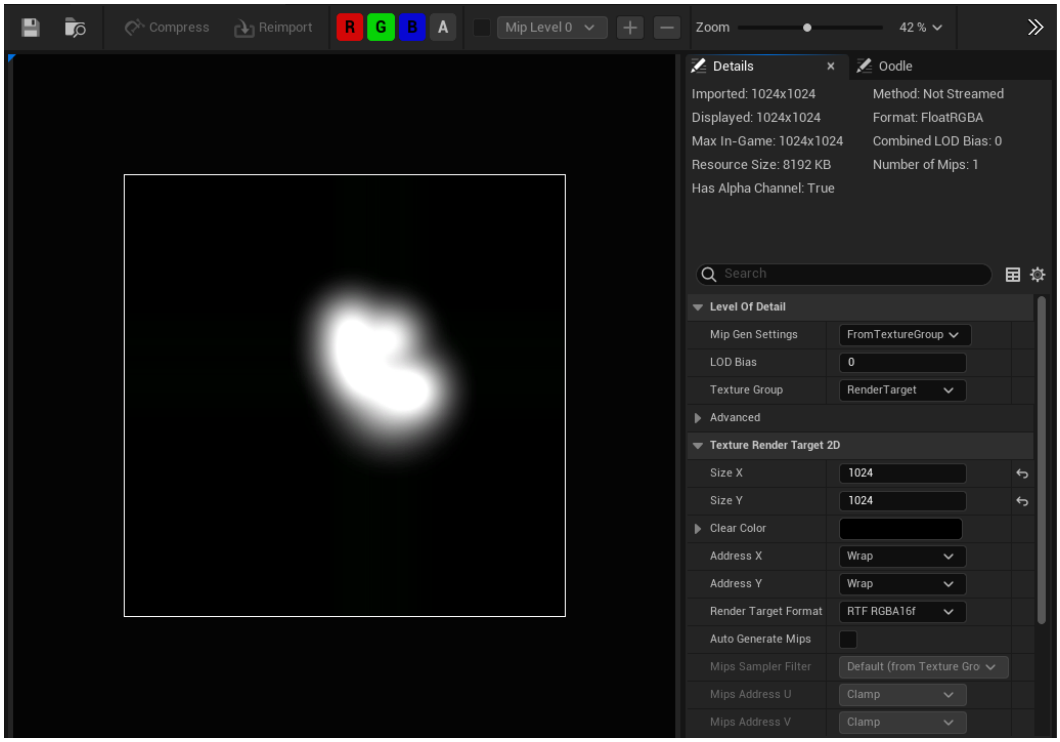

Figure 8. Render Target displaying the deformation in 2D space.

3.6 Texture Masking

A Texture Mask is typically a grayscale texture or a single channel (R, G, B, or A) of a texture, used to constrain the area of an effect within a Material. (Unreal Engine Documentation 2024b.)

In this project, the generated Render Target can be directly used as an mask, which is already an grayscale texture. The darker areas of the Render Target can represent regions where the effect is minimized or entirely absent, while lighter areas indicate where the effect should be fully applied (Figure 9). This technique not only simplifies the workflow by eliminating the need for a specific texture mask but also provides a high degree of control over the deformation effect, giving more precious and realistic results.

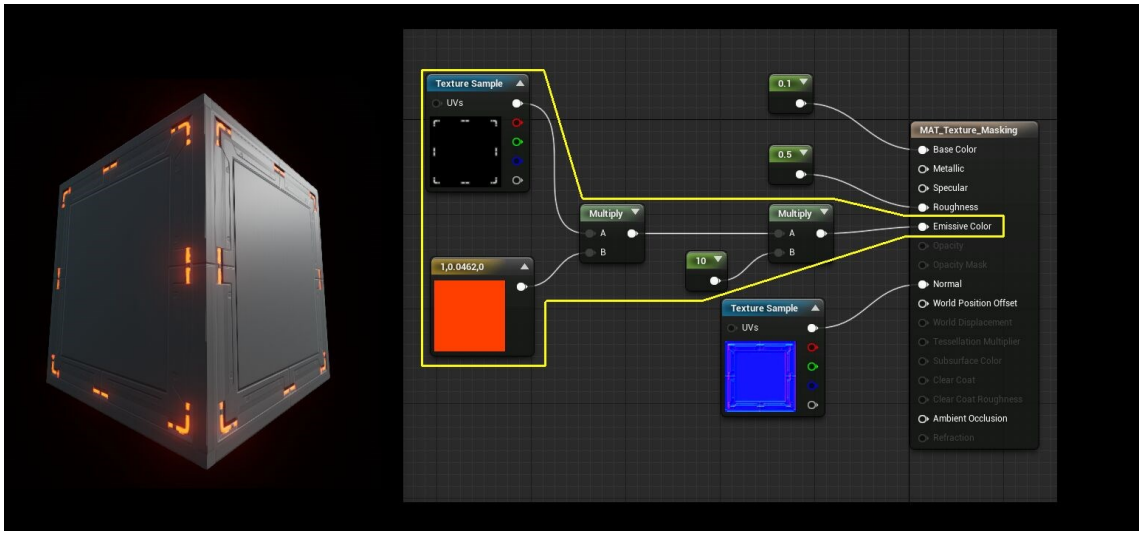

Figure 9. The Texture Mask is being used to mask emissive values to the Material. (Unreal Engine Documentation 2024b.)

3.7 Actor Components

In Unreal Engine, everything the player sees or interacts with in the game world is ultimately managed by some type of component. This is because components are essential for rendering meshes and images, implementing collision, and playing audio. The Actor Component serves as the base class for all components in the engine. (Unreal Engine Documentation 2024c.)

Actor Components can be both Blueprint or C++ based. These components provide a principle of reusability (Unreal Engine Documentation 2024g.). This design philosophy allows for the same component to be utilized across various Actors, significantly reducing development time and encouraging a modular, efficient approach to building game worlds.

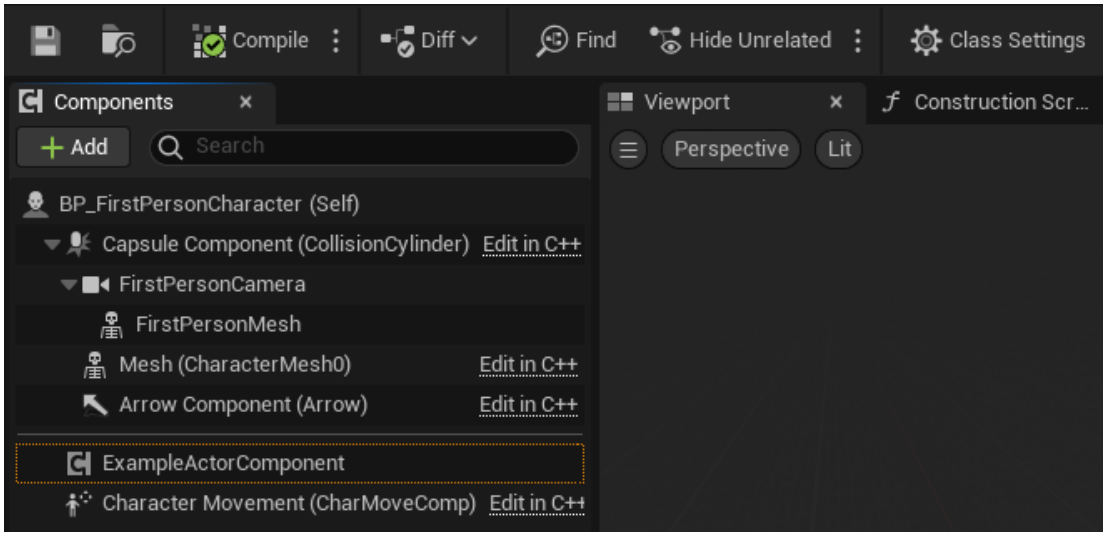

Figure 10. Example Actor Component attached to a Player Pawn Blueprint.

4 Mesh Deformation

Deformation occurs when physical stress is applied to an object, leading to a change in its shape. Physical forces like weight and gravity can change the shape of objects, depending on their composition. (DuBose 2023.)

4.1 Objectives

The primary objective is to develop a 3D mesh deformation shader system that is adaptable and easily integrated into a wide variety of projects within Unreal Engine 5, regardless of their nature. This system should be designed with flexibility and modularity in mind, allowing for seamless implementation in both game development and film production workflows.

Another objective is to ensure that the mesh deformation achieved by the shader exhibits realism, closely mimicking the physical properties and behaviors of different materials. The system will be designed to interpret and respond to comprehensive set of parameters that define the extent and nature of the deformation. Not only for the Deformable meshes, but also for the projectile or the other force that is causing the deformation when colliding. These parameters will allow users to fine-tune the deformation effects to achieve the desired results.

4.2 Environment

4.2.1 Unreal Engine

The project is developed using Unreal Engine version 5.2, using the First Person Template (Figure 11) to provide a starting point with essential first-person gameplay mechanics like movement, interaction and shooting a projectile, which will be used to apply the deformation. The shader is compatible also with older versions in Unreal Engine 4.

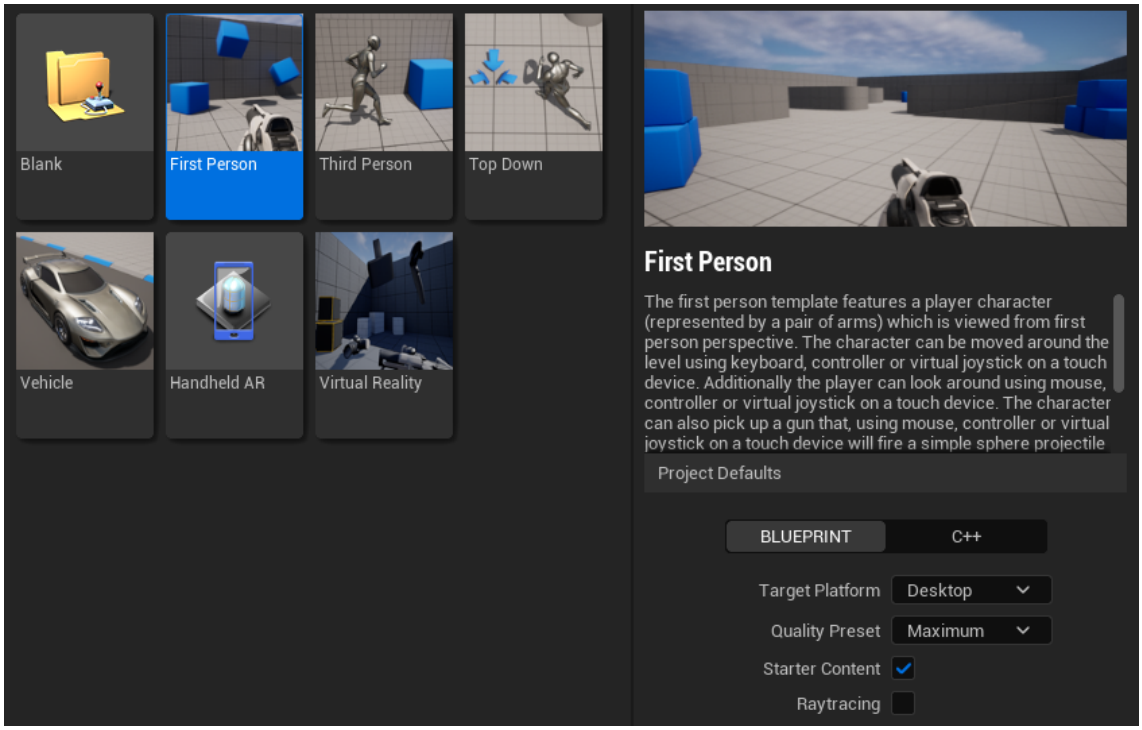

Figure 11. Template selection when creating new project in Unreal Engine.

4.2.2 Deformable Mesh

The example mesh to deform in this project is a 3D model of a Barrel, created in 3D-modeling software Blender (Figure 12). It is designed with a high vertex count, as discussed in the “3.3 3D Mesh” section, noting that the shader is a vertex-based solution.

Figure 12. 3D Mesh of the deformable mesh Barrel with applied textures.

The Barrel model (Figure 12) is primarily used to demonstrate how it can be deformed by the impact of projectiles. The model is chosen for its relatable and easily visualized shape, serving as a example to showcase the deformation.

As projectiles strike the Barrel, they produce a variety of deformation effects, from small dents to significant warping, depending on the amount of force, angle of impact and the parameters of the Actor Components, mentioned in next section.

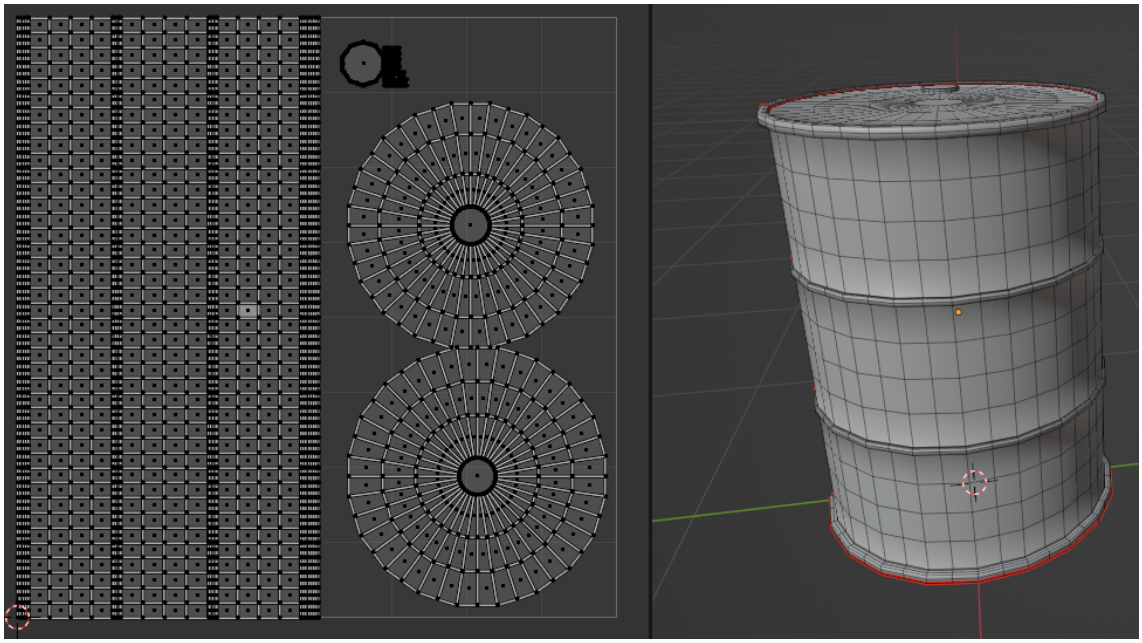

Figure 13. UV map and 3D Mesh of the Barrel.

A mesh with more vertices can be shaped in more detail, allowing for a broader range of deformation effects. This makes high vertex count meshes (Figure 13) better suited for models where detailed damage or alterations need to be applied. The higher the vertex-count is in the mesh, the more moldable the mesh becomes.

4.3 Actor Components in the project

4.3.1 Deformer Component

Deformer Component represents Actor Component that is attached to the projectile (Figure 14), causing deformation to the mesh targeted for alteration. Its primary function is to facilitate the deformation of other meshes upon impact, effectively simulating the physical interaction and resulting alteration to the objects like the barrel model.

When the projectile collides within the mesh, it obtains UV coordinates of the impact hit and notifies the “Deformable Mesh Component” about the collision. This notification triggers to initiate the deformation process.

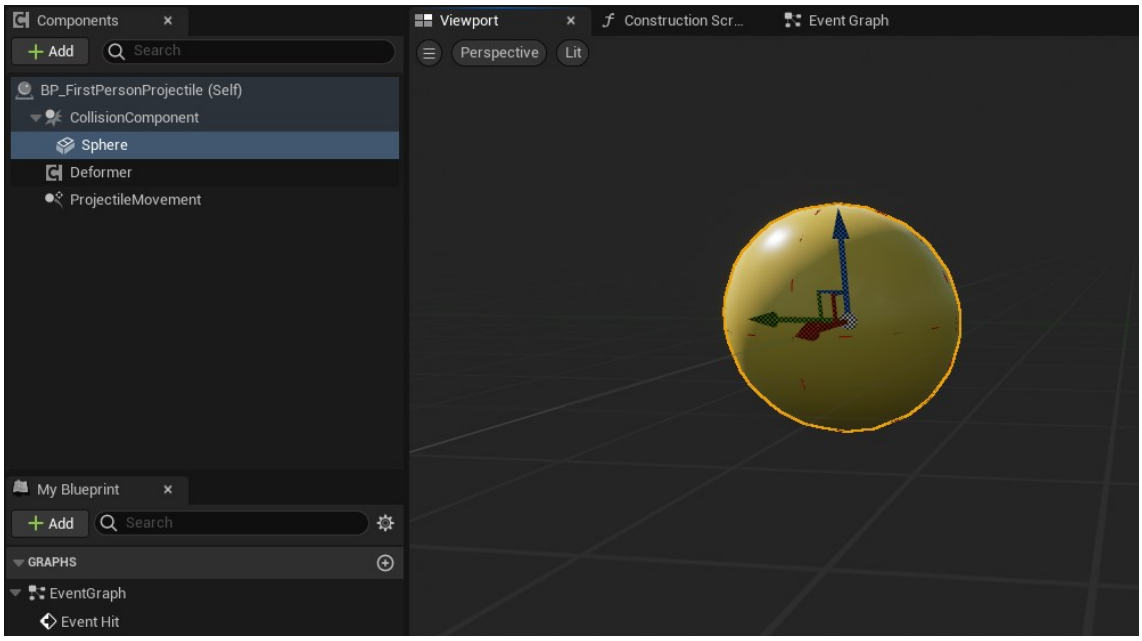

Figure 14. Deformer Component attached to the projectile.

4.3.2 Deformable Mesh Component

“Deformable Mesh Component” is Actor Component attached to any object in the game environment that is intended to be deformed (Figure 15). It contains multiple parameters which the user can tweak to alter the results to fit for the use case. In this case, there is a steel barrel where a certain minimum amount of velocity is required for the projectile to deform the mesh. Steel is also rather strong material so the “deforming multiplier” values are set up to make the deformation visible but moderate enough to remain realistic.

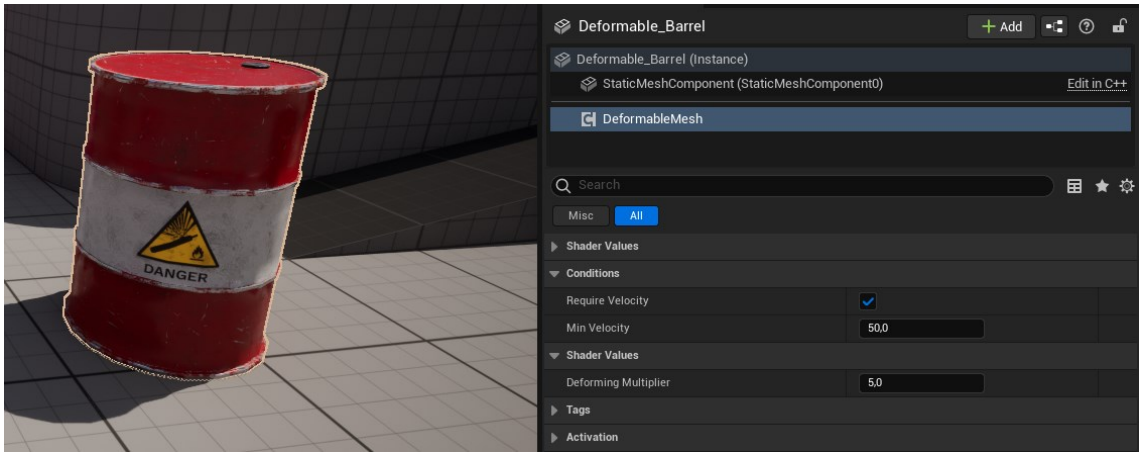

Figure 15. Barrel static mesh Actor with "Deformable Mesh Component" attached.

This Actor Component (Figure 15) manages the deformation process with tweakable parameters to control the deformation properties. It also controls the visual aspects like creating Material Instances for the object and directly influences its values in real time.

4.4 Capturing UV Coordinates

Capturing the UV Coordinate location to display the deformation happens when impacting with the “Deformer Component” onto the object which has the “Deformable Mesh Component”. If the projectile hits with enough force and is a valid object, the UV Hit location is captured, saved into a “Hit Result” variable, and then continued onto painting to the Render Target.

There is also alternative ways to capture the UV coordinates using Line Trace or the “Hitscan” based solution (Figure 16), which can be applied in instances where the impact is immediate, like when shooting a bullet from a gun. In such cases there is no need to use the “Deformer” component, instead Line Trace can be used to get the first object with a “Deformable Mesh Component” and apply the deformation to the hit location.

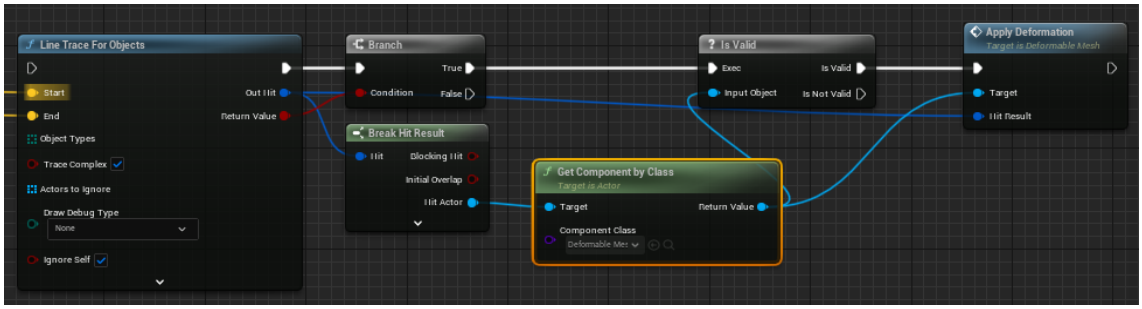

Figure 16. Using Line Trace to apply the deformation.

4.5 Painting to Render Target

After the collisions and parameter checks, when the object is ready to be deformed, either a new Render Target is created, or if the Component already has one, that is used. To paint onto the Render Target, a specific brush Material is used. It contains a custom image for painting. In this example, a default “glow” texture image, which is included in Unreal Engine starter content, is used to display the deformation onto the barrel mesh.

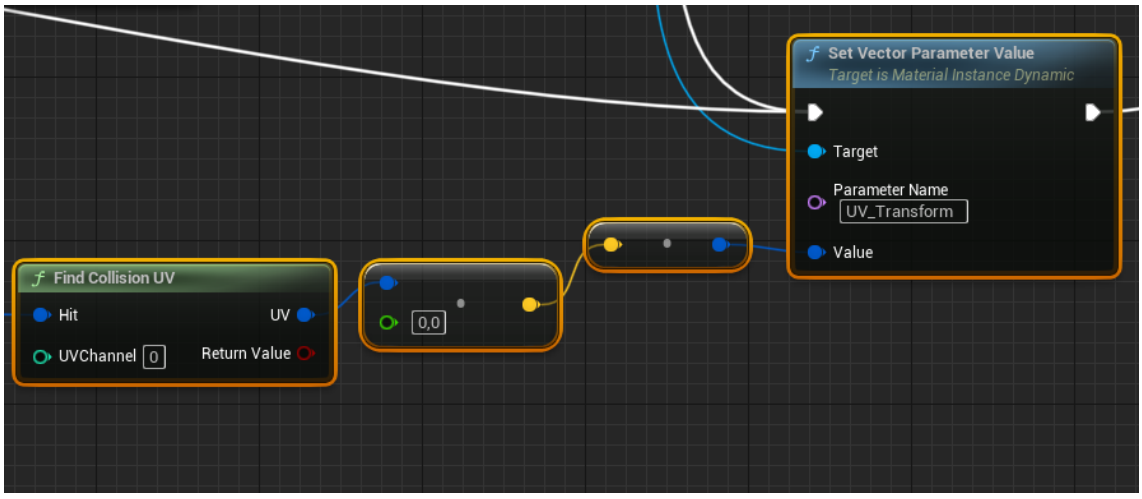

Figure 17. Using the UV Coordinate Location to center the brush Material image to the same location.

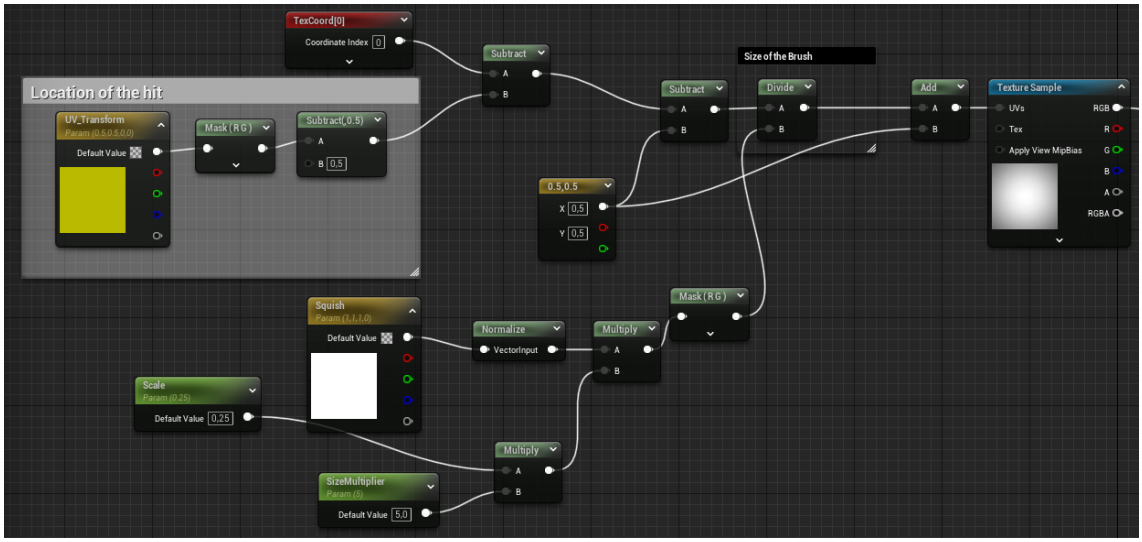

Figure 18. Brush Material.

The image (Figure 18) demonstrates the process of painting onto the Render Target. The UV location (UV_Transform) is set directly to the Hit Location, the scale of the image is based on the mesh size, and finally, the material is drawn onto the Render Target (Figure 19), which can then be used to mask the deformation effect (Figure 20).

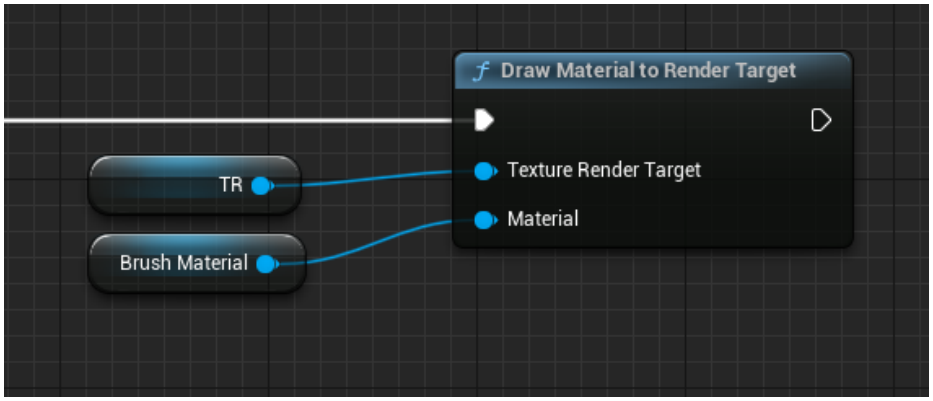

Figure 19. Drawing the brush Material onto the Render Target.

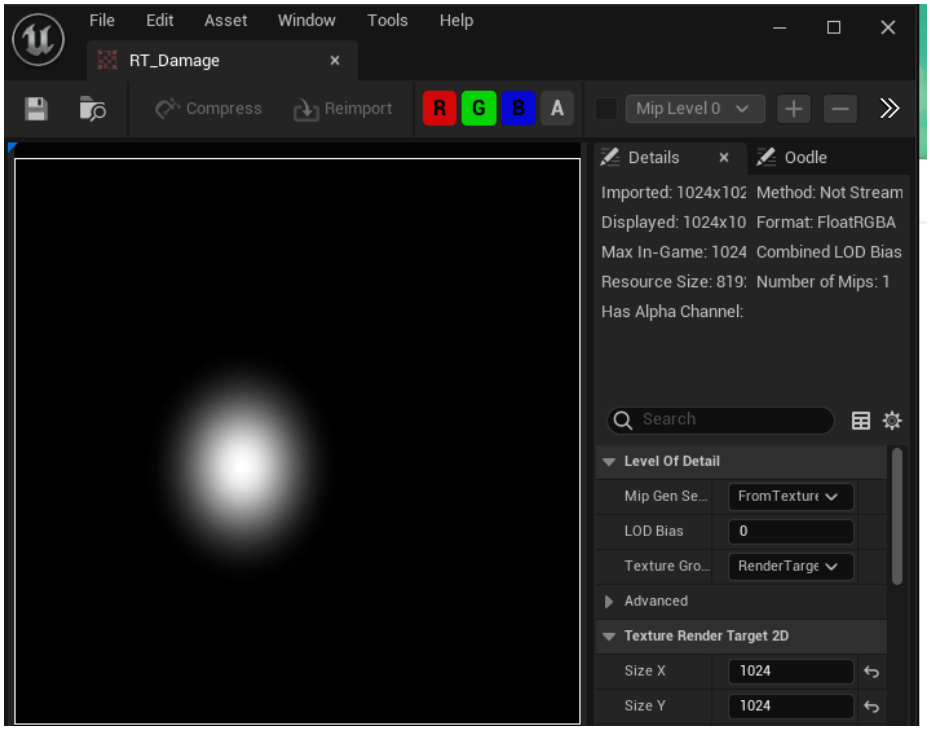

Figure 20. Render Target with the painted Brush image.

Now the wanted deformation is painted to the Render Target (Figure 20) and it can be used as a grayscale texture mask to apply the deformation effect onto the targeted mesh. Essentially, the brush Material is painted onto the collision hit location. Users can use any kind of grayscale texture brush to paint onto the Render Target.

4.6 Applying deformation

The final step involves on utilizing the painted Render Target, a deformation multiplier which is a floating point variable stored in “Deformable Mesh Component”, which then is sent to the Material Instance that is applied to the deformable mesh object.

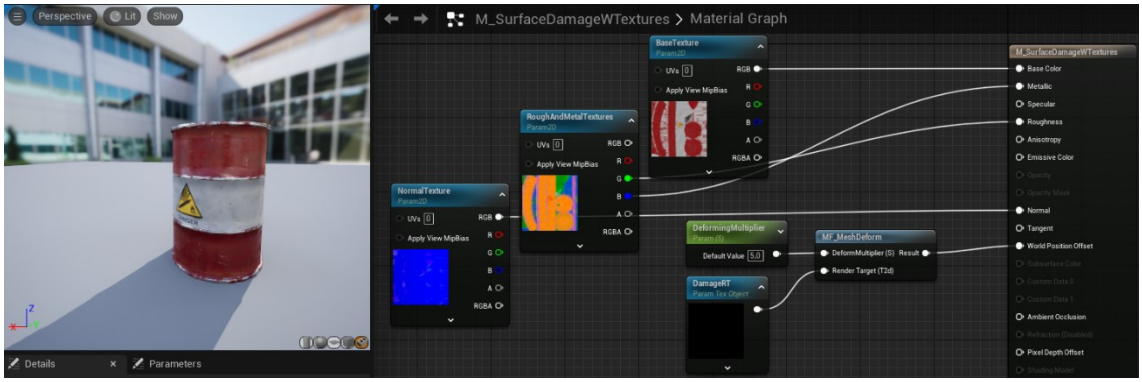

Figure 21. Material view of the Barrel.

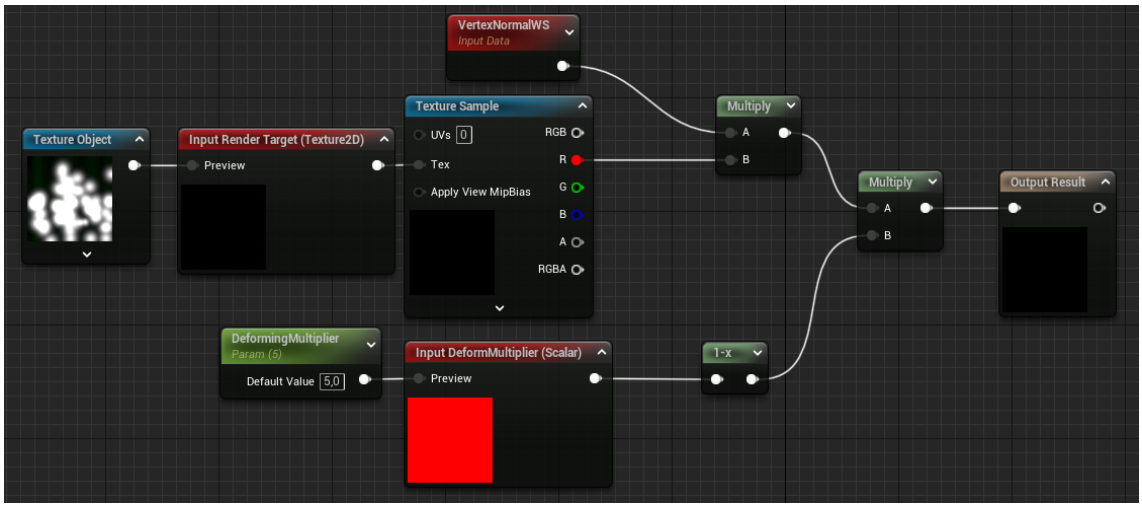

Figure 22. Material Function "MF_MeshDeform".

“MF_MeshDeform” (Figure 22) is the final part of the deformation process. This Material Function takes the deformation multiplier and the grayscale texture generated from the painted Render Target – where the white parts highlight the areas of deformation (Figure 20) and utilizes it within the Unreal Engine’s Material system.

By connecting this to the “World Position Offset” node (Figure 21), the engine is instructed to adjust the vertices of the mesh in real-time, based on the painted data in the Render Target. As it was mentioned in section “3.6 Texture Masking”, the darker areas indicate regions of minimal or no deformation, where as the white parts indicate where the deformation should occur. This shifts the vertices

of the mesh by the UV map creating the visual effect of dents, bulges and warps exactly where the impact was registered.

4.7 Results

In this section, various deformations are presented through a display setting that includes images from different views, the specific brush used in the deformation process and an image of the Render Target. The “deformation multiplier” variable to enhance the power for the deformation effect is same for all the images displayed.

These images collectively showcase the versatility and detail of what the deformation system can achieve. By varying brushes and parameters, different deformation effects can be created depending on the projectile or other forces used to cause the deformation, by selecting a brush that fits for the use case, offering for more dynamic and interactive game worlds.

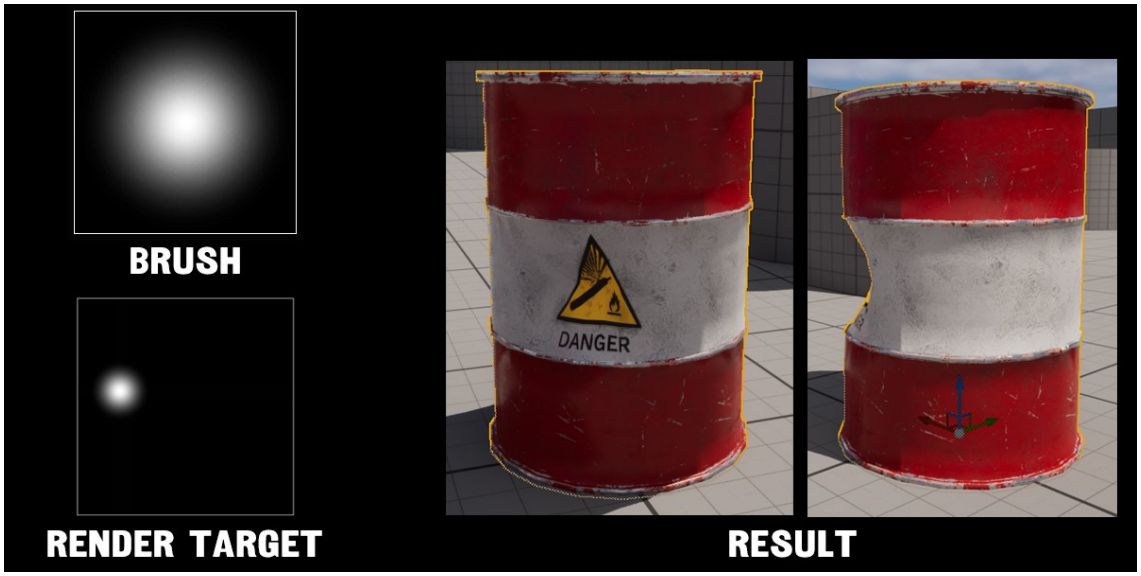

Figure 23. The Barrel mesh with the default brush that has been used as an example in this thesis.

Figure 24. Square brush texture with hard edges.

Figure 25. Multiple projectile collisions with default brush.

Figure 26. Multiple projectile collisions on top of the Barrel mesh with default brush.

For additional variation and to further illustrate the deformation effect on a different mesh, the following includes a typical 3D model also frequently used in variety of video games, provided with UV mapping and wireframe image of the 3D Mesh. (Figure 27)

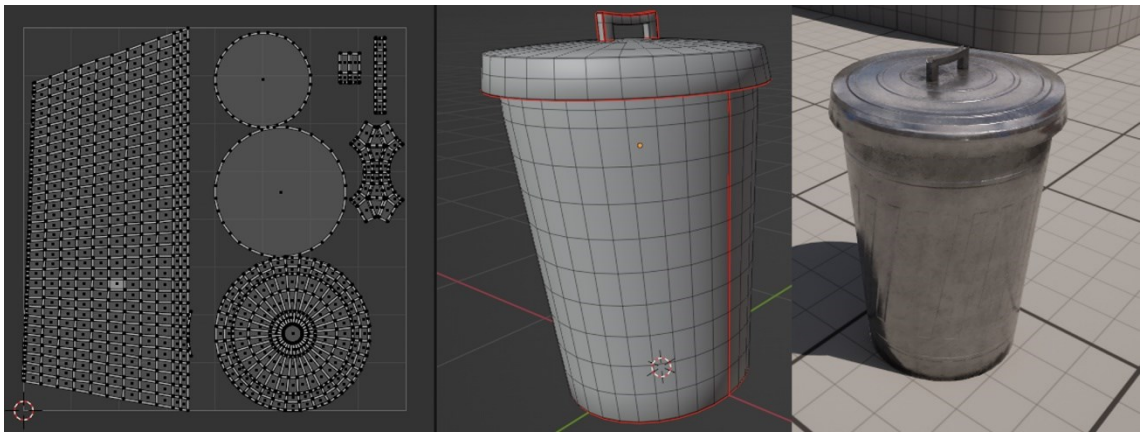

Figure 27. UV map, wireframe and 3D Mesh of high-vertex count trashcan model.

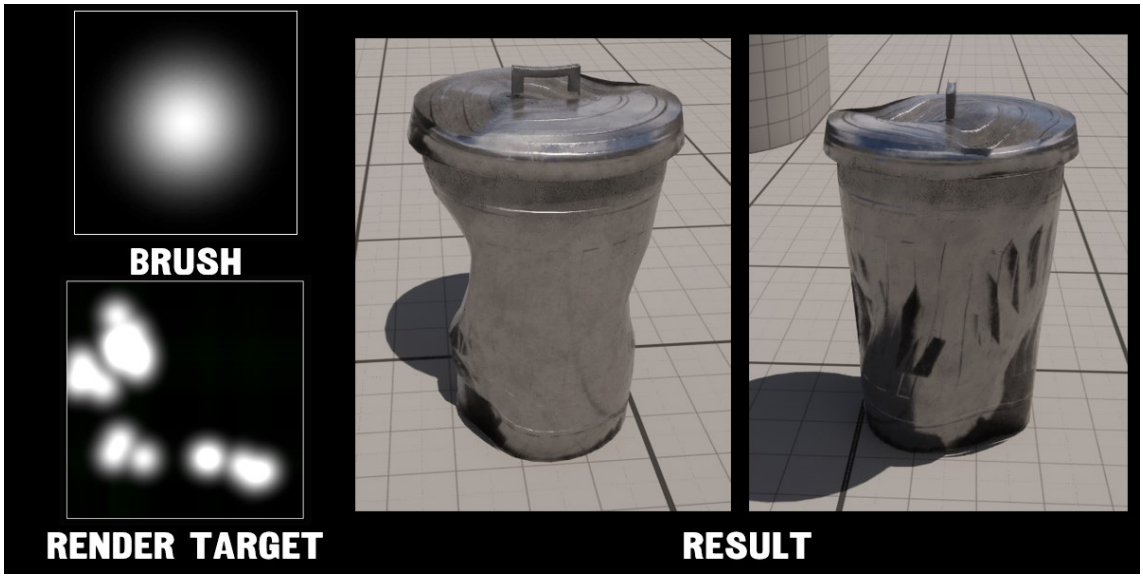

Figure 28. Multiple projectile collisions to trashcan mesh.

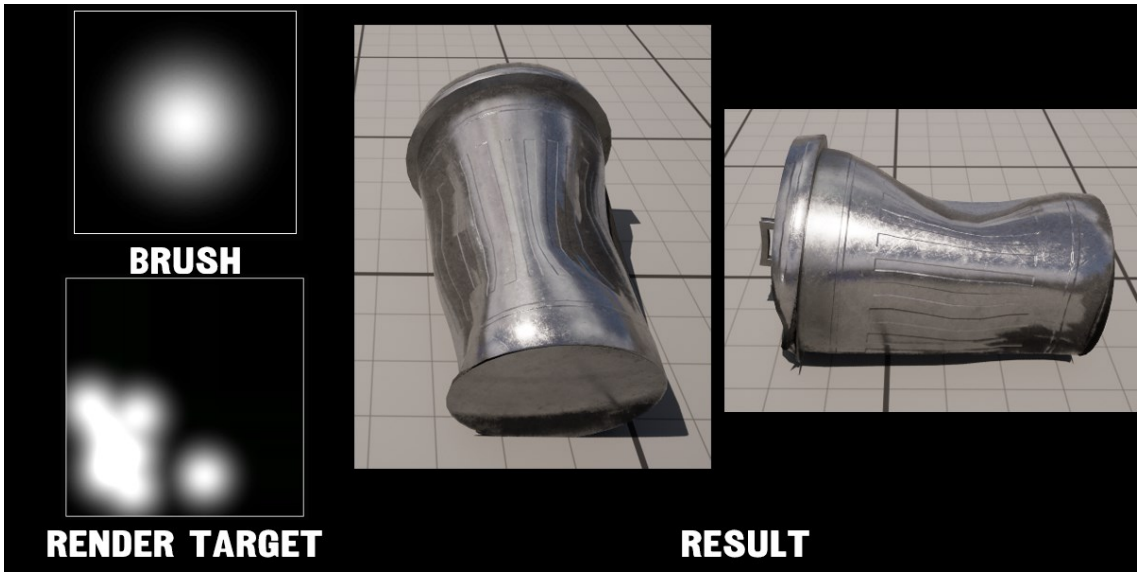

Figure 29. Multiple projectile collisions to rotated trashcan mesh.

5 Discussion

Throughout the development work, the shader was designed to be universally applicable, allowing it to work seamlessly with any static mesh that is wanted to be deformed. Requirement for the meshes are to have single, well-structured UV map without overlapping islands to ensure the deformations are accurately applied, and also to be vertex-heavy.

For performance considerations, the user needs to be cautious when applying too many objects with deformation components. Creating an excessive number of high-quality Render Targets can lead to significant performance bottlenecks. This is true in scenarios where memory resources are finite, and computational overhead can quickly escalate, leading to decreased frame rates and potentially impacting the overall smoothness and responsiveness of the game or application.

For possible further development of the shader system, the client might consider further developing the shader system by exploring into integrating with Unreal Engine 5's Nanite system. The benefits of using Nanite system is that frame budgets are no longer constrained by polycounts, draw calls and mesh memory usage, and the Level of Detail (LOD) is automatically handled (Unreal Engine Documentation 2024h.). That experimentation could potentially negate the side effects of using a high vertex-count 3D models. Other improvements for the system could be applying scratches and dirt upon deformation, adding even more realism and visual fidelity to the deformations. Also, the brushes to paint onto the Render Target could be experimented with procedurally generated brush that would add variation to the deformations.

Lastly an advancement would involve extending the shader's functionality to work with skeletal meshes, as for now it only is applied to static meshes. Developing a solution that allows for the deformation effects to be applied to animated characters and objects would significantly increase the shader's utility, applying an extra layer of immersion and realism to skeletal meshes.

6 Conclusion

The primary objective of this thesis was to develop a realistic, user-friendly deformation shader within Unreal Engine 5. This system introduces realistic and customizable deformation effects to 3D static meshes enhancing the visual fidelity in video game and film projects. Furthermore, it ensures ease of use for the developer, allowing them to utilize this tool seamlessly across different projects within Unreal Engine.

The project was successful, resulting in a working Render Target and vertex-based deformation shader that the developer can easily alter and apply through various projects with any static mesh implementing “Deformable Mesh Component”. Another objective was to develop a parameter-driven reusable component to tweak the values for the deformation amount, stiffness and durability of the various meshes so the developer can adjust values inbetween different object materials like metals, wood or plastic.

However, there are essential aspects the user needs to be aware of, for example that the deformations only affect the visual outlook of the mesh, it does not alter the collision bounds. This distinction is crucial as it can lead to scenarios where the visual deformation and the physical collision boundaries of an object do not align, resulting in unwanted outcomes.

The shader represents a great addition to games and interactive applications, offering developers the tool to add more immersive and visually compelling experiences for both video game and film industries. Examples for different kind of applications or use cases for the deformation shader would be crashing cars, poles, traffic signs, barrels, or walls that could be deformed when getting impacted by a force.

The success of this project not only completed the predefined objectives, but can also be used to create new Render Target based systems as a framework – like cleaning, polishing, environment interaction or painting systems by using similar principles like in the thesis.

References

3D-Ace, (2024). Does polygon count matter in 3D modeling for game assets? Available at: https://3d-ace.com/blog/polygon-count-in-3d-modeling-for-game-assets/ [Accessed 20.4.2024].

Awati, R. (2024). 3D mesh. Available at: https://www.techtarget.com/whatis/definition/3D-mesh [Accessed 15.4.2024].

Brightman, J. (2012). Epic Games launches Unreal Government Network for serious games applications. Available at: https://www.gamesindustry.biz/epic-games-launches-unreal-government-network-for-serious-games-applications [Accessed 16.3.2024].

Dean, I. (2023). Is Unreal Engine the future of filmmaking? Available at: https://www.creativebloq.com/features/use-unreal-engine-for-filmmaking [Accessed 16.4.2024].

Devereaux, J. (2023). Power Shift: How the Evolution of Computer Hardware Transformed the Gaming Industry. Available at: https://www.pczone.co.uk/power-shift-how-the-evolution-of-computer-hardware-transformed-the-gaming-industry/ [Accessed 12.4.2024].

Doucet, L., & Pecorella, A. (2021). Game engines on Steam: The definitive breakdown. Available at: https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown [Accessed 15.3.2024].

DuBose, A. (2023). Deformation. Available at: https://study.com/academy/lesson/what-is-deformation-definition-types-process.html [Accessed 30.4.2024].

Eugene, P. (2024). What is a render target. Available at: https://www.easytechjunkie.com/what-is-a-render-target.html [Accessed 25.3.2024].

Farris, J. (2020). Forging new paths for filmmakers on “The Mandalorian”. Available at: https://www.unrealengine.com/en-US/blog/forging-new-paths-for-filmmakers-on-the-mandalorian [Accessed 16.3.2024].

FasterCapital (2024). Rise of Gaming. Available at: https://fastercapital.com/startup-topic/Rise-of-Gaming.html [Accessed 15.4.2024].

Ferrier, A. (1999.) Real-Time Soft-Object Animation Using Free-Form Deformation. Available at: https://www.gamedeveloper.com/programming/real-time-soft-object-animation-using-free-form-deformation [Accessed 16.3.2024].

Foundry Learn (2024). UV Unwrapping. Available at: https://learn.foundry.com/nuke/content/comp_environment/modelbuilder/uv_unwrapping.html [Accessed 25.4.2024].

Kodeco (2018). Unreal Engine 4 Tutorial: Painting With Render Targets. Available at: https://www.kodeco.com/5246-unreal-engine-4-tutorial-painting-with-render-targets [Accessed 25.4.2024].

Merzlikin, A. (2022). Deforming a mesh in Unity. Available at: https://blog.logrocket.com/deforming-mesh-unity/ [Accessed 25.3.2024].

Pecorella, A. (2021a). Engine Launches Each Year. Available at: https://infogram.com/1d560b7e-21a1-437a-91f4-198309bf3e25 [Accessed 15.3.2024].

Pecorella, A. (2021b). Launch Prices by Engine. Available at: https://infogram.com/ee89ac9f-84eb-403c-ad7e-dbe527b51dd9 [Accessed 15.3.2024].

Phuc, D. (2023). Is Unreal Engine Good For Indie Games? The Advantages and Challenges. Available at: https://animost.com/ideas-inspirations/is-unreal-engine-good-for-indie-games/ [Accessed 5.4.2024].

Pohl, B. (2017). HALON evolves previs with Unreal Engine. Available at: https://www.unrealengine.com/en-US/developer-interviews/halon-evolves-previs-with-unreal-engine [Accessed 21.3.2024].

Ponnam, K. (2023). Creating Custom Shaders in Flutter: A Step-by-Step Guide. Available at: https://medium.com/flutter-community/creating-custom-shaders-in-flutter-a-step-by-step-guide-49ec86bec20d [Accessed 1.4.2024].

Raturi, G. (2023). How is Unreal Engine Transforming the Future of Gaming? Available at: https://medium.com/codex/how-is-unreal-engine-transforming-the-future-of-gaming-c0ff9ad5d9fe [Accessed 17.3.2024].

Robinson, M. (n.d.). UV Mapping explained. Available at: https://myndworkshop.com/resources/uv-mapping-explained [Accessed 22.3.2024].

Schouzib, I. (2023). Shaders in video games: Types and Techniques. Available at: https://invogames.com/blog/what-are-shaders-in-video-games/ [Accessed 17.4.2024].

Sweeney, T. (2005). GPU Gems 2 – Foreword. Available at: https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_frontmatter.html [Accessed 15.3.2024].

Sweeney, T. (2018). Announcing the Epic Games store. Available at: https://www.unrealengine.com/en-US/blog/announcing-the-epic-games-store [Accessed 15.3.2024].

Thomas R. & Zhang W. (2023.) Real-time fracturing in video games https://link.springer.com/article/10.1007/s11042-022-13049-x

Unreal Engine Documentation, (2024a). Material Editor Reference. Available at: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/Editor/ [Accessed 30.4.2024].

Unreal Engine Documentation, (2024b). Using Texture Masks. Available at: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/HowTo/Masking/ [Accessed 5.4.2024].

Unreal Engine Documentation, (2024c). Components. Available at: https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Actors/Components/ [Accessed 28.4.2024].

Unreal Engine Documentation, (2024d). Static Meshes. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/static-meshes?application_version=5.0 [Accessed 28.4.2024].

Unreal Engine Documentation, (2024e). Skeletal Meshes. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/skeletal-mesh-assets-in-unreal-engine?application_version=5.2 [Accessed 28.4.2024].

Unreal Engine Documentation, (2024f). Working with UV Channels. Available at: https://docs.unrealengine.com/4.26/en-

US/WorkingWithContent/Types/StaticMeshes/HowTo/UVChannels/ [Accessed 15.4.2024].

Unreal Engine Documentation, (2024g). Actor Component. Available at: https://docs.unrealengine.com/5.0/en-US/PythonAPI/class/ActorComponent.html [Accessed 15.4.2024].

Unreal Engine Documentation, (2024h). Nanite Virtualized Geometry. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine?application_version=5.0 [Accessed 25.4.2024].

VFXDoc (2024). Shaders: Technical Overview. Available at: https://vfxdoc.readthedocs.io/en/latest/shaders/overview/ [Accessed 22.4.2024].

Yu X., Dai P., Li W., Ma L., Liu Z., & Qi, X. (2023). Texture Generation on 3D Meshes with Point-UV Diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4206-4216. Available at: https://openaccess.thecvf.com/content/ICCV2023/html/Yu_Texture_Generation_on_3D_Meshes_with_Point-UV_Diffusion_ICCV_2023_paper.html [Accessed 7.4.2024].