Tampereen ammattikorkeakoulu

Creating Foliage for Video Games

Aleksi Karppinen

BACHELOR'S THESIS
October 2023

Business Information Systems
Game Production

ABSTRACT

Tampereen Ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game Production

KARPPINEN, ALEKSI:
Creating Foliage for Video Games

Bachelor's thesis 30 pages
October 2023

The objective of this thesis was to focus on different ways of creating foliage for video games using Blender as the modeling tool and Unreal Engine 5 as the game engine. Additionally, the thesis covers the material creation process for Blender and Unreal Engine, since it is a big part of foliage creation.

The methods used in this thesis range from basic alpha-plane foliage creation to more advanced approaches, such as Nanite. Nanite is a feature in Unreal Engine that enables the creation of highly detailed foliage with reduced resource consumption compared to fully modeled foliage. Using well-optimized and resource-efficient techniques is crucial in situations where games need to deliver smooth performance even on lower-end devices.

This thesis shows examples for the creation process of grass, flowers, and tree leaves, with suggestions on how to improve them efficiently without increasing polygon counts significantly. Moreover, it includes visual references from Fortnite by Epic Games, developed in Unreal Engine, with a detailed breakdown of possible foliage creation techniques used.

While there may not be a definitive best approach, this thesis provides a deeper insight into the possibilities available. Comparing the polygon counts of these techniques provides a clearer direction for deciding how to approach foliage creation.

Key words: game foliage, game art, Blender, Unreal Engine

CONTENTS

ABBREVIATIONS AND TERMS

AI	Artificial intelligence
Alpha texture	Includes transparency, allowing an image or an object to be partially invisible
Asset	Digital resource like 3D model, texture, or sound file used to build the game's world
Atlas	Texture atlas, an image containing smaller textures which reduces the amount of texture lookups
Blender	Free 3D modeling software
Game engine	Software that provides essential tools and functionalities for game development
GPU	Graphics processing unit, renders images for the user
LOD	Level of detail
Megascans	Collection of high-quality scanned materials and assets
Mesh	Structural build of a 3D model formed by polygons
Nanite	Unreal Engine’s technology which efficiently handles complex meshes by adjusting the polycount based on distance
Node	Executes various operations on the material, altering its appearance when applied to object
Photopea	Free online photo editor

Plane	Flat surface
Polycount	Number of polygons needed for the 3D model
Polygon	Flat, closed shape with a three or more straight sides
Real-time rendering	Continuously generating and displaying graphics in interactive applications, like in video games
Rendering	Generating an image which requires calculating how lighting, shadows and other visual effects affect the object
Texture	Digital images or maps applied to 3D models to give them surface details, multiple textures are used to create a material
Topology	Defines how vertices, edges, and faces are arranged and distributed within a 3D object
Triangle (tri)	Three-sided polygon, game engines tend to split polygons into triangles
Unreal Engine	Game engine developed by Epic Games
Vertex	Each angular point of a polygon

1 INTRODUCTION

The rapid growth of mobile games and the soaring popularity of multiplayer online games are reasons why game developers must put more focus on the poly-counts. Ensuring that the games run smoothly on lower-end devices is important especially in multiplayer games so that there is no disadvantage for others. Foliage is an important part of many games nowadays and it can easily consume a lot of resources from the game engine. Game engine is the software that provides tools for game development and well optimized engines can handle complex objects better.

Almost every game uses some sort of vegetation to bring the game worlds alive. “Vegetation most definitely plays a large role in helping establish mood and atmosphere for any given area.” (Karmaker 2016). Plants are a great way to bring some color and life to the area or just simply fill up the space to make it look more vivid.

Creating vegetation for 3D games can be challenging since artists does not want to make its polycounts too high but still want to make it look good. If every leaf is modeled individually, the polygon counts can rise to millions which is not very optimal for video games. Polygons are flat, closed shapes that form the mesh, which defines the overall structure and shape of an object.

Popular way of making foliage is to use alpha textures which appear partly invisible and allow to create more interesting shapes without using a lot of polygons. Alpha planes can be used in many ways to create foliage, but it always requires a bit more time and planning to make them appealing for the player. Flat planes can disappear wholly from certain angles which breaks the illusion of fully modeled foliage.

The thesis will be going through different general ways of foliage creation using free tools ensuring that the instructions are accessible even for beginners. The intention behind the thesis is to show and get a deeper understanding into ways which can be used to create foliage. Blender is a very capable 3D modeling tool

and Unreal Engine 5 is Epic Games' newest version of their popular game engine. The same methods can easily be adapted to other modeling tools and game engines. Some of the features, like Nanite, are exclusive to Unreal Engine 5 but similar technologies could be found in other engines as well.

2 CREATING MATERIALS

2.1 Fundamentals of materials

Materials are used to paint the objects and bring them to life through enhanced visual appeal. They define how the objects look in terms of color, shininess, opacity, and many other properties. In technical terms, materials are used to calculate how the light interacts with the surface when it hits it (Materials n.d.).

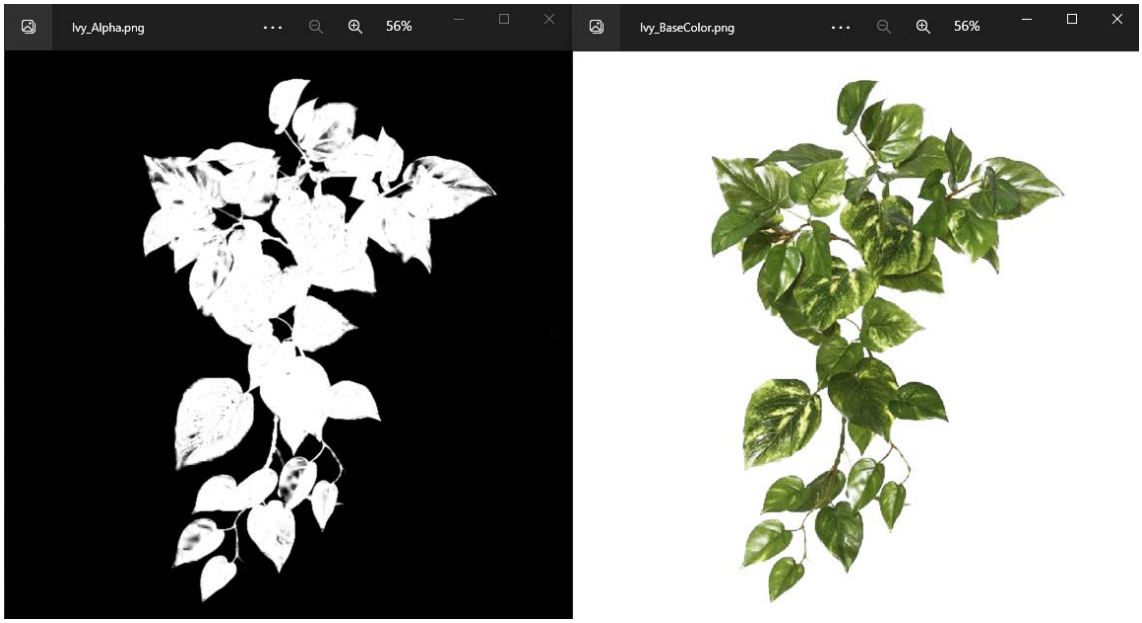
Game engines have material editors, also known as shader editors, which allow texture images to be used to create new materials. Materials consist of multiple values or texture images, for example base color, roughness, and normal maps. Textures are images or maps that give the surface details for the object and define how the material looks and reacts to the environment.

2.2 Creating textures

Photopea is a great and free online tool to create the textures that will be used for masked materials. Two textures are required for the masked material to work, one with base color and another one with alpha mask. Other textures can also be used but commonly these values, like roughness, can be adjusted in the game engine or modeling tool itself. Creating a texture map, for example for roughness, comes in handy when there is variability in the values across the material. If the whole material is using a single value, there is no need to create a map but adjusting the value in the editor works just fine.

Generating alpha masks is a quick and easy process within the photo editor, which in this case is Photopea. Applying a color overlay from the layer styles and then adjusting the curves in the opposite direction, turning the white background into black, creates the required mask (Zavhorodnia 2022). Since there is no transparency, these textures can be exported in multiple different formats, for example in PNG, TGA, or JPG. The white parts of the mask will be rendered, and black

parts will stay transparent. Picture 1 shows the finished mask and base color texture used for an ivy.

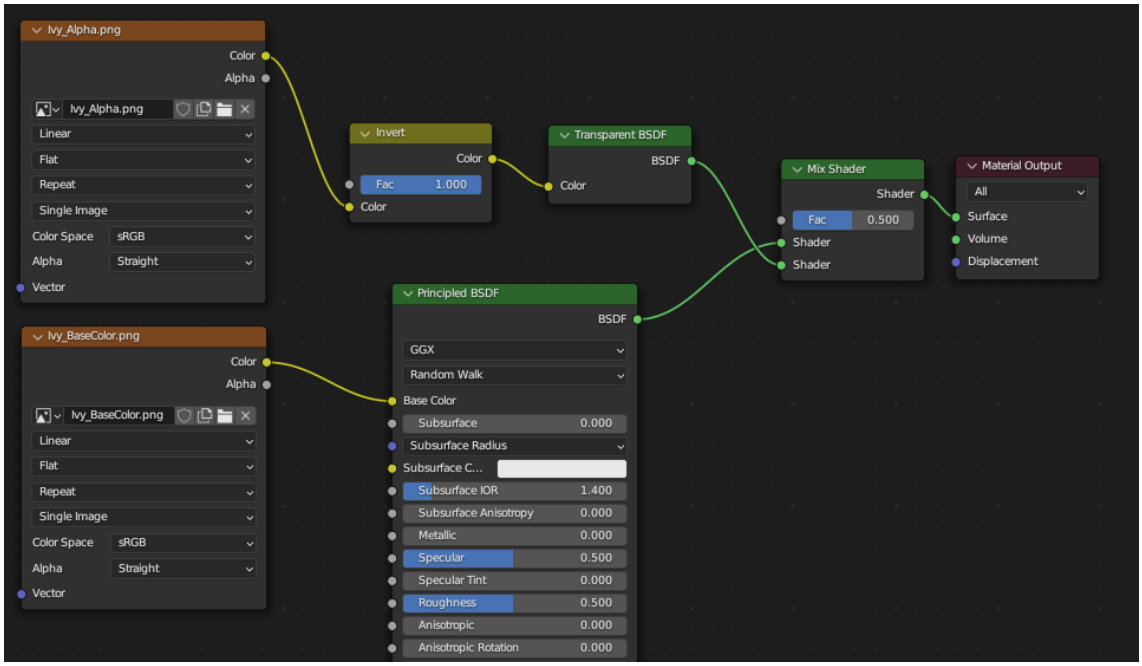


PICTURE 1. Alpha and base color textures. Ivy hanging image: Brett Croft 2022 (CC BY-NC 4.0).

2.3 Alpha materials in Blender

Using alpha textures in Blender can be helpful when modeling plants and environments. It helps to have a clear idea of how everything is going to look in the finished game without needing to go back and forth the game engine and the modeling tool. Blender offers multiple different blend modes for alpha textures, and they all have slightly different looks.

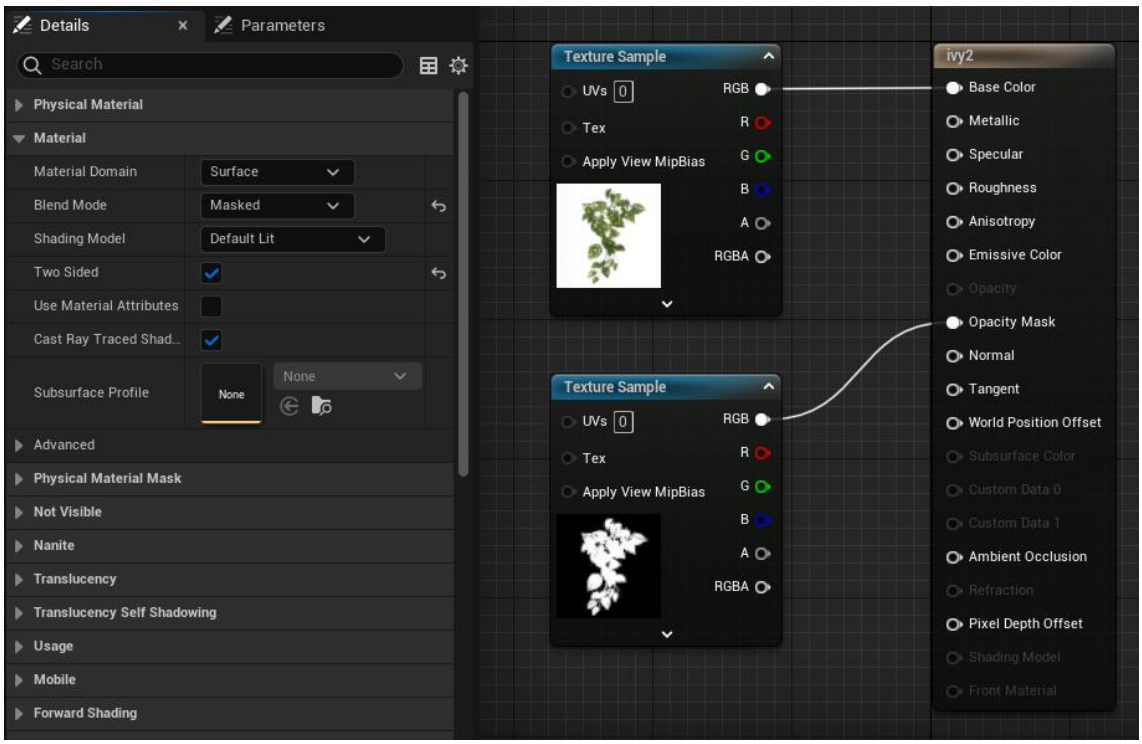
“Alpha clip is the cheapest to use in terms of performance, but it works from a clip value. This means that we can't have partially transparent pixels.” (Selin n.d.). Alpha clip mode is very efficient, and it is very close to how the textures are going to look in Unreal Engine. In Blender the default blend mode is set to opaque, and it must be changed from material properties for the transparency of the material to work. The clip threshold must be adjusted to be over 0.5 for the background to disappear. Material editors use nodes to execute various operations on the material, altering its appearance when applied to the object (Blender n.d.). All the necessary nodes for alpha clip blend mode can be seen in picture 2.



PICTURE 2. Blender nodes used for alpha clip blend mode.

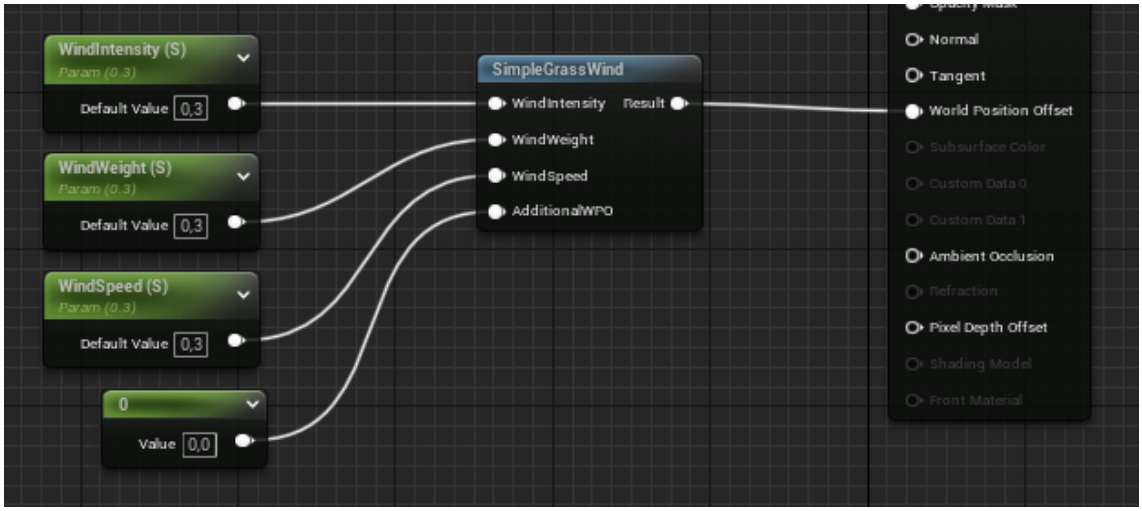
2.4 Alpha materials in Unreal Engine

Textures can be imported to Unreal Engine by simply dragging them into the content browser. The simplest way of making partly invisible material in Unreal Engine is using images for base color and opacity mask. Blend mode must be set to masked for the opacity mask to work. Black parts in the mask are not being rendered, leading to them not showing reflections or casting shadows, which results in performance savings on the graphics processing unit (GPU) (Material blend modes in Unreal Engine n.d.). GPU is responsible for rendering images for the user and its performance is impacted by the complexity of the assets. Enabling two-sided material can be useful if the plants are going to be visible from both sides. Picture 3 shows all the necessary nodes and options needed for alpha materials to work in Unreal Engine.



PICTURE 3. Material nodes in Unreal Engine 5.

Unreal Engine has an easy-to-use SimpleGrassWind function which gives a very basic and non-directional movement for foliage (World position offset material functions n.d.). It has inputs for intensity, weight, and speed of the wind with an additional world position offset setting, which all can be adjusted separately to create as harsh wind as needed. The SimpleGrassWind function needs parameters to give it some values. Scalar parameters are used in picture 4 to give a small movement for the foliage. Adding a bit of movement for the plants, especially outside, helps immersing players with the world and makes it a bit more realistic.



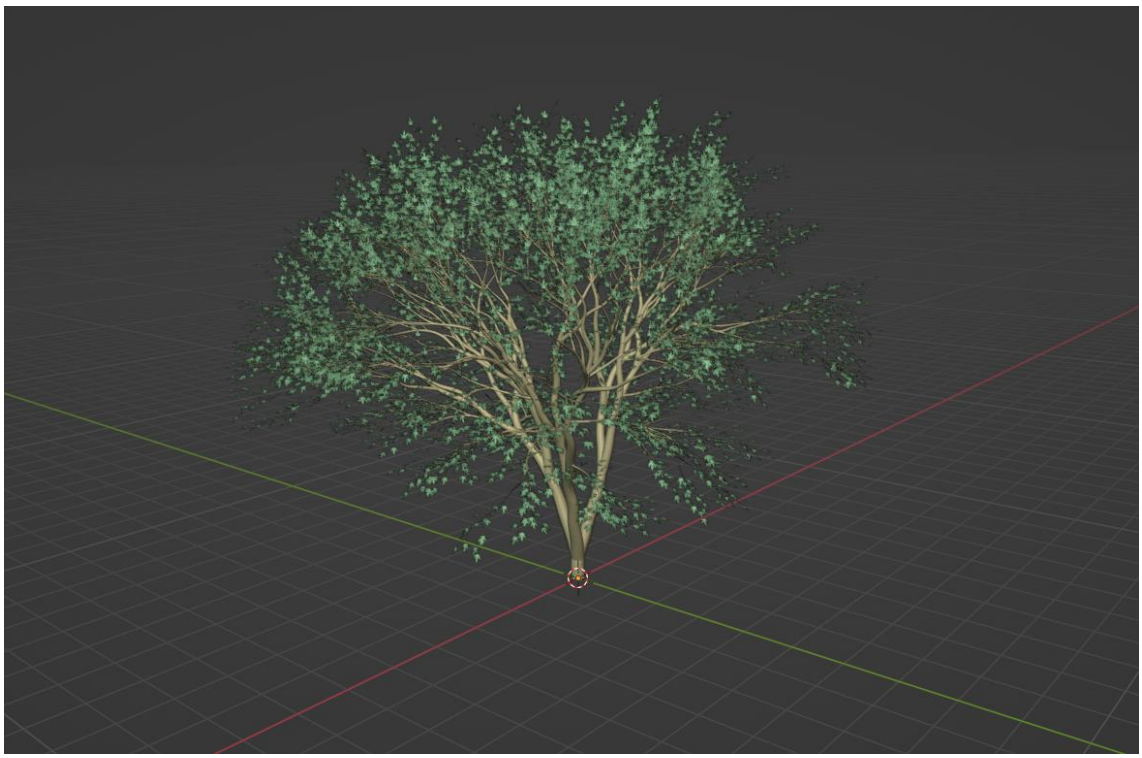
PICTURE 4. SimpleGrassWind function and the required inputs.

3 FOLIAGE CREATION TECHNIQUES

3.1 Detailed geometry

Blender offers a free add-on called Sapling Tree Gen which allows users to generate custom trees with different parameters. This method offers a fast way to create stunning and realistic trees but since these trees are formed with hundreds of thousands of polygons, they are very inefficient for game purposes. Proper amount of polygons for a PC game prop should be around 1 000 to 20 000 (Maxwell 2019).

The tree in picture 5 would be perfectly detailed for pre-rendered images and videos but it would be too much for games which are rendered in real time. Real-time rendering needs to be a lot faster than pre-renders or else it would cause lagging and interfere with the gaming experience. Using too complex meshes can cause lagging, and framerate drops since the game engine or the players' hardware cannot handle rendering so many polygons so quickly. Using just a ten of those trees would use over two million triangles which is more than some game levels even use. Game engines split every polygon into triangles, which are formed from three vertices and one face. It is better to use more geometry for bigger things, like buildings, than use the whole polygon limit for highly detailed leaves that no one even looks that close to notice.



PICTURE 5. Tree created using the Sapling Tree Gen. Leaf triangles: 213 600.

3.2 Nanite

Unreal Engine 5 supports a new virtualized geometry system called Nanite. Nanite works with automatic levels of detail (LODs), changing the topology and details of the mesh based on how far away it is. Topology of the object defines how vertices, edges, and faces are distributed within the object. Basically, Nanite automates the process of doing multiple different variants of the assets with different LODs. Assets are resources used for game development, including 3D models, textures, and sound elements.

Enabling Nanite allows using high poly meshes in games, but it is important to remember that it only reduces the polycount when the object is far away. From the close range the 200 000 polygon trees are still going to be very detailed and take a lot of resources. Nanite can be turned on for static objects from the asset settings (Maxwell 2021).

Preserve area was designed especially for foliage meshes that usually disappear from further away. This scales each leaf up instead of simplifying the mesh into

triangles and quads. (Nanite virtualized geometry in Unreal Engine n.d.) Over-simplification of the meshes tends to lead to them disappearing wholly, which happened to the leaves of the tree in middle in picture 6.

PICTURE 6. Left is without Nanite, middle one is with Nanite and right one is with Nanite and preserve area on. Tree is created using Sapling Tree Gen.

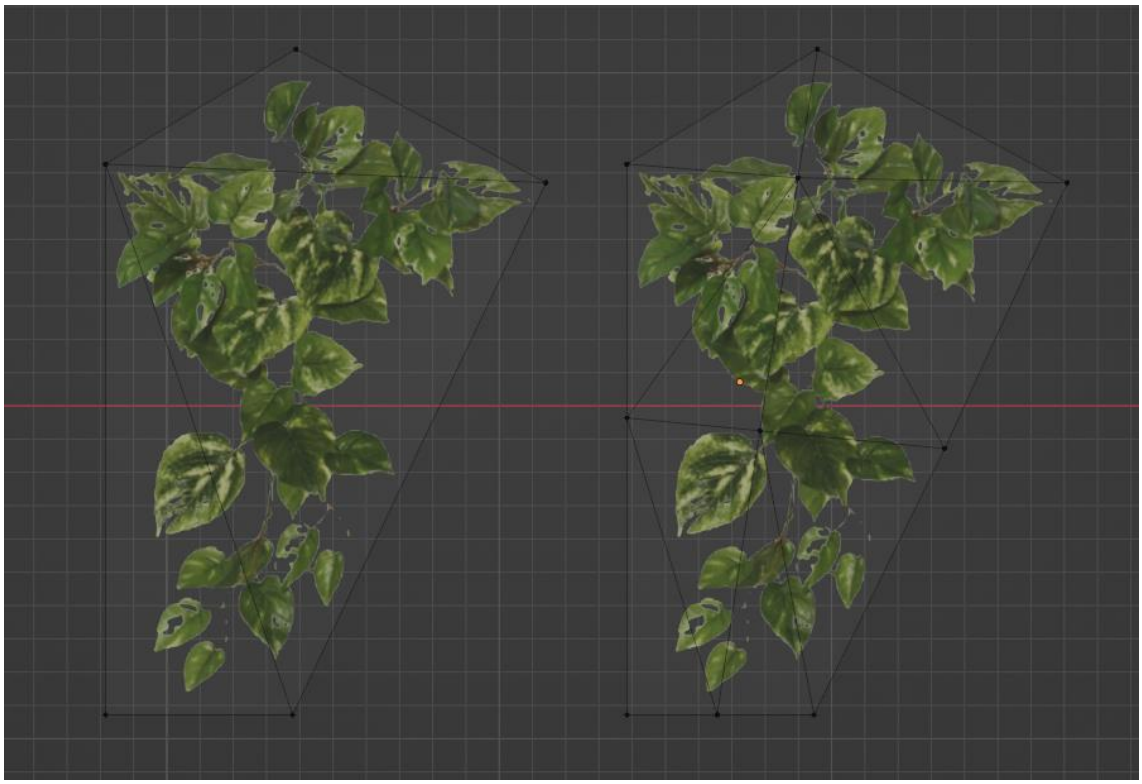
3.3 Alpha masked materials

A common and resource-efficient approach for adding foliage to video games is to use planes, which are flat surfaces, and then applying foliage textures to them using alpha masked materials. Creation process of masked materials was gone through in section 2.2. This way the polycount does not rise too high and game engines can render the object faster. Rendering objects requires the engine to calculate how lighting, shadows, and other visual effects affect it and it is a lot faster for less complex objects.

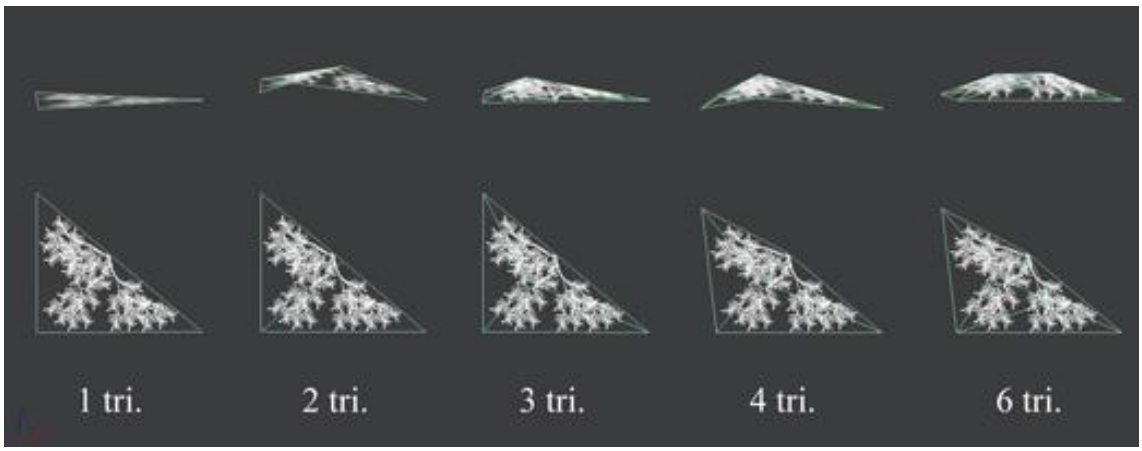
Using simple planes does not necessarily mean using just flat surfaces but there are ways to make them more interesting and appear to have more geometry than they really do. Using a knife tool in Blender to cut the planar mesh containing a plant into smaller pieces enables slight modifications to its geometry. Splitting mesh into smaller pieces does not increase polygon count significantly but makes it easier to form more interesting shapes. Moving some vertices in the 3D space

and adding a little rotation for the mesh prevents it disappearing wholly when looking at it from the sides.

Since the game engine is going to split polygons into triangles, working with triangles instead of polygons in the modeling software can be helpful. Moving the vertices have less impact on the structure allowing fine tuning it without making very noticeable harsh edges. In Blender, there is a feature to triangulate mesh when opening the face-menu in edit mode. In picture 7 the mesh is split in two different ways showing how a few more triangles can give a much more to work with. Picture 8 shows differently split planes and side views for an easy comparison.

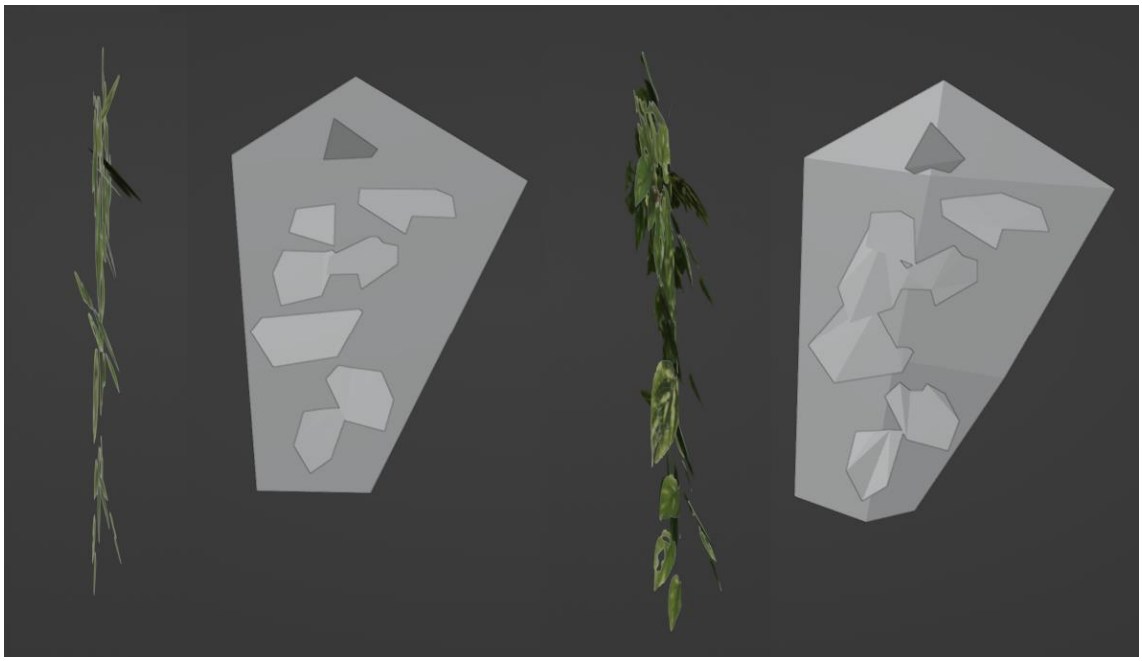


PICTURE 7. Meshes split differently; the left is formed by 3 triangles and the right by 10 triangles.



PICTURE 8. Differently divided planes showing how few triangles enhance the overall shape (Serr 2019).

Cutting a few leaves separately and adding them in different angles to the plant gives it a more realistic look and illusion that it has more geometry. This is a good way to add leaves especially for plants that are going to be looked at close range. Combined with slightly curved planes it can create stunning foliage without costing too many polygons. Picture 9 compares flat mesh with a slightly curved version, showing how little adjusting can have a very noticeable impact on the outcome.



PICTURE 9. Comparing flat vegetation with curved vegetation.

3.4 Megascans

Quixel Megascans is a large library of high-quality real-world 3D and 2D assets - including textures, materials, and models. These are free to use for game development in Unreal Engine. Megascans have complete 3D meshes which include many different LODs and options for material quality. Megascans assets are very realistic which results in the most high-quality versions using a lot of polygons.

Instead of using the 3D objects from Megascans, there are atlases which are large textures images containing multiple smaller textures. Atlases can be split into smaller pieces the same way as masked materials in section 3.3. This way the amount of polygons can be reduced to fit the possible limits of the game and it allows greater freedom to create vegetation as one desires. Picture 10 shows some examples of how the atlases look and a highly detailed 3D plant inside Unreal Engine.

PICTURE 10. Megascans plant atlases in the library and a Megascans 3D object in Unreal Engine.

4 CREATING VEGETATION

4.1 Generally

When creating foliage, it is important to follow the same art style as the other elements in the game to make them match. Seasonal variations in foliage, like falling leaves or snow, enhance game settings and storytelling for a more immersive experience.

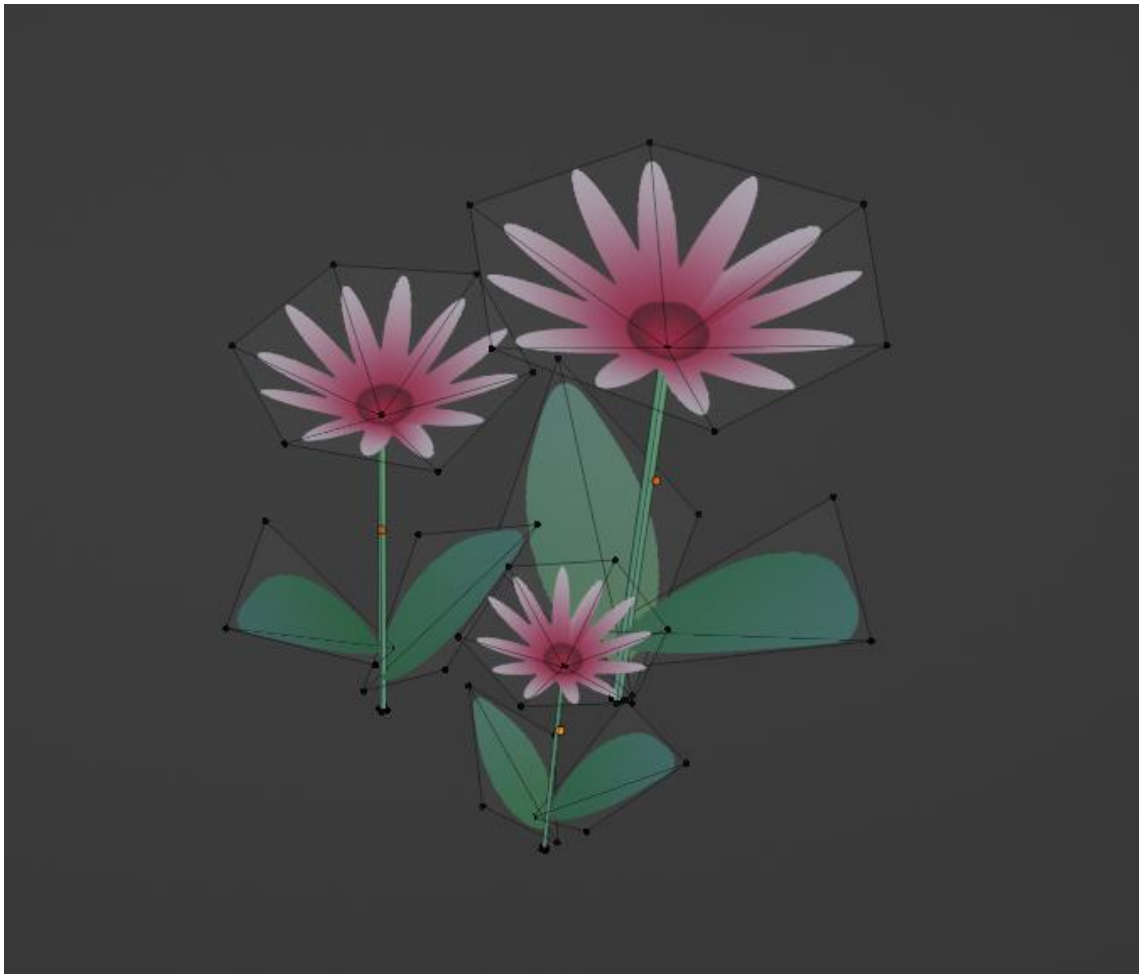
In Unreal Engine there is a setting called cull distance, which adjusts how far away the foliage is going to be visible. This setting can be found in foliage mode under instance settings and changing the maximum value changes how far away the objects are going to be rendered. Depending on the size and verticality of the environment, many objects cannot be seen from everywhere and there is no reason to always render them. Utilizing this technique is effective especially for grass and flowers, but it can also serve as a viable alternative to using LODs in larger environments when working with trees.

To further enhance realism in all the foliage, including subtle differences in sizes, colors, and rotation angles helps. Material settings in Unreal Engine enable adding slight color variation to materials, which allows to create more distinctive foliage. When using foliage painting tools, it is possible to add variations for the sizes by adjusting the scale values.

4.2 Flowers

Creation of flowers follows the same methods as gone through in section 3.3, using alpha materials for more geometry. One texture can be used for multiple different variations of flowers which reduces the materials needed. Less materials require less draw calls which affects performance instantly. Small objects, like flowers, do not really require very high-quality materials which is why it makes sense to fit multiple flower variations within a single texture.

Simple flower shapes can be easily created by moving the middle part to be a bit lower than the edges, as seen in picture 11. Using smooth or auto-smooth shading is recommended as it conceals sharp edges between the petals making it harder to notice how little polygons are used for them. When using SimpleGrassWind in Unreal Engine, be sure to connect the flower meshes to the stem meshes to prevent the wind effect from treating them as separate objects which can potentially cause the stems to poke through the flowers.



PICTURE 11. Flowers with wireframe showing. Each one is formed by 22 triangles.

Instead of relying solely on a single plane per flower, another technique involves using multiple planes per flower. By duplicating and rotating these planes at different angles, one can create a more three-dimensional appearance and add depth to the flowers. Depending on the style of the flowers, this approach enhances their visual appeal from different angles. This technique was used to create the flowers from Fortnite, as seen in Picture 12.

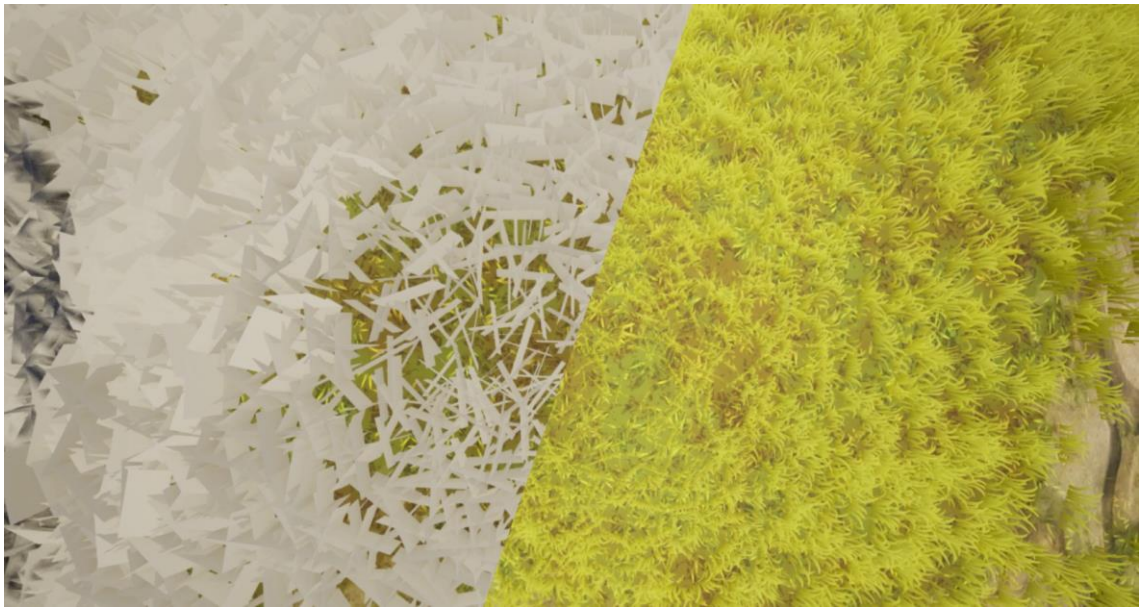
PICTURE 12. Flowers from Fortnite.

4.3 Grass

Focusing on polycount when creating grass or other foliage that is fairly small and is going to be used in most of the scenery can be crucial. When using thousands of grass blades, even one polygon ends up adding a polycount by thousands (Norris 2017). Considering when more geometry is really needed and when simpler assets get the job done can really affect the smoothness of the gameplay.

When creating a dense grass area, a single grass plane can simply be formed with a single plane to reduce polycounts. Changing some of the grass to be at a slight angle helps it look better and fuller when looking from above. Typical way for artists to create grass for games is to use two separate grass planes positioned at 90-degree angles from each other to increase the density. This way the grass planes are not using too many polygons but can fill the area well enough. If the scene is using only a few grass blades, increasing the polycount and adding subtle rotations and bends to them prevents them disappearing from certain an-

gles while maintaining a lush look (Norris 2017). Picture 13 shows densely scattered grass planes without and with material, which helps to understand how many planes are required for thick grass.



PICTURE 13. Grass planes without and with material.

4.4 Trees

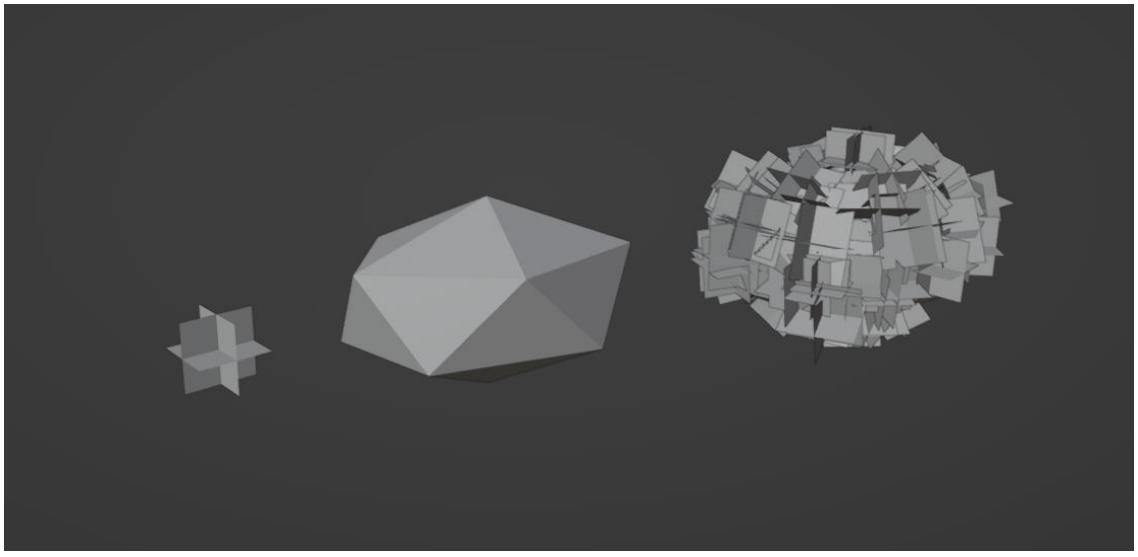
The art style of the game affects a lot on how the trees should be made to fit the style. Modeling the trunk of a tree does not require anything special but planning on how to create the leaves can be challenging. It is common to use multiple planes for leaves, creating an asymmetrical and lush appearance for a realistic effect. For a more stylized look there are other ways, like creating simple shapes and applying materials to them instead of using numerous planes.

Trees can be a very complex and prominent part of the environment which is why one should consider using LODs to improve the performance and ensure a smooth player experience. The following tree design instructions can easily be adapted for the creation of shrubs and bushes.

4.4.1 Particle system

Rather than manually placing tens of planes for leaves, the particle system in Blender is a fast way to scatter objects automatically around another shape. This process requires two separate objects, one for the basic shape and another one to act as the leaves. Allowing rotation for the leaf-object reduces the number of objects needed and gives it a fuller look. The basic shape of a tree can be pretty simple, as long as there is enough room for the leaf planes on the surface. Applying scale in Blender makes sure that all the leaves on the base and in the planes are the same size.

Setting up in Blender, the particle system needs to be set to hair and advanced settings need to be turned on. The most important settings of the system include setting the number of hairs and adjusting the render settings to render as an object while selecting the planes as instance objects. Picture 14 shows the leaf object, basic shape, and the final outcome achieved by using 75 hairs. Scale of the planes can be changed from render settings, and adding some randomness to the scale adds a bit more realism to it. Turning on rotation settings and altering orientation axis to normal allows adding randomness to the rotations as well. Changing shading to auto smooth helps the planes to settle into a more rounded shape and fill the shape better. If there are still empty spots, adjusting the seed or the number of hairs can help to fill it. (Stylized Station 2021.)



PICTURE 14. Planes, basic shape, and the result without materials.

The triangle counts on the trees in picture 15 is 470 triangles each, which is very optimal for games. With bigger planes for leaves the polycount could be even lower without affecting the looks very much. Manually scaling or even deleting some planes can be helpful especially if there are some misplaced objects. Picture 15 shows the finished trees with materials in Unreal Engine but some of the leaves are sticking out of the base very noticeably. Such mishaps can be spotted early on by applying materials during the modeling phase rather than solely within the game engine, making the process smoother.

PICTURE 15. Trees with materials in Unreal Engine.

4.4.2 Basic shapes

Another way to quickly make trees is to create a simple shape and use leaf material on it. This way the shape of the tree can be more interesting, and its silhouette can help the artistic vision to come alive. If the leaf material is not too dense, it could be helpful to create multiple layered shapes, so the foliage is not too see-through.

Leaves on trees in picture 16 are made with three layers of the same model. This method allows more freedom to create interesting shapes. The left one is more

complex and uses 2 040 triangles, and the right one only uses 1 068 triangles. These polycounts are not too high but noticeably higher than in the trees created by using the particle system in section 4.4.1. Using a denser material for the leaves could reduce the number of layers needed, which directly affects the polycount positively.

PICTURE 16. Trees created with three layers of round shapes.

4.4.3 Examples from Fortnite

Epic Games' Fortnite uses alpha textures for foliage. Bushes are created with big and flat planes rotated in different angles to make them dense. The edges of the planes are very noticeable from the right angles, but the polycounts are staying low. Picture 17 showcases a tree and a bush from Fortnite.

Many of the trees are made with curved dome-like shapes that are stacked on top of each other with some leaves sticking out to break the symmetry. Foliage is created like a shell to cover the top parts of the tree leaving the interior of the tree empty without unnecessary leaves which also allows a little light to come through.

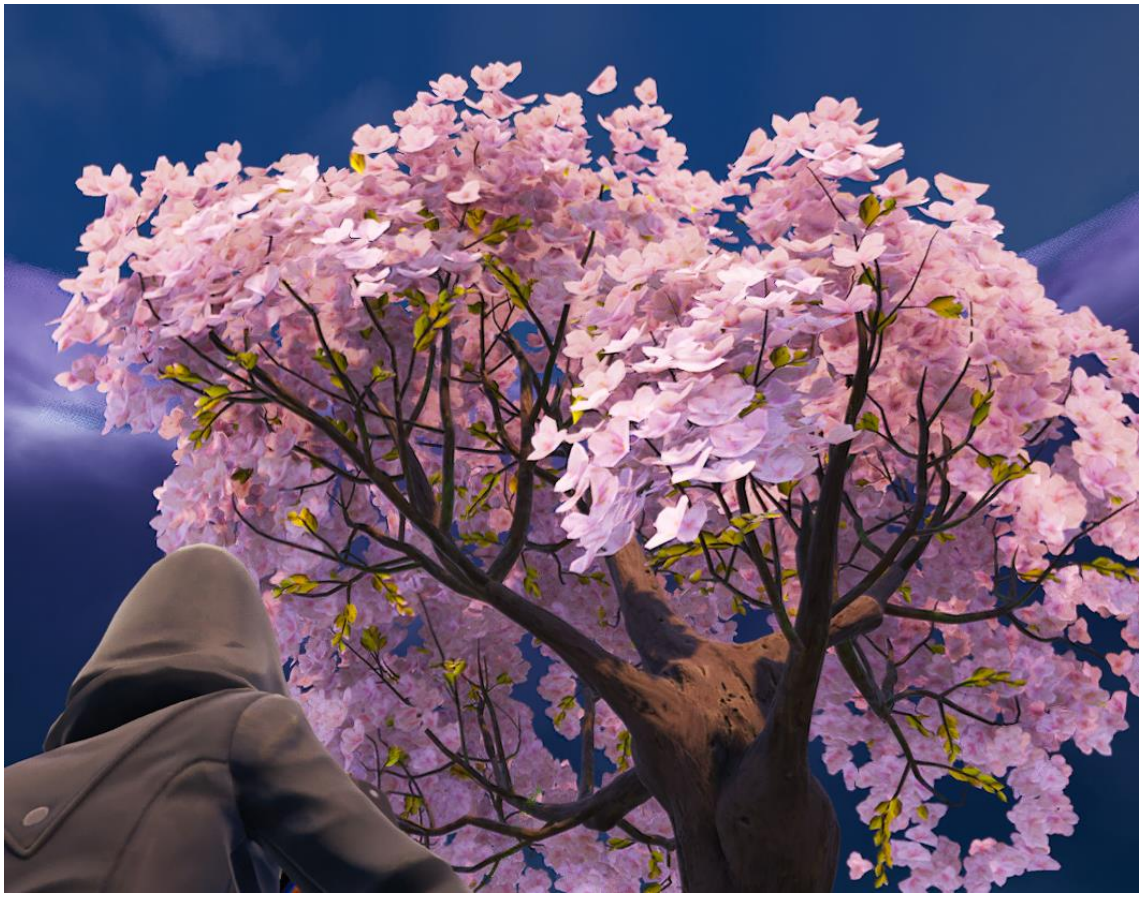
PICTURE 17. Tree and bush from Fortnite.

Other types of trees, like conifers, can also be made with flat alpha textures. The tree in picture 18 uses folded and curved planes to mimic pine needles. In this scenario, the addition of minor geometric details to the planes enhances their visual appeal. Additionally, the inclusion of folded tips on some branches reduces the visibility of the edges of the planes.

PICTURE 18. Pines from Fortnite.

Nanite graphics are a toggleable feature in Fortnite, which should not come as a big surprise since Epic Games is behind both Fortnite and Unreal Engine. However, using Nanite can really affect the frame rate negatively especially when there are a lot of objects on the screen. Fortnite is accessible on various devices but Nanite on the other hand runs smoothly only on high end consoles and computers, which limits its potential user base.

Nanite allows to create insanely detailed and beautiful foliage, but it also requires a lot of performance from the hardware. In Fortnite, each tree with Nanite has around 300 000 polygons, which is quite a lot compared to other methods, like alpha planes (Fortnite 2022). Picture 19 shows a stunning cherry blossom tree using Nanite in Fortnite.



PICTURE 19. Tree using Nanite in Fortnite.

5 CONCLUSIONS AND DISCUSSION

In this constantly evolving game industry, artists are no longer constrained by the limitations of the past. The objective of the thesis was to show different approaches to foliage creation and explore how differently they look and perform compared to each other. The completion of the thesis remained mostly on schedule and the scope grew a bit during the information acquisition. The initial objective was to explore manual methods for creating materials and assets using alpha planes. Technologies like Nanite and Megascans were not initially part of the thesis but their undeniable significance in modern foliage creation led to their natural inclusion in the thesis.

The reliability of the information is provided by referencing official documentation from Unreal Engine and Blender. The examples demonstrated within this work were created by the author, unless explicitly stated otherwise. The thorough testing of these methods was completed in Unreal Engine 5.1.1 and Blender 3.4.1 to ensure their effectiveness and accuracy.

The tools and techniques explored in this thesis have shown the path towards more immersive and visually stunning game foliage. As games continue to push the boundaries of what is achievable, the creative potential and the array of methods for foliage design expand. The groundbreaking innovations, such as Nanite and Megascans, are transforming game development artistry rapidly making artists' jobs much more pleasant. The tools enable working more productively with automating repetitive tasks and letting artists focus on their creative vision instead of technical obstacles. This will ultimately lead to the development of richer and more immersive game environments with a faster pace.

In terms of different approaches, there is not a one definitively best method for creating foliage. Everything is situational and artists have to make right decisions on how to approach the foliage creation based on the targeted market and any limitations it might have. Alpha materials are still a viable option for most scenarios but as time goes and tools improve, Nanite-like technologies might become the new standard. Nanite has a potential to become the core technology in real-time rendering, changing the way games are developed and experienced. It might

lead to more detailed virtualized geometry environments and as hardware capabilities continue to progress, Nanite will become more accessible and widely available.

Currently, artificial intelligence (AI) is making significant advancements and in the future, it can be helpful for artists to create assets and materials faster. AI can help reduce the time needed for optimizing materials and assets making the overall development process faster. As AI technology continually advances, there are associated risks including the possibility of it replacing some of the roles in game development. However, AI has the potential to inspire artists with unlimited creative ideas, ensuring the creation of unique and artistic games.

Games are becoming more and more realistic all the time and the environments are starting to mirror reality. The advances in technology consistently expand the possibilities of what can be accomplished in the foliage creation and only the future tells what technologies will be invented and how the current technologies are going to be shaking the industry in the next few years.

REFERENCES

Blender. n.d. Introduction to Nodes. Webpage. Read on 26.9.2023. https://docs.blender.org/manual/en/2.79/render/blender_render/materials/nodes/introduction.html

Fortnite. 2022. Drop into the next generation of Fortnite Battle Royale, powered by Unreal Engine 5.1. Webpage. Read on 5.9.2023. https://www.fortnite.com/news/drop-into-the-next-generation-of-fortnite-battle-royale-powered-by-unreal-engine-5-1

Ivy hanging. CC BY-NC 4.0. Free PNG img. Image. Viewed on 2.8.2023. https://freepngimg.com/png/112296-ivy-hanging-download-free-image

Karmaker, J. 2016. Vegetation creation for video games. 80lv. Webpage. Read on 18.8.2023. https://80.lv/articles/vegetation-creation-for-video-games/

Material blend modes in Unreal Engine. n.d. Unreal Engine. Webpage. Read on 20.8.2023. https://docs.unrealengine.com/5.1/en-US/material-blend-modes-in-unreal-engine/

Materials. n.d. Unreal Engine. Webpage. Read on 31.8.2023. https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/

Maxwell, W. 2019. How many polygons should a game model have. CG Obsession. Webpage. Read on 2.8.2023. https://cgobsession.com/how-many-polygons-should-a-game-model-have/

Maxwell, W. 2021. Does Unreal Engines 5 Nanite make poly count irrelevant. CG Obsession. Webpage. Read on 30.8.2023. https://cgobsession.com/does-unreal-engines-5-nanite-make-poly-count-irrelevant/

Nanite virtualized geometry in Unreal Engine. n.d. Unreal Engine. Webpage. Read on 31.8.2023. https://docs.unrealengine.com/5.2/en-US/nanite-virtualized-geometry-in-unreal-engine/

Norris, J. 2017. Learn plant modelling for video games. Purepolygons. Webpage. Read on 20.8.2023. http://www.purepolygons.com/uploads/6/0/5/5/60558851/tdw216.tutorial.pdf

Selin, E. n.d. How to use alpha transparent textures in Blender. Artistic Render. Webpage. Read on 2.8.2013. https://artisticrender.com/how-to-use-alpha-transparent-textures-in-blender/

Serr, A. 2009. Xtreme plant optimization. Wolfire games. Webpage. Read on 18.10.2023. http://blog.wolfire.com/2009/11/xtreme-plant-optimization/

Stylized Station. 2021. The 3D artist's guide to modular environments - Unreal Engine environment breakdown. YouTube video. Released 27.10.2021. Watched on 29.8.2023. https://www.youtube.com/watch?v=dJnAuVtwxPI

World position offset material functions. n.d. Unreal Engine. Webpage. Read on 18.8.2023. https://docs.unrealengine.com/5.1/en-US/world-position-offset-material-functions-in-unreal-engine/

Zavhorodnia, V. 2022. Stylized Flowers Tutorial. YouTube-video. Released 2.11.2022. Watched on 3.8.2023. https://www.youtube.com/watch?v=5Pov-TfBD29I