

# GRADUATION REPORT

INDEPENDENT ARTS SOFTWARE

Student: Chloe Isabelle Bläker 466660

Coach: Yvens Serpa



# Abstract

There is a lack of coherency in style and easy adaptability to target platforms of digital tree assets on online marketplaces necessitating efficient in-house creation of these assets for Independent Arts Software GmbH. This thesis investigates the feasibility of using Speedtree to create trees in-house and the ease of adaptability to the diverse range of target platforms used by the client. A sample tree species was created in Speedtree based on research and observational data to streamline the creation process as much as possible, and its adaptability was validated via A/B performance testing. Results showed a clear improvement in performance on lower-end target platforms with only very little work done for optimization. It is concluded that Speedtree is a viable tool for efficiently creating digital tree assets and offers great changeability for performance and style requirements.



# Table of Contents

Introduction .....	2
Speedtree Software .....	3
Other Approaches .....	3
Scope .....	3
Indicators of Success .....	4
Research Question.....	4
Theory .....	5
Plan of Approach .....	5
Biology Terms .....	5
Tree Research .....	6
Computer Graphics Terms.....	7
Speedtree Terms .....	9
Development.....	11
Building Birch Trees.....	11
Overview.....	11
Branching Structure.....	12
Leaves .....	14
Textures .....	15
Optimization .....	15
Testing.....	16
Overview .....	16
Setup .....	17
Test A.....	17
Test B.....	19
Conclusion & Future Work.....	23
Reflection .....	24
References.....	25

# Introduction

Independent Arts Software GmbH (IAS) is one of Germany's oldest video game studios (Independent Arts Software GmbH, n.d.). Established in 1990 by Holger Kuchling, it has since produced over 250 titles (Independent Arts Software GmbH, n.d.). There are currently 35 employees, working on several projects in parallel. IAS also provides non-gaming services for software development and IT.

IAS's portfolio spans several generations of consoles. Earlier work focused on handhelds like the Nintendo DS, PlayStation Portable, and Gameboy Advance, while more modern projects typically release on all hardware of the current generation of video games such as PlayStation 5, Xbox Series X and PC (Independent Arts Software GmbH, n.d.). Additionally, IAS has extensive experience in console and handheld ports.

Because of this wide range of target platforms in the company's project portfolio, performance optimization and work efficiency play a large role. Games must perform well on different hardware and assets must be easily adaptable to the specific requirements imposed by the platform. For artists, this poses some specific problems. In recent years, photoscanned textures and 3D assets have risen in popularity as website traffic research from two of the most popular platforms, Quixel Megascans and Poliigon, shows (Figure 1, SEMRush, n.d.).

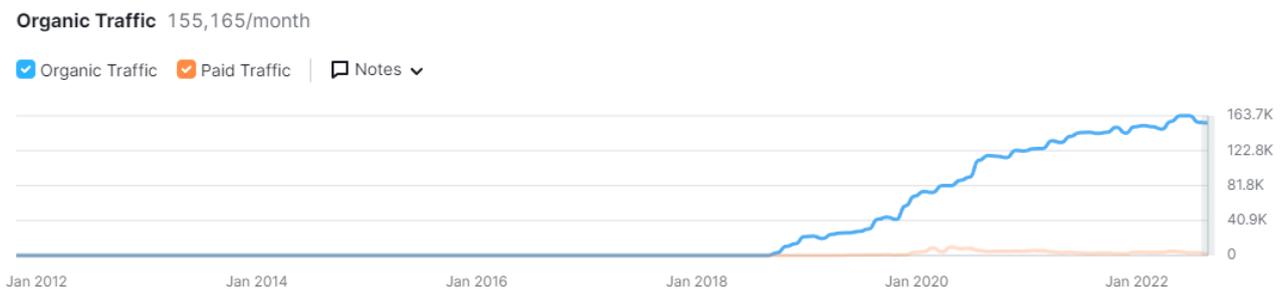


Figure 1: Quixel.com website traffic graph

The use of photogrammetry in video games has also steadily increased in the timespan between 2014 to 2019 (Statham, Jacob, & Fridenfalk, 2020). However, in the context of foliage, more specifically trees, photoscanning is inapplicable due to the issues with scanning such large and constantly moving objects in an outdoor environment. For this reason, Quixel Megascans, Poliigon and Textures.com do not offer full trees on their platforms but rather only separate stumps and trunks (Quixel, n.d., Textures.com, n.d.). This means that artists at Independent Arts have to source 3D trees via different means. There is a very large market for buying, selling, and sharing 3D models such as trees, with leading platforms Turbosquid, CGTrader and Sketchfab hosting hundreds of thousands of products (Turbosquid, n.d., CGTrader, n.d.). Because these platforms are community marketplaces for third-party sellers though, quality, style, structure, and optimization are not consistent between sellers or individual products. Most of these products are also baked models or sculpts leaving little room for major modification without a lot of manual labour by the artist.



Given these limitations, producing these assets in-house to have greater control over the result, is the preferred course of action. Several options for tree creation have been explored by the client in the past with the conclusion that Speedtree (Speedtree, n.d.) would offer the greatest flexibility and cost/work efficiency. However, with only one employee currently trained in using Speedtree, there is an overall lack of experience in the team. Additionally, with the software being very complex, the company cannot afford extensive training for every employee on this tool.

## Speedtree Software

Speedtree is a 3D modelling tool suite for creating plants and trees. It utilises mainly procedural approaches to modelling, letting the user define aspects of a tree's growth and shape, while also including free-hand modelling tools to fine-tune the details of the assets.

This facilitates control over different aspects of a tree's growth and randomizing variations to allow for maximum adaptability of shape, look, age and season depending on the situational needs (Speedtree, n.d.).

This workflow allows the artist to define the constraints and guidelines for an individual tree to grow rather than meticulously modelling each aspect of the asset by hand. Random variation can be added to these constraints to generate unlimited variations of each tree quickly and easily.

## Other Approaches

While there are many ways to create 3D models of trees, they can generally be separated into two categories: procedural and traditional.

Procedural generation of trees can be achieved with Speedtree but there are a few other alternatives to this software specifically. E-on Software publishes PlantFactory, which is another node-based, procedural tree and plant generation software that behaves very similarly to Speedtree but is mainly targeted for cinematic use rather than games (E-on Software, n.d.). There is also Houdini, published by SideFX which is a highly in-depth procedural generation computer graphics tool suite not specifically aimed at trees and plants (SideFX, n.d.).

Traditionally, 3D modelling suites like Blender, 3DS Max or Maya can be used to model trees and plants entirely by hand using various approaches such as sculpting, box modelling or path-based modelling.

## Scope

The scope of this thesis is limited to creating one species of a tree using Speedtree. Since the goal of this thesis is to further knowledge of Speedtree, creating one species is sufficient to achieve these goals. Non-tree plants such as bushes, weeds or flowers will not be explored.

The usage of Speedtree as a requirement is a given, as stated by the client and other options for creating trees such as through traditional modelling techniques will not be explored. Furthermore, stylization of trees will not be explored; As agreed with the company, it is more valuable to deliver the final product in a realistic style.



Tree creation will focus on (above ground) roots, trunks, and branches. Bark, leaves, flowers, and fruits are not part of the scope of this thesis and will be included in the final product via third-party textures.

## Indicators of Success

Success will be indicated by the validation of target FPS across a variety of target platforms. Value creation for the company will be highest if assets can be quickly adapted to run well on different hardware which should include at least: PC, Nintendo Switch, Mobile Phones (limited to device availability at the company for testing), PS4, Xbox One.

Test data will be analyzed to give recommendations regarding optimization considerations during the design process in Speedtree.

## Research Question

***“How do you increase work efficiency by creating adaptable and flexible 3D trees using Speedtree that can be quickly adapted to run on a variety of target platforms?”***

Several sub-questions can be derived from this main research question:

1. How do you translate real-world trees from data such as theoretical research or observations into digital tree assets using Speedtree?
2. How do you efficiently adapt trees created in Speedtree for different use cases and target platforms?
3. How do you streamline the tree creation process in Speedtree to be as efficient as possible?



# Theory

## Plan of Approach

To authentically recreate a given tree type digitally, very specific information must be gathered. While traditional modelling methods and real-world reconstruction such as through LiDAR (a radar-like scanning system using light) scanning can provide satisfactory results with less input data (Neubert, Franken, & Deussen, 2007), procedural modelling relies on defining a wide set of specific rules or parameters which the software uses as guidelines to grow the resulting tree from. This process needs a deeper understanding of the underlying principles of tree growth.

There is a large selection of books available on common broad-leaf trees and identifying different species such as *Our Native Trees and how to Identify Them: A Popular Study of Their Habits and Their Peculiarities* (Keeler, 1902) or *Trees: A Guide to the Trees of Great Britain and Europe (A Little Guide in Colour)* (Bretau, 1966). These types of resources are commonly available but give little attention to branching behaviour and wooden structure and instead focus on the identification of trees based on leaves, bark, fruit and flowers.

For this reason, inspired by the work done by Stava et al. in *Inverse Procedural Modelling of Trees* (2014), a diverse selection of input sources such as real-world observations and visual research will be used to substitute the written information.

## Biology Terms

### ***Bifurcation / Forking***

According to research from Slater and Harbinson, a fork is defined as the joining of two limbs (branches) of roughly equal diameter, in contrast to a branch which is a tree limb with a significantly smaller diameter than its parent trunk. However, this paper points out that the difference is an artificially created one for ease of reference and that many limbs found on trees fall neither into the category of branches nor forks.

## Fractal / Fractal Growth

Trees generally follow a fractal structure in growth. This means that larger branches split off into increasingly smaller branches which exponentially increases the overall number of branches the smaller the level of detail gets. Fractal patterns like this can be found everywhere in nature as Figures 2 and 3 show.



Figure 2: Fractal growth on a fern



Figure 3: Fractal structures in veins of a leaf

## Tree Research

The decision for which tree to focus on for this project was determined by two overlapping factors:

- **(Future) usability for the client:** that is how much use of the specific tree the client will have in future projects
- **Availability of information:** that is how much information is available and easily accessible on a given species of tree

Because of the unknown specifics of the client's projects in the future, it is safest to make assumptions based on probability. Choosing species of trees that are native to large parts of the world and their habitats provides the greatest chance of the product being of immediate value to any future projects. Furthermore, a choice of a common tree species will also provide the largest breadth of available information during product design regarding the tree's growth. Additionally, several species of tree may be combined into one to maximise available information usage. This especially applies to species that share most of their large-scale identifying features and less so for smaller-scale features like leaves.

### **Birches**

Taking into consideration the final product's target audience, which is video game enthusiasts rather than biologists, it has been decided to combine three species of birch tree, that commonly grow in North America and Europe, and are closely related in their broader characteristics. These three species are: *Betula papyrifera* (*Paper Birch*), growing commonly in the northern regions of the United

States of America (Kew Science, n.d.), *Betula pendula* (Silver Birch) and *Betula pubescens* (Downy Birch), both of which grow commonly in middle to northern Europe (EUFORGEN Secretariat, n.d.)

Birches are part of the *Betula* genus in the *Betulaceae* family. They are medium-sized trees usually growing 15-25 meters tall (Myers, 2022) and distributed in temperate and cooler climates (Kew Science, n.d., EUFORGEN Secretariat, n.d.). They are easily recognizable by their smooth, silvery white or grey bark (Figure 4) and their light canopy featuring small oval or triangle-shaped leaves (Figure 5, Beck, Caudullo, de Rigo, & Tinner, 2016).

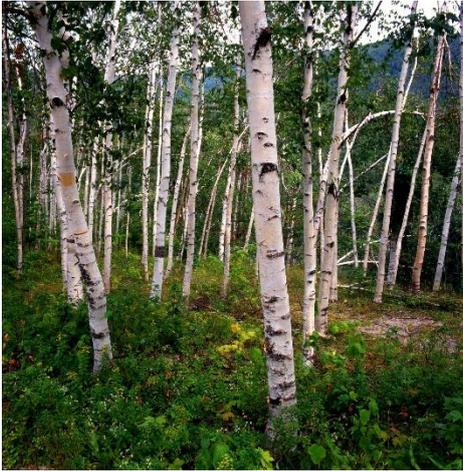


Figure 4: Paper birch trees



Figure 5: Paper birch leaves

## Computer Graphics Terms

The inherently technical aspect of working with and optimizing content for computer graphics generally and game development specifically requires some understanding of the terms and concepts involved in designing and validating assets. Several terms and concepts are explained for the reader's understanding in the following.

### **Procedural Generation**

Procedural generation is the process of creating data using computational methods (Van Brummelen & Chen, 2018). This can be entirely self-sufficient although in the context of computer graphics, this is usually the creation of data by computers with specific inputs by the user. In this context, data refers to graphics such as 3D models or textures. In the procedural generation of computer graphics, the user inputs specific parameters that act as guidelines for the computer to generate variable data from. This data, such as a tree 3D model, will be constrained to the bounds of the parameters input by the user but large parts of otherwise manual labour in modelling or sculpting are replaced by computer processing. This allows for faster and more efficient work while also allowing variability in results which can be a desirable effect for randomness.

### **LOD (Level of Detail)**

Level of Detail or LOD in computer graphics is the process of reducing the quality and resolution of a 3D model and/or texture based on its distance to the viewer (Arm Developer, n.d.). The further away the asset is from the viewer, the less detail is necessary to be displayed due to the smaller size on-screen. Reducing an asset in this way conserves system resources and results in better

performance. LODs are generated semi-automatically or manually and divided into stages or levels which are numbered starting at 0. LOD0 is the highest resolution, LOD1 is one stage below that and so forth. There is no fixed minimum number of levels as this is dependent on the original asset's fidelity and the target platform. Figure 6 shows an example of different LODs on a rabbit model.

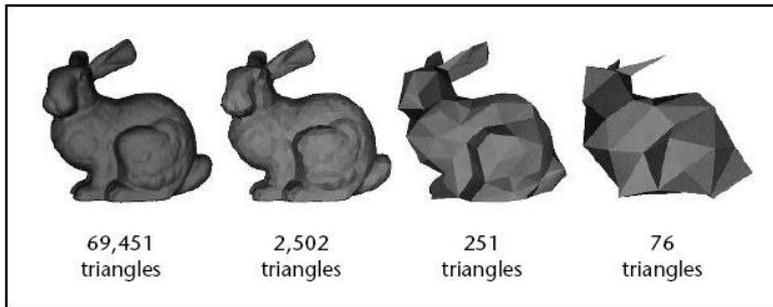


Figure 6: Level of detail on a rabbit 3D model

### Texture Atlas

A texture atlas is a way of combining several smaller texture images into one large texture to conserve system resources. Texture atlases usually contain frequently accessed textures that are shared among a larger amount of assets. This approach of combining textures can improve performance because only one texture has to be loaded into memory and accessed rather than multiple at once (polycount wiki, n.d.). Figure 7 shows an example of a texture atlas.



Figure 7: Texture Atlas

### Nodes

Nodes in computer graphics are a way of displaying connected blocks of data or logic to the user in an intuitive way, usually as rectangles on a 2D canvas (Figures 8 and 9). These nodes are connected to each other and can be chained together to create larger structures. The data these nodes contain can be edited and influence the final output of the node tree.

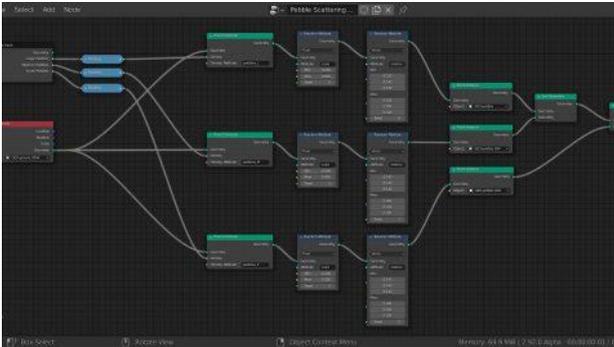


Figure 8: Nodes in Speedtree

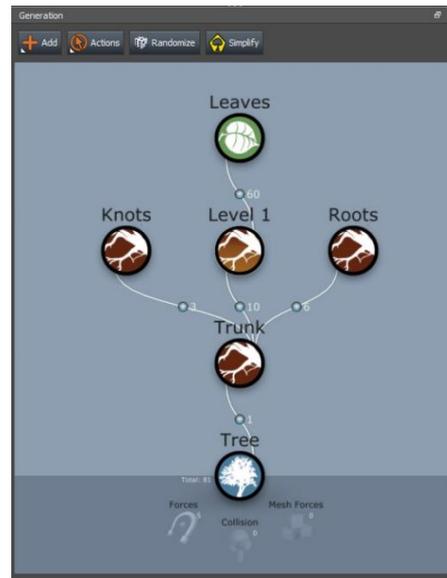


Figure 9: Nodes in Blender

## Frame Time

Frame time is a common performance metric that describes the time it takes a hardware platform to draw a single frame onto the screen. This is usually measured in milliseconds. It serves as a more accurate metric than frames per second (FPS) due to the way both metrics are commonly measured. Frames per second are usually counted over the course of several frames and extrapolated out into one full second whereas frame time will take a perfect snapshot of just one singular frame and display the time it took to render said frame. This can make it easier to spot stuttering as the results are not averaged or smoothed out. Frame time metrics can be easily converted to the FPS format for ease of understanding (Roach, 2022).

## Speedtree Terms

### Generation Modes

There are several different generation modes for branches in Speedtree. The most important ones are:

- **Bifurcation:** Branches are placed along bends or kinks in the parents' trunk. This yields realistic results but leaves little direct control
- **Proportional:** The number of branches varies depending on the size of the parent. Smaller parents will create fewer branches with this method
- **Absolute:** A fixed number of branches will always be generated
- **Interval:** Branches are created alongside a trunk in intervals. The distance between these intervals can be adjusted. This can produce equally spaced-out branches along a trunk.

## Clusters

Instead of using individual leaves which can quickly increase polygon counts, a texture containing many leaves in a twig branching structure is mapped onto a low-resolution mesh that can be further deformed. This is referred to as a “cluster” in Speedtree. An example of a cluster with leaves can be seen in Figure 10.



Figure 10: Screenshot of a birch leaf cluster

# Development

## Building Birch Trees

### Overview

Basic parameters from the previous research were used inside Speedtree to constrain the overall shape and size of the birch tree. This included a height of between 15-25 meters and an overall slender shape with upward sprouting branches that thin rapidly and can be easily weighed down by leaf clusters.

Further reference for determining branching structure and shape was drawn from observational data such as in Figure 11.



Figure 11: Birch Tree



Figure 12: Birch tree with branch hierarchy painted in

To ensure common understanding, the branching structure of the tree will be referred to in hierarchy levels from now on. Branches will be referred to as first, second, third and so on level based on their hierarchical distance from the trunk. Branches that are sprouting directly off the trunk are first-level branches, branches that are generated off these are called second-level and so forth. Figure 12 shows the trunk in blue, first-level branches in red and second-level branches in yellow.

## Branching Structure

In Speedtree, the trunk was set to be long and slender with a mostly straight shape although some large-scale bends were introduced as noise to the trunk's shape to add slight variation and act as generation points for bifurcating (dividing) branches later. As observed from the reference pictures such as Figures 11 and 12, these bends are usually limited to the upper 3/4ths of the trunk to ensure that the lower fourth of the tree keeps a mostly straight and branchless trunk.

The bifurcation branch generation method was used on these trunks to generate a low number of large branches. This generation method places branches on kinks and bends of the parent structure and creates a very natural-looking branch although artistic control over placement is more limited due to being dependent on parent deformations. For this reason, it was only used for a low number of larger branches off the parent and most other branches were generated using a different method. For these first-level branches, it was important to limit each individual's length based on its height on the tree so that branches would naturally become thinner and shorter the higher up they were generated on the trunk. Otherwise, there would be unnaturally long and thick branches coming from the top end of the trunk. A negative example of branches not decreasing in length the higher up the trunk they are can be seen in Figure 13

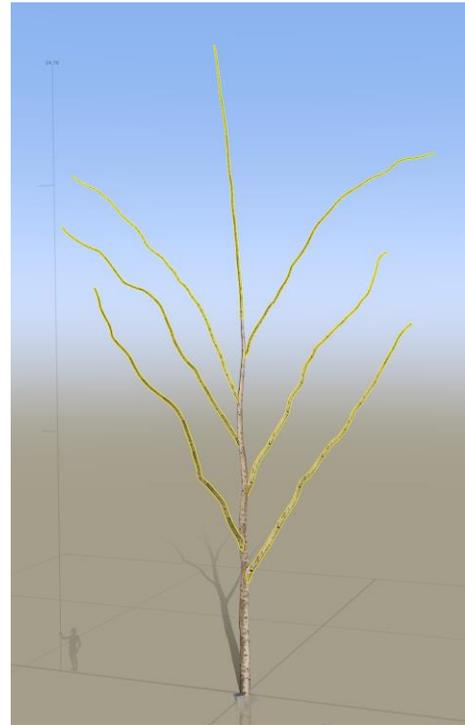


Figure 13: Trunk with branches not tapering off in length

A second layer of bifurcating branches, meaning branches that sprout off bends of their parents, was added to the first to finalise the overall deciding shape of the tree's crown with only a small number of defining branches.



Figure 14: Smaller, Second level branches in yellow

To each layer of branch, a child branch node was added that generates hanging branches. Because of the slenderness of birches and smaller branches, these would be weighed down heavily by the leaves that will eventually be generated onto them (Keeler, 1902, p. 300). Two generation methods were chosen here: Interval and Proportional. Interval generation places branches at fixed intervals along a parent branch's length with each branch being spaced an equal distance apart, whereas the proportional generation method places a varying number of branches on a trunk based on the size of that parent trunk. Interval generation was used for the first-level branches as they are the longest and needed to be covered consistently in branches. Interval generation also provides a high level of artistic control in the exact placement of branches since it is a direct generation

method, unlike bifurcation which is dependent on the shape of a parent.

Proportional generation places nodes according to the size of the parent, the larger the parent the more nodes are generated. This creates a more random distribution than interval generation and was used for the second-level branches. Figure 14 shows these hanging branches on the tree marked in yellow. Similar hanging branches were placed using the proportional method along the trunk of the tree as this is commonly seen on larger birch trees. These are complemented by another set of first-level branches coming off the trunk that sit in length between these hanging branches and the original bifurcating branches. These come out of the tree at shallower angles and branch off into thin twigs and are marked in yellow in Figure 15

The final branching structure of the Birch tree is two to three levels deep and contains 9 nodes including the trunk. Figure 16 shows the node tree that defines the overall structure and hierarchy of the branching structure inside Speedtree



Figure 15: Smaller-level branches



Figure 16: Tree branch nodes in Speedtree

## Leaves

Trees generally follow a fractal structure in growth. This means that larger branches split off into increasingly smaller branches which exponentially increases the overall number of branches the smaller the level of detail gets. Creating this amount of detail by modelling the smallest twigs in a tree's canopy separately would quickly result in a total triangle count too high to perform well in games. For this reason, twigs and leaves were created using Speedtree's cluster system. Small twig and leaf structures are created separately and rendered as textures which can then be mapped onto much lower resolution mesh planes on the tree with fewer triangles than modelling each twig separately.

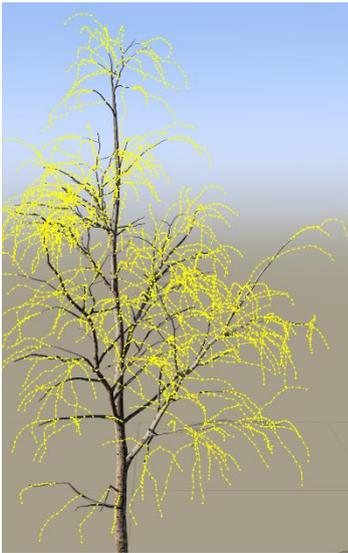


Figure 17: Spline branches



Figure 18: Bare clusters mapped to spline branches

For this tree, three kinds of clusters were created. A larger scale bare cluster that has no leaves but anchor points for other clusters. These bare clusters follow the shape of branches without geometry in Speedtree to take advantage of the branch deformation system. Figure 17 shows spline-only branches in yellow and a close-up of a cluster mapped onto one of these branches in Figure 18.

Figure 19 shows a cluster with leaves and a bare cluster. Leaf clusters were created similarly but instead of being mapped onto branches; they were distributed on anchor points manually placed on the bare clusters beforehand.



Figure 19: Leaf cluster and bare cluster in Speedtree

Figure 20 shows leaf clusters distributed on bare clusters. These leaf clusters are curled and folded at a random angle so that a natural look is created.

This hierarchy of bare twig clusters creating anchor points for leaf clusters is repeated for every high-level branch on the tree creating the final canopy as seen in Figure 21

## Textures

Bark textures were taken from the Quixel Megascans library. The typical papery white birch bark texture, as seen in Figure 4 and referred to in the theory section, is mapped onto the trunk and large bifurcating branches whereas smaller branches are assigned a more uniformly brown texture. UV settings in Speedtree had to be adjusted slightly to create the desired scale of the texture and a twist was added to hide texture tiling more easily.

## Optimization

Several performance and optimization considerations were made before validating testing to prevent any excess triangle counts without sacrificing quality. This mostly consists of a higher radial polygon density compared to lengthwise along a trunk or branch resulting in rectangular faces rather than square ones. Additionally, the mesh resolution of the smallest branches was reduced as much as possible since these are mostly hidden by the canopy. Welding was also disabled for smaller branches. Welding in Speedtree refers to the connection point of one branch into another being smooth rather than a sharp edge, this adds polygons and is unnecessary for smaller branches and twigs.

Dynamic LOD was also enabled which will automatically reduce radial and lengthwise mesh detail based on the distance to the camera. This system also allows for the culling of smaller branches and leaves.

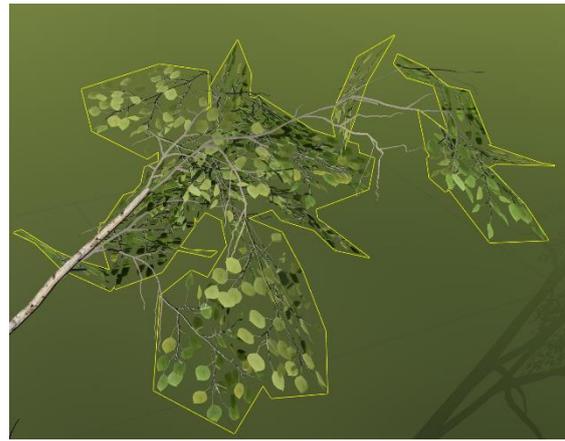


Figure 20: Leaf clusters generated on top of bare clusters



Figure 21: Final canopy

# Testing

## Overview

The research question stated earlier: “How do you increase work efficiency by creating adaptable and flexible 3D trees using Speedtree that can be quickly adapted to run on a variety of target platforms?”, serves as a hypothesis to validate or disprove as part of the testing phase. Quick and easy adaptability for a variety of target platforms was a key factor for the choice of Speedtree and therefore, this should be validated accordingly.

Testing has been set up following an A/B testing methodology. In A/B testing two or more variations of a product are shown to users (or in this case tested for performance) and the better-performing variation is chosen (Young, 2014).

The performance of the trees created for this project has been measured in a controlled environment in a series of tests with increasing density of trees displayed on the screen. The environment is a flat terrain plane with a series of cubes in the middle to periodically occlude trees and other parts of the environment during the camera flythrough to increase realism. Figure 22 shows a screenshot of this environment

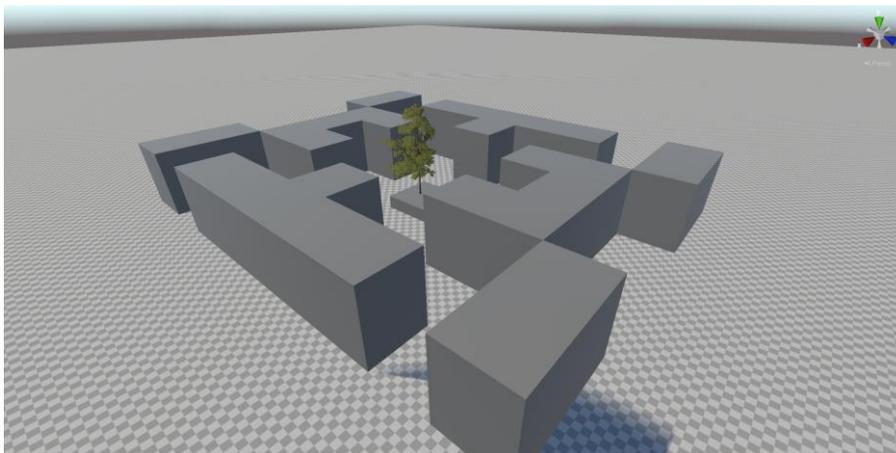


Figure 22: Blank environment scene

Performance has been recorded in frame time and converted to FPS for the convenience of the reader. Test A serves as a control and no optimization modifications have been made to the tree or its setup in the game engine beyond what was described in the previous chapter. It can therefore be assumed to be an out-of-the-box experience with minimal modification.

For test B, trees have been quickly modified based on the results of test A, such as reducing the number of triangles, adjusting the LOD settings and reducing texture resolution, to increase performance for lower-end hardware.

The goal of the testing phase is not to determine overall performance, but rather the ease and quickness with which optimizations can be made inside Speedtree to increase overall performance. Since the platform targets are all consoles in this context 60fps is considered ideal and above 30fps is

considered playable (IONOS, 2022), with the exception of the Nintendo Switch where 30fps is considered an ideal target due to the lower hardware specifications.

## Setup

A custom benchmarking tool was written in Unity that required minimum user input. It can automatically load all relevant scenes and log performance metrics into CSV files. There are 7 scenes in the benchmark that are relevant to the test. A control scene containing an empty environment and terrain and 6 copies of this scene with an increasing number of trees placed in them: 10,000; 20,000; 40,000; 60,000; 200,000 and 500,000 trees respectively. A pre-determined camera path of 30 seconds in length is flown off for each run before every scene is automatically unloaded, a new scene loaded, and the benchmark run on the new scene. To increase the sample size this run of 7 scenes is repeated afterwards and the results are averaged between the two runs.

Tests were conducted on the following platforms: PlayStation 4, PlayStation 5, Xbox Series X, and Nintendo Switch which make up the majority of platforms Independent Arts currently develops for.

The full test results can be found in Appendix **XX**.

## Test A

Test A was run with no modifications made to the tree beyond what was described in the previous chapter, i.e., very roughly defining automatic LOD settings and reducing polygons along the trunk and branches to a reasonable amount. The exported tree had the following triangle counts (Figure 23):

LOD0: 55,252



Figure 23.1

LOD2: 6,458



Figure 23.2



LOD1: 22,298



Figure 23.3

LOD3  
(Billboard): 36

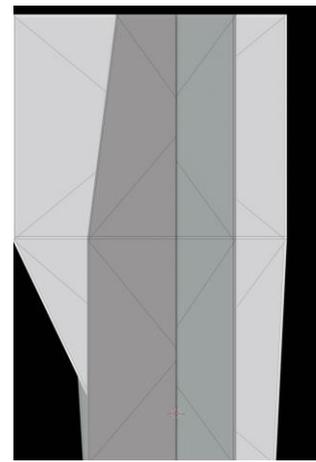


Figure 23.4

Figure 23: Test A Tree LODs

## Results

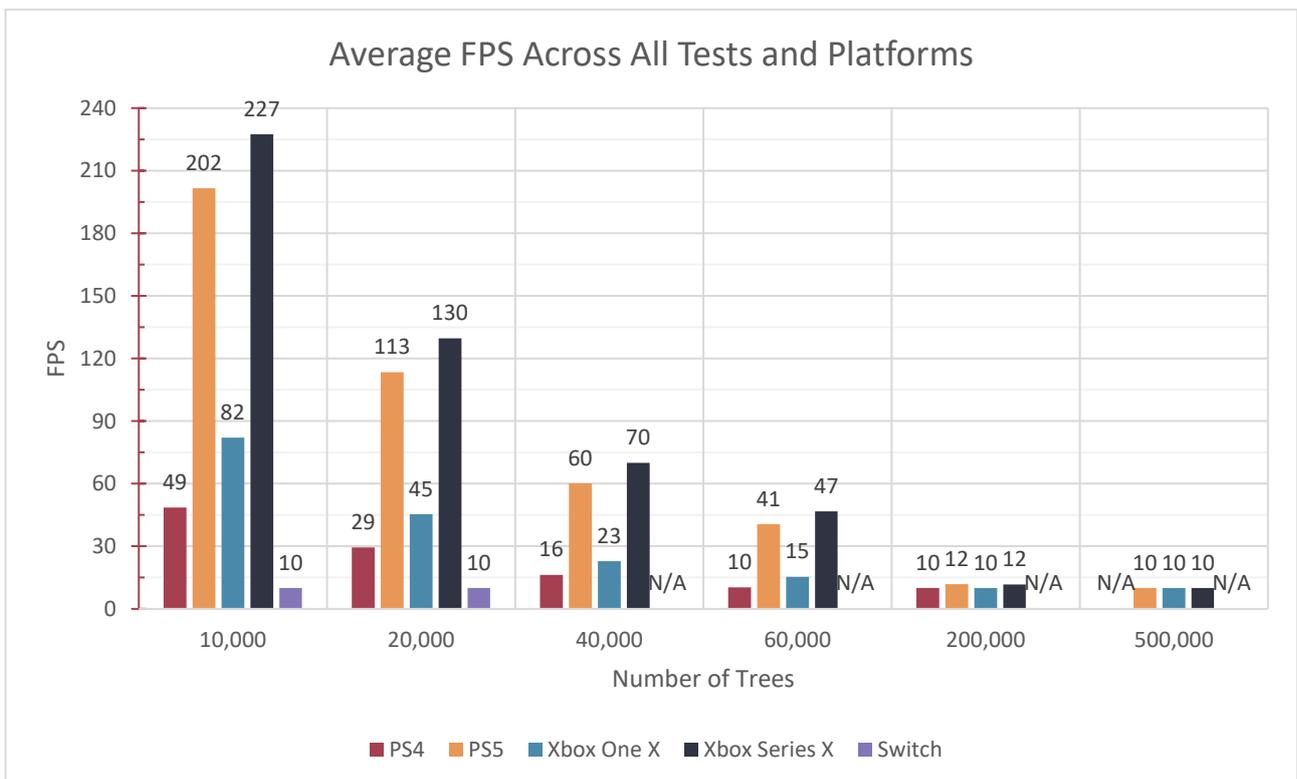
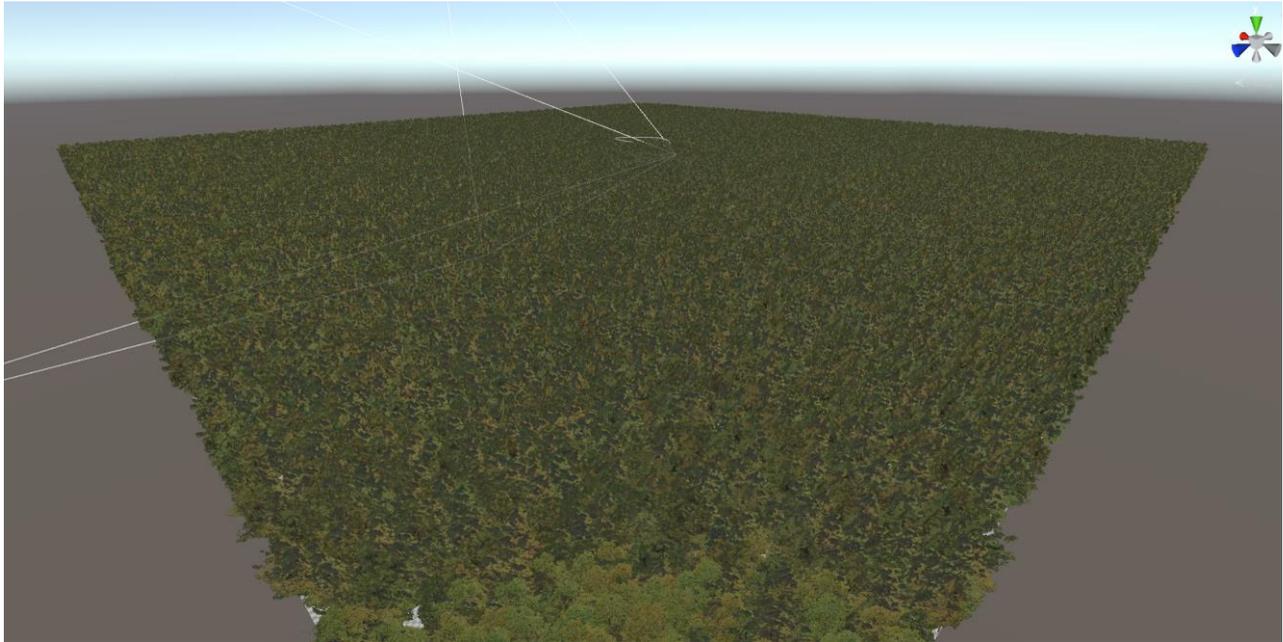


Figure 24: Test A result chart

## Analysis

It is important to note that the logs of the Xbox Series X were incomplete due to an unknown issue with console output, but the results are still illustrated well enough. The two current generation consoles, PlayStation 5 and Xbox Series X were able to handle 40,000 trees at or above 60 frames per second and 60,000 trees at over 30 frames per second (Figure 24). Figure 25 shows the benchmark scene with 60,000 trees loaded.



*Figure 25: Tree scene with 60,000 trees*

The PlayStation 4 data is more difficult to read as the console seems to be clamping its FPS output in steps to the power of 3. It was however able to 20,000 trees at around 30 frames per second which is expectedly lower than the newer and more powerful consoles.

The Nintendo Switch was not able to run even 10,000 trees on screen at more than 10 FPS. Therefore, this test is considered a failure for the Nintendo Switch and adjustments have to be made especially for this console for test B as this was the worst-performing platform out of all. Improvements may yet be seen on other platforms as well, but playability was a given at some amount of rendered trees on all platforms except the Switch, which failed outright.

## **Test B**

Two modifications were made in test B. First, the polygon count of the trees was reduced dramatically from 55,252 triangles in LOD0 of test A to 9,821 triangles in LOD0 of test B as seen in Figures 25 and 26. This was achieved by reducing the length and radial mesh resolution of the trunk and branches inside Speedtree. Furthermore, the leaf and twig clusters were modified to be larger in size and contain more leaves each, this allows for use of fewer overall clusters on the tree which reduces the triangle count further. Additionally, texture resolutions were halved from 2048px to 1024px, and textures were combined onto one atlas instead of being handled separately.



LOD0: 9,821



Figure 26.1

LOD2: 1,329

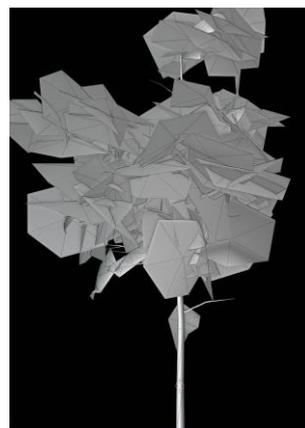


Figure 26.2

LOD1: 3,903



Figure 26.3

LOD3:  
(Billboard): 36

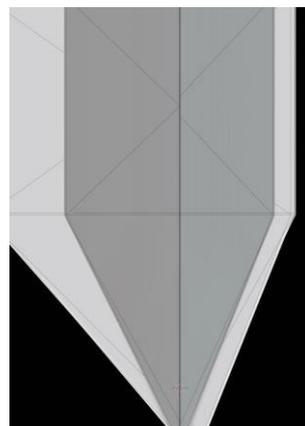


Figure 26.4

Figure 26: Test B Tree LODs

Secondly, the LOD settings inside Unity were tweaked. Figures 27.1 (Test A) and 27.2 (Test B) show that lower resolution LOD levels take up a higher percentage of the total which renders them at a closer distance to the camera, reducing overall triangle counts on-screen. It is important to note that no changes were made to the scenes themselves, the number of trees inside each scene and the culling settings of the Unity project as the only variable between the two tests should be the trees themselves.

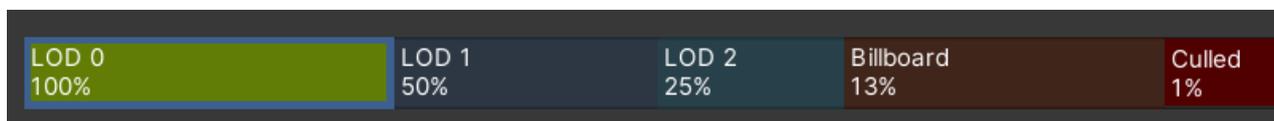


Figure 27.1: LOD group Test A

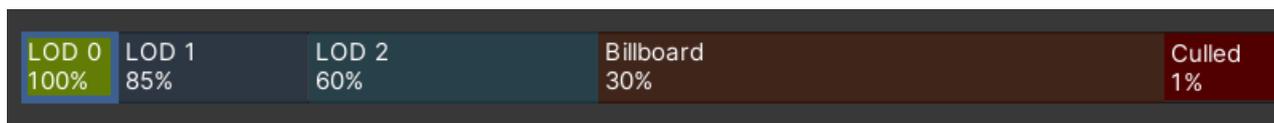


Figure 27.2: LOD Group Test B

Figure 27: Unity LOD group configurations

All these modifications were made within about one hour of work.

## Results

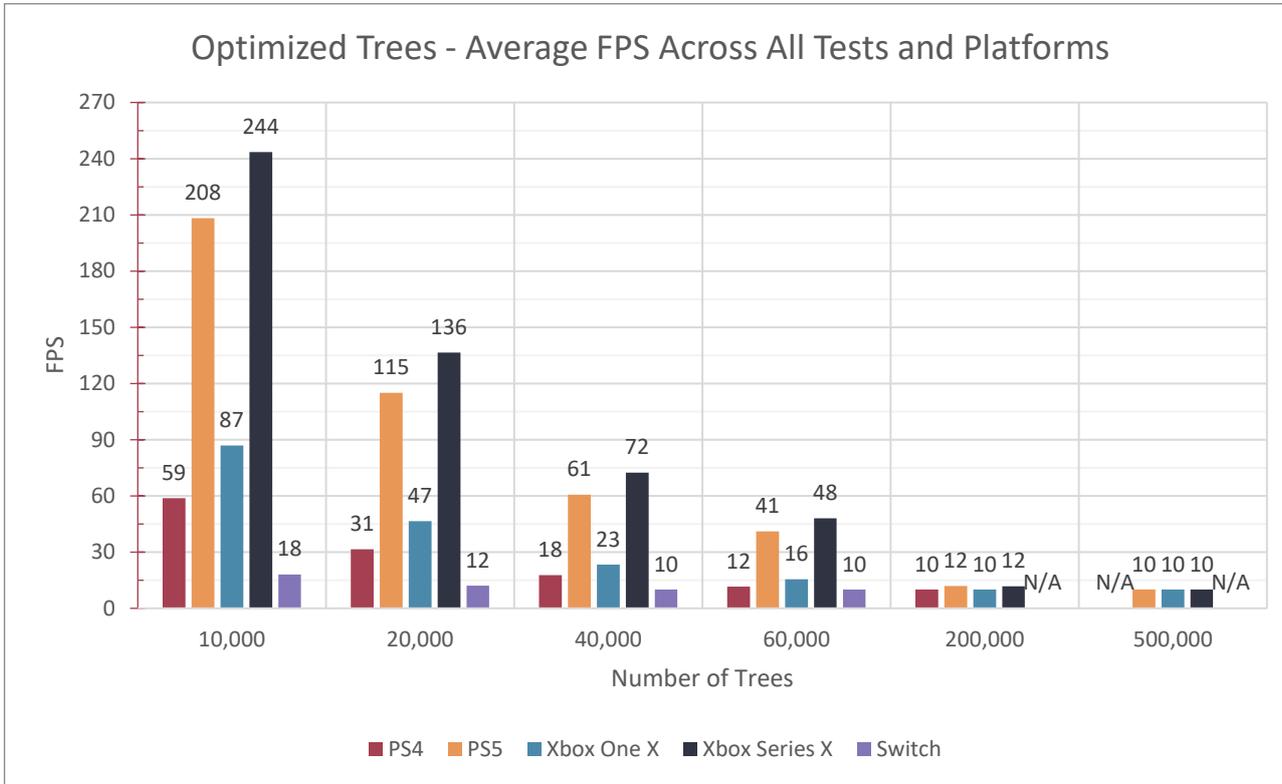


Figure 28: Test B result chart

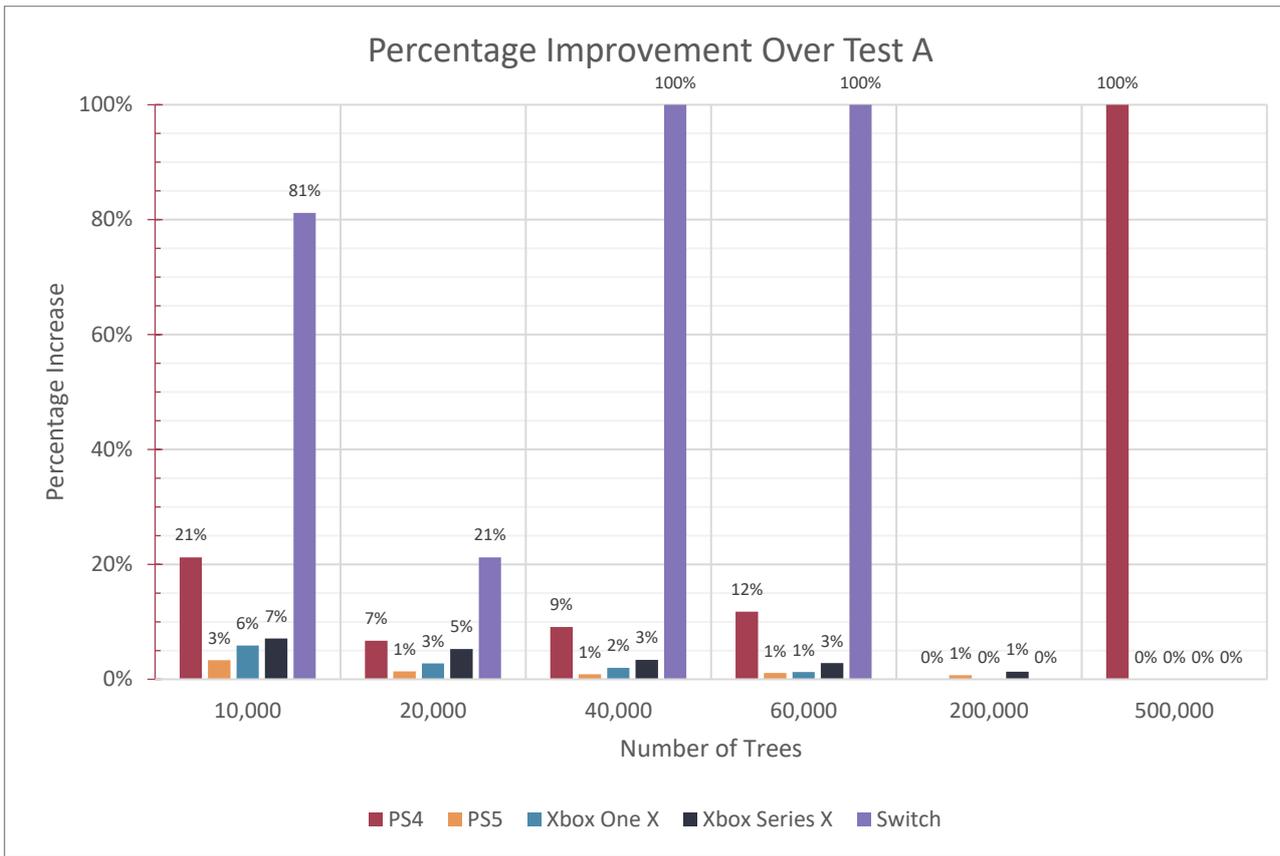


Figure 29: Percentage improvement of test B over test A

## Analysis

Figure 28 shows the absolute performance results of test B, while Figure 29 illustrates the percentage improvement in average FPS as compared to test A and shows that all platforms have seen an increased average FPS. Especially the two worst-performing platforms of test A: PlayStation 4 and Nintendo Switch, have seen massive improvements. In test A, the Switch crashed at rendering 40,000 trees but was able to render up to 60,000 in test B which has been marked as a 100% improvement in Figure 29 for clarity despite technically being an infinite improvement. The PS4 has seen an average performance improvement of 8% across tests with a peak of 21% in the 10,000 test. The Switch has seen an average improvement of 34% across tests with a peak of 81% in the 10,000 trees test.

All the other platforms only saw a marginal performance improvement which may be due to other factors not included in this test. For example, since the culling settings were not changed between the tests there was always the same number of trees rendered in tests A and B and the performance bottleneck in higher tree number tests could be due to draw call amounts. This could be explored further in the future.

The modifications made to the trees between the two tests were quick and not a huge amount of time or effort was required to produce a noticeably better-performing result which is a very positive result for this test.



# Conclusion & Future Work

The research question: “*How do you increase work efficiency by creating adaptable and flexible 3D trees using Speedtree that can be quickly adapted to run on a variety of target platforms?*” was broken down into several sub-questions:

1. How do you translate real-world trees from data such as theoretical research or observations into digital tree assets using Speedtree?
2. How do you efficiently adapt trees created in Speedtree for different use cases and target platforms?
3. How do you streamline the tree creation process in Speedtree to be as efficient as possible?

When recreating real-world trees inside Speedtree it is helpful to use theory and literature as a baseline to acquire foundational facts about the tree one is trying to recreate. This can include its height, width, general shape, density of the canopy and branches and bark colour. It is also important to consider taxonomy, the science of categorizing and classifying (Convention on Biological Diversity, 2010) to determine what species of tree is to be recreated and whether several species can be summarised into one tree due to very similar features. This can then drive the research for observational data. Observational data such as reference images, diagrams or real-world observations are most helpful in the active modelling phase. It is easiest to break a tree down into its hierarchical elements from the lowest, trunk, to the highest, leaves, and focus on each element in isolation, gradually narrowing the focus to details and building up a tree one element at a time. This process also strongly correlates with streamlining the tree creation process inside Speedtree as the software uses a hierarchical system of nodes to represent each element of a tree on a flat canvas with the ability to change each element individually.

Speedtree also offers simple but powerful tools for optimization in regard to a wide variety of target platforms. Polygon counts and texture resolutions of each element can be changed quickly and easily and a semi-automatic LOD system speeds up the process of LOD creation immensely. However, it is worth considering a targeted platform early in the tree creation process as some elements such as the cluster system used for creating leaves and fronds require creating those as separate files and rendering them out as textures. It is easier to tailor the clusters for a specific platform while creating them rather than changing them later. Although not impossible, this saves time in the optimization process.

The main indicator of success described the quick adaptability of trees to meet a specific performance target on different target platforms. Testing has shown that in a very small amount of time, changes can be made easily to the tree to improve performance on lower-end hardware dramatically.

As the scope of this thesis was limited there are several recommendations for future work and investigation to be made. It is possible to use Speedtree to create non-tree plants such as bushes, flowers, grass, and other foliage. The software’s usability and adaptability in these use cases may need to be examined in addition to the tree creation process. Furthermore, alternative ways of creating digital foliage 3D models exist and could be explored and compared to the findings of this thesis. Usability tests were not run due to time constraints and may provide further insights into the learning curve of Speedtree and compared to other methods of foliage creation. This may directly impact the value creation for the company as employees need to spend time learning new software and tools. A



guide or recommendation document could be created to aid in teaching tree creation using Speedtree to other artists more quickly. Finally, more time may be spent investigating performance bottlenecks discovered in the testing phase that may be unrelated to the work done inside Speedtree.

## Reflection

Overall, this project went just as planned and expected. However, there are a few things I would have liked to do differently and some things that had to be scaled back to fit within the timeframe of this semester.

Originally, I intended to create a guideline document, as a direct result of the findings in the research and development part, that could be used by colleagues to ease the learning curve should they need to pick up Speedtree in the future. This document could have served as a cheat sheet to provide easy steps to follow to quickly research and build trees inside Speedtree. Sadly, the time did not allow for this.

Additionally, I intended to build more than one species of tree although one was perfectly adequate for the goals set out in the beginning and the testing phase.

One of the biggest challenges was managing 20 hours of non-study work in the company beside the graduation project work. This went well in the first month or so, but I noticed symptoms of stress and being overworked very quickly and although in the last month I focussed entirely on my graduation, I think a lot of stress could have been avoided doing this from the beginning.

There are also several points I would have liked to follow up on, but that time did not allow for such as testing at least one alternative way of modelling a tree such as inside Blender and doing a comparison to the usability and adaptability of Speedtree. This would have served validation of viability a lot better as different alternatives would have been presented and/or ruled out.

I am however happy with the professional product, my company supervisor has expressed his satisfaction with the value generation to the company, especially me being now trained in using Speedtree for future projects, and I think the thesis turned out better than I originally expected going into this.



# References

- Arm Developer. (n.d.). Geometry Best Practices for Artists. Retrieved October 28, 2022, from <https://developer.arm.com/documentation/102496/0100/Level-of-Detail---LOD>
- Beck, P., Caudullo, G., de Rigo, D., & Tinner, W. (2016). *Betula pendula*, *Betula pubescens* and other birches in Europe: Distribution, habitat, usage and threats. In *European Atlas of Forest Tree Species*. European Commission. <https://doi.org/10.2760/776635>
- Blender Base Camp. (n.d.). What Are Geometry Nodes In Blender? Retrieved September 27, 2022, from <https://www.blenderbasecamp.com/what-are-geometry-nodes-in-blender/>
- Bretaudeau, J. (1966). *Trees: A Guide to the Trees of Great Britain and Europe (A Little Guide in Colour)* (First Translated Edition). Paul Hamlyn Ltd. Retrieved from <https://www.amazon.de/-/en/J-Bretaudeau/dp/B002QXJTH4>
- CGTrader. (n.d.). Buy Professional 3D Models. Retrieved September 13, 2022, from <https://www.cgtrader.com/3d-models>
- Cold Stream Farm. (n.d.). *Paper Birch (Betula papyrifera)*. Retrieved from <https://www.coldstreamfarm.net/product/paper-birch-betula-papyrifera/>
- Convention on Biological Diversity. (2010, June 4). What is Taxonomy? Retrieved January 16, 2023, from <https://www.cbd.int/gti/taxonomy.shtml>
- E-on Software. (n.d.). PlantFactory: Overview. Retrieved October 28, 2022, from <https://info.e-onsoftware.com/plantfactory/overview>
- EUFORGEN Secretariat. (n.d.-a). *Betula pendula* - EUFORGEN. Retrieved October 6, 2022, from <https://www.euforgen.org/species/betula-pendula/>
- EUFORGEN Secretariat. (n.d.-b). *Betula pubescens* - EUFORGEN. Retrieved October 6, 2022, from <https://www.euforgen.org/species/betula-pubescens/>
- Felinto, D. (2020, December 2). *Everything Nodes and the Scattered Stone*. Retrieved from <https://code.blender.org/2020/12/everything-nodes-and-the-scattered-stone/>
- Independent Arts Software GmbH. (n.d.-a). Games – Independent Arts Software. Retrieved September 27, 2022, from <https://independent-arts-software.de/games/>
- Independent Arts Software GmbH. (n.d.-b). Independent Arts Software. Retrieved September 12, 2022, from <https://independent-arts-software.de/>
- IONOS. (2022, June 22). Frames per second (FPS) in TV, cinema, and gaming. Retrieved January 10, 2023, from <https://www.ionos.com/digitalguide/server/know-how/fps/>
- Ivanov, I. (2006, January 26). *Practical Texture Atlas*. Retrieved from <https://www.gamedeveloper.com/programming/practical-texture-atlases>



Keeler, H. L. (1902). *Our Native Trees and how to Identify Them: A Popular Study of Their Habits and Their Peculiarities* [Digital Scan]. New York, United States of America: Charles Scribner's Sons. Retrieved from <https://archive.org/details/ournativetreesa02keelgoog/page/294/mode/2up>

Kew Science. (n.d.). *Betula papyrifera* Marshall | Plants of the World Online | Kew Science. Retrieved October 6, 2022, from <https://powo.science.kew.org/taxon/urn:lsid:ipni.org:names:32197-2>

Lucy, M. (2021, October 1). *The network of veins that move fluids around inside a leaf shows clear fractal structure. The circulatory system of animals is similar.* Retrieved from <https://cosmosmagazine.com/science/mathematics/fractals-in-nature/>

Myers, V. R. (2022, July 29). 11 Birch Trees Common to North American Landscapes. Retrieved October 6, 2022, from <https://www.thespruce.com/twelve-species-cultivars-of-birch-trees-3269660>

Neubert, B., Franken, T., & Deussen, O. (2007). Approximate image-based tree-modeling using particle flows. *ACM Transactions on Graphics*, 26(3), 88. <https://doi.org/10.1145/1276377.1276487>

polycount wiki. (n.d.). Texture atlas - polycount. Retrieved October 28, 2022, from [http://wiki.polycount.com/wiki/Texture\\_atlas](http://wiki.polycount.com/wiki/Texture_atlas)

potter1992. (n.d.). *Level of Detail*. Retrieved from <https://potter1992.wordpress.com/abc-animation-course-2013-2014/01-3d-modelling-for-animation/level-of-detail/>

Quixel. (n.d.). Quixel Megascans | Trees. Retrieved September 13, 2022, from <https://quixel.com/megascans/home?category=3D%20asset&category=nature&category=tree>

Roach, J. (2022, October 27). What is frame time, and why is it so important in games? Retrieved from <https://www.digitaltrends.com/computing/what-is-frame-time/>

SEMRush. (n.d.-a). poliigon.com: Domain Overview. Retrieved September 27, 2022, from <https://www.semrush.com/analytics/overview/?q=poliigon.com&searchType=domain>

SEMRush. (n.d.-b). quixel.com: Domain Overview. Retrieved September 27, 2022, from <https://www.semrush.com/analytics/overview/?q=quixel.com&searchType=domain>

SideFX. (n.d.). Houdini | 3D Procedural Software for Film, TV & Gamedev | SideFX. Retrieved October 28, 2022, from <https://www.sidefx.com/products/houdini/>

Slater, D., & Harbinson, C. (2010). TOWARDS A NEW MODEL OF BRANCH ATTACHMENT. *Arboricultural Journal*, 33(2), 95–105. <https://doi.org/10.1080/03071375.2010.9747599>

Speedtree. (2014). *Getting Started*. Retrieved from <https://docs.speedtree.com/doku.php?id=gettingstarted>

Speedtree. (2017). What Is Speedtree? Retrieved October 28, 2022, from <https://docs8.speedtree.com/modeler/doku.php?id=st8whatis>

Speedtree. (n.d.). SpeedTree Games – SpeedTree. Retrieved September 13, 2022, from <https://store.speedtree.com/games/>



Statham, N., Jacob, J., & Fridenfalk, M. (2020). *Photogrammetry for Game Environments 2014-2019: What Happened Since The Vanishing of Ethan Carter*. Uppsala University Publications. Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-424429>

Stava, O., Pirk, S., Kratt, J., Chen, B., Měch, R., Deussen, O., & Benes, B. (2014). Inverse Procedural Modelling of Trees. *Computer Graphics Forum*, 33(6), 118–131. <https://doi.org/10.1111/cgf.12282>

Taylor, R. (2017, March 31). *A fern repeats its pattern at various scales*. Retrieved from <https://theconversation.com/fractal-patterns-in-nature-and-art-are-aesthetically-pleasing-and-stress-reducing-73255>

Textures.com. (n.d.). 3D Models and Objects vor Games, Archviz and Rendering. Retrieved September 13, 2022, from <https://www.textures.com/browse/3d-objects/114553#trees>

Turbosquid. (n.d.). 3D Models | Over a Million Models for Download. Retrieved September 13, 2022, from <https://www.turbosquid.com/Search/3D-Models>

Van Brummelen, J., & Chen, B. (2018, May 29). Procedural Generation: Creating 3D Worlds With Deep Learning. Retrieved October 28, 2022, from [https://www.mit.edu/~jessicav/6.S198/Blog\\_Post/ProceduralGeneration.html](https://www.mit.edu/~jessicav/6.S198/Blog_Post/ProceduralGeneration.html)

Vermont Center for Ecostudies. (n.d.). *Paper Birches*. Retrieved from <https://vtecostudies.org/wildlife/plants/paper-birches/>

Young, S. W. H. (2014). Improving Library User Experience with A/B Testing: Principles and Process. *Weave: Journal of Library User Experience*, 1(1). <https://doi.org/10.3998/weave.12535642.0001.101>