FINAL GRADUATION REPORT 2019

smart2it

How to enhance the visual aspect of the application 'Innobrix' keeping the target platforms' limitations in mind?

Saxion University of Applied Sciences
Enschede, the Netherlands

Student:
Name: Bruce Boulton
Student ID: 427360
Email: brucealecboulton@gmail.com

Saxion supervisor:
Name: Taco van Loon
Email: t.vanloon@saxion.nl

Smart2IT supervisor:
Name: Gaston Smit
Email: smit@Smart2IT.nl

Abstract

This abstract contains a brief summary of the research paper created by the student, Bruce Boulton. The main topic discussed was how the environment could be improved for the company's product, Innobrix. Innobrix is a house configurator tool that is used to modify houses in a 3D scene. A client will provide the development team with a highly detailed house (configurable) which is then converted into a low detailed house manually (detailed volume). This is done so that the detailed volume will be loaded first in order for the scene to perform efficiently. It runs on a website using a new framework for rendering 3D applications called three.js. The main goals were for the Innobrix demo to be upgraded with a better-looking environment and contain new detailed volume houses, while maintaining a stable performance. Photogrammetry did not work as a viable method for producing assets. Shadows and reflections don't work well when they are included in the processing stages, as having them displayed incorrectly in the Innobrix scene reduced the realism aspect. Likewise, thin detailed assets did not have great results, as small details were excluded. Overall, photogrammetry would have taken up too much time in order to setup correctly and get the desired results. Repetitive textures were another challenge to overcome. Three different tests were provided in order to see which worked well. In the end, it was found that overlaying models over other assets produced the best results. For instance, placing some flowers over the grass texture diminished the repetitiveness efficiently. Shadows were also discovered to be a great source to break up the repetitive textures, therefore tall trees and other assets are placed in the scene to maximise the amount of shadows emitted. In terms of optimizing models and textures for Innobrix, a few different methods where used effectively. One of the main performance consumptions was the amount of draw calls in the scene. All models and textures add to the total draw call amount. Therefore, the fewer models and textures, the better the performance would be. Lowering the triangle counts, texture resolutions and combining similar textures together in the same UV space were also great methods to enhance the overall performance. Substance Painter has a glTF exporter built-in, which meant that suitable texture files were automatically created for the pipeline the company would use in the future. Lowering the dilation-padding of this exporter and changing the texture file types to jpg also lowered the sizes of the outputted files. Normals from Blender did not export well into Unity; therefore, it was advised to let Unity calculate the normals inside the model's individual settings. A PBR validate node was used for textures created in order to justify choices made in the design stages. Most textures ran at a resolution of 1024x1024, however since some models were extremely small in comparison to the scene, they could be downgraded to 512, or even 256.

It was discovered that the detailed volume houses in the scene drained the performance significantly due to their sheer amount of draw calls. These houses were already created by the team at Innobrix, not by the student. Therefore, the performance tests were done with the houses disabled in order to obtain results from what was created during the graduation project. The student highly recommended that the development team of Innobrix optimizes the houses in the scene more efficiently before they put them online. Combining each detailed volume house into a single model would enhance performance significantly. It will be a choice between having more customizable options, or better overall performance. Lastly, performances were recorded of the finished Innobrix scene with the environment the student created. Some devices performed considerably better than others, with Apple and Huawei leading the way for mobile devices. Tablets and laptops performed relatively poorly due to them using older hardware. All of the desktop computers performed well due to them having stronger hardware. One interesting point to note was that using a 144 Hz monitor would remove the three.js 60 FPS limit, meaning a higher framerate and thus smoother viewing experience. This would be beneficial for when presenting the product to new clients.

Table of Contents

1. Introduction

1.1 Company background

Smart2IT is an innovative software company in the Netherlands, which specializes in digital applications based on the latest visualization technologies. They develop practical and affordable SAAS solutions to facilitate the use of augmented reality, 3D visualizations and virtual reality. Smart2VR, eyeQ, and Innobrix are Smart2IT products. In total they have fifteen employees. The company's office is situated in Hengelo, Diamantstraat 3, 7554 TA, Overijssel. It was founded in 2014 by Sander Martinec, the sales director, and Gaston Smit, the chief technical officer (CTO). They have various construction companies as clients who use their products, such as Brummelhuis, Heembouw and Trebbe.

1.2 Product description

Innobrix is their own product that allows customers to configure interactive virtual homes inside a 3D environment. The company is currently developing the Innobrix application into a web-based platform using *three.js* in order to reach a wider public. Previously Innobrix was developed with Unity and built into a *WebGL* version. WebGL is a JavaScript API that's used for displaying graphics within any popular browser without the use of plugins. Since WebGL is mostly integrated with other web browsers, it allows for full usability of the GPU (Graphics Processing Unit) and post processing effects included into the web page (Tavares, 2012).

Under the direction of the CTO, Gaston Smit, the Unity WebGL version of Innobrix is being transformed into a three.js version. Three.js is also a JavaScript API used to create and show 3D graphics in a web browser and it even uses WebGL aspects. However, it is more customizable in terms of development. The house models are provided by the client's architectural construction companies. The development team at Smart2IT implements all the models into the client's custom Innobrix project, adding their own textures plus extra assets to the scene. The Smart2IT client is then provided with the final finished project so that their own customers can experience newly designed homes through desktop, mobile and/or virtual reality devices.

Some of the main features of Innobrix are listed below:

- Allows users to interactively customize and experience their own house inside a 1:1 scale ratio, further increasing the immersion.
- Users can see and adjust the costs of adding extra features to their houses.
- The date and time are adjustable for the entire year, meaning shadows and sun direction can be seen in the project at any realistic time.

Figure 1: Innobrix online home configurator setup.

A project team consisting of four employees was set up in order to help with certain aspects of the assignment. The employees worked with clients on their own projects, while the student worked on a new Innobrix high-quality demo web version. This high-quality demo version (HIFI, i.e. high fidelity) needed houses to be allocated and more environmental assets to be created. It was done in such a way that the final result would be visually appealing and interactive to the CTO and future clients. Attractive graphics have been proven to help users maintain interest and excitement towards an application (Rooney, 2012). The final result created in the student's five-month graduation project would be used to showcase the product at its highest potential.

The final product can be used on every acceptable platform such as; Microsoft Windows and Apple computers, smartphones (Samsung, Apple, HTC, and OnePlus), virtual reality (HTC Vive) and augmented reality devices (Microsoft HoloLens).

2. Reason (problem indication)

The purpose of this research report and the assignment was to create a realistic 3D environment for the company's house configuration product, Innobrix. The client - Smart2IT - wanted to further develop and implement new assets (textured models) into their recently developed web demo version of Innobrix in order to make it visually appealing to future customers. The first version of the Innobrix demo web version was not aesthetically acceptable.

This research is about the creation of optimized environment assets for the graduation project. Previously the student had worked with 3D modelling and texturing in past projects, specializing in realistic environment design. Since both parties' goals towards the assignment correlated positively, the student was given the task of creating a plan-of-action in order to construct a new environment.

Figure 2: The Innobrix online demo, lacking in environmental features.

The environment was not optimized and was visually bland. UV-mapping was skipped on some assets and there were many adjustments that had to be made. Later on - in the report - the performance of the environment is recorded, alongside the performance of the configurable houses. The environment created by the student had to be extremely lightweight because future use of the assets created would have to be useable in other projects run by the company.

At the time of writing this report the company was in a transition phase, moving from Unity to a fully online three.js framework of building Innobrix projects.

3. Preliminary problem statement

3.1 Problem analysis

The problem brought up by the client was reasonable within certain specifications, the two main constraints being quality and performance.

For instance, creating a fully realistic 2048x2048 textured resolution environment would never be able to run smoothly on mobile devices, since they are not powerful enough. As a result of this, the overall quality of the environment needed to be downscaled. This had drastic effects on immersion because the higher the quality, the more detail could be shown. Downscaling the resolution of textures removes the realistic aspect, especially when viewing the environment from up close. Textures play a major role in the performance of any program, but so does model polygon density (total amount of triangles). Testing needed to be done for both of the constraints mentioned above in order to find the optimal settings to balance quality and performance, bearing in mind the minimum target number of thirty FPS (frames per second) for each device platform.

Taking all of this into account, the environment needed to be created in such a way that it would match the rest of the assets already made in its visual style and that the performance would remain consistent throughout. The very first version of the Innobrix demo contained an environment that can be described as mediocre. The engine was only capable of using albedo (colour) texture maps, since other maps had not been implemented into the engine yet. Most of the assets had not been implemented and it was lacking in optimization. The company wanted the environment, especially, to be an attractive aspect of the product's overall appearance. It needed to captivate the user and be easy to navigate. The performance was one problem that could be dealt with later on. However, if enough testing of the three.js Innobrix configurator was done, then the limits could be determined. The optimisation of models and textures was done for the entire demo, including the work already created with the environment. It was tested towards the end of the graduation assignment. Based on the discussion above, it was decided to test the following scenarios and record the average data usage amounts:

a. (Low-range devices) average FPS, Memory and MS.
b. (Mid-range devices) average FPS, Memory and MS.
c. (High-range devices) average FPS, Memory and MS.

These variables are discussed more in-depth in the results chapter. Another issue that could arise would be the sheer number of models being added to the scene. This would significantly increase the amount of draw calls and therefore reduce the performance.

3.2 Conditions of Satisfaction (CoS)

The main question the client asked was: "How can the environment be improved and used for future projects?". The product they are currently developing and iterating on is called Innobrix. The parties concerned were the employees at Smart2IT, CTO, Saxion graduation coach and the student completing his graduation assignment. The client wanted the finished work and assets the student created to be accessible from an organized library. The project still needed some basic ground-rules in order to achieve the Conditions of Satisfaction between the company and the student, i.e.:

- Create a planning document (asset list) that, once approved, would be followed by the student.
- Do research on optimizing assets for more efficient performances.
- Work with employees from the company in order to get models and textures implemented into the Innobrix demo scene.

- Change already created models and textures only if needed, remake assets only if absolutely necessary.
- Design a visually appealing environment to suit the client's and customer's needs.
- Performance must be optimal for all devices, aim for 30 FPS (frames per second).
- The asset list must be updated and followed, with the student constantly receiving and acting on feedback from the client and project team.
- Aim for the environment to look realistic in the same style as the rest of the project.
- The student must create a library of assets that can be easily implemented into current and future projects.
- The end result of the graduation project is an Innobrix web version which looks good and performs well.
- A weekly graduation update document was created in order to inform all parties of the work done; it can be viewed at **Annex 11**: https://docs.google.com/document/d/1EiEl5L0s44tg05KTGan3JcRNoX3McHjVhDNmBQ7rA3U/edit?usp=sharing

4. Theoretical framework

4.1 Information behind three.js, WebGL, and Unity

WebGL has amazing possibilities, but also some disadvantages. Microsoft has deemed it too vulnerable in terms of a security threat to projects and have disabled most WebGL/ Three.js websites in order for them not to load (Nazarov & Galletly, 2013). The 'Innobrix' configurator therefore doesn't work on Internet Explorer browsers. Three.js, as described before, is based on the same JavaScript API framework as WebGL, however it is more customizable in terms of development.

An article was discovered that made an excellent comparison between Unity and Three.js after working on a project with both. The table below contains a summary of the differences:

Unity WebGL:	**Three.js:**
Designed to create heavy GPU and CPU applications; game and installations.	Designed to create light-weight applications; creative storytelling, real-time animations and data visuals.
VR ready.	Not suitable for VR yet.
Great Unity API documentation.	Lacks updated references, constantly being iterated in development.

(Wu, 2016)

In the last few years, a lot has been refined in terms of JavaScript physics and rendering applications, however there is still a great deal of improvement to be made. Very complex and high-quality 3D scenes may not yet be feasible. As pointed out in a research paper, the main advantage of this technology is that there is no need to install or download anything for viewing a high-quality 3D application (Nazarov & Galletly, 2013). The main issues of performance cost are: high draw calls, complicated resource-heavy shader code and post-processing. Post-processing would not be used in the project due to its high-performance cost. This also goes for shaders, as only the standard ones are needed in the early development stages of three.js. High draw call amounts can be solved combining models instead of having individual meshes and materials.

For this project, the fbx (Filmbox) file format would be used to export models to Unity first, because the Unity version currently doesn't support glb or glTF files (GL Transmission Format). The company is not working with Unity WebGL anymore due to the slow loading times and performance issues with previous projects. They are now using Unity just for exporting their projects to three.js.

Fbx files would be discontinued when the company finally moves over to Blender to export projects. This is because fbx files, while being an industry standard, are not customizable in terms of reading their file properties. Autodesk has locked any fbx property reading in order to maintain their own private file type for legal reasons, limiting the potential. The glTF and glb file formats are open source and highly customizable. This is beneficial to the project, since the company is moving over to three.js, which has the same development standards.

4.2 Aspects of digital 3D property planning and configuration tools

The Innobrix site is intended to be used by users wishing to customize their houses in a near-realistic scaling system. A similar program that has the same function would be the Sims (https://www.ea.com/games/the-sims). This was therefore used as a reference on certain elements regarding the environment design. For instance, hedges in the game were observed and used as a reference to see how these models were designed.

Figure 3: The Sims 4 in-game screenshot, (Kolenberg, 2014)

Creating hedges and fences that are modular and buildable in the Innobrix engine would be a significant part of the project. Creating correctly scaled assets that match the size of the scene would be crucial in order to achieve maximum efficiency and keep the realism aspect. Models that are not scaled correctly would visually remove the realistic immersion. Every modular asset needed to fit together seamlessly to preserve the immersion of the user. The pivot points needed to be facing the same direction once exported, in order to maintain using consistent placement system (Mader, 2005). Blender was chosen as the main 3D program because it would be used in the next version of 'Innobrix' and Unity would be ditched altogether, since it would no longer be needed to position models. Blender is also a free open-source 3D modelling engine; therefore, no licensing would be required. This is explained more in-depth in section *4.5.a)*.

4.3 Performance and optimisation for a wide range of devices.

According to the research report '*Optimized Graphics for Handheld Real-time CG Applications'*, a few different approaches were found in order to optimize the asset creation workflow, i.e.:

a) **Model and texture priority** – the assets most seen should be the most detailed in triangle count and texture resolution.

b) **Lower amount of draw calls** – Models that have high triangle counts and that are repeated throughout the scene should be combined into a single model, instead of containing individual meshes.
c) **Use a LOD (Level of Detail) system** - Optimize the assets polygon count for each device. This means that on PC devices the models should load into the scene with higher triangle counts as compared to mobile devices. Since the environment is not the main priority, focus on creating assets as low detailed as reasonably possible.
d) **Compress textures** - Optimize UV space by combining similar materials used by multiple assets into one material that they all share. Smaller assets can have smaller texture resolutions as long as the outcome does not ruin the realism aspect.

Adapting the triangle count will benefit the GPU by lowering workload. Decreasing the amount of draw calls will make the application run smoother. Compressing texture resolutions will increase loading performance by lowering the amount of data needed (Powell, 2014).

Compressing textures from multiple materials into one would hugely benefit the scene in terms of optimisation. The quality might decrease slightly as the UV maps' space would fill up, but since the models in the environment are not prioritized as the most important in the scene, their downgraded quality would not be an issue. It has been proven that using a texture atlas for the process of combining multiple textures reduces the number of HTTP requests for texture images (Cozzi & Riccio, 2012). At the end of the assignment, the results were concluded from a wide range of selected devices. These devices were chosen based on what was available at the company and the trend of the current popular brands (Oxborrow, 2018).

4.5 Project setup overview

This section is used to back up each phase of the graduation project in order to justify choices and also state limitations. Many of the workflows discussed below can be viewed from their original source diagrams, attached in **Annex 3A**.

a.) Modelling

Blender, UV mapping

Modelling program

Every modelling program has their own specific settings, layouts, hotkeys and special features to provide their products with unique selling points. The fundamental outcome is that the final results are the same no matter which program is used. The user's level of experience with the wide range of programs will determine what is used in the workflow. This doesn't apply to studios that use a universal workflow for all of their employees.

Within the company it was decided that Blender (https://www.blender.org/) would be used as the main 3D modelling program for two reasons. Firstly, it is a free to use open-source licensed program and secondly, there are many useful community created add-ons for Blender which extend its potential even further. Using the file formats glb and glTF allows the company developers to create new plugins in Blender, in order to make better workflows and allow further customization in relation to the new three.js Innobrix version.

Modelling techniques

Since the Innobrix application is intended to run on every platform, models were created as efficiently as possible. The models in the environment are not meant to be viewed from nearby, since the primary focus would be on the houses. This means that assets are created first in a high poly version to be used for small details and then subsequently decimated to a low polygon count version to be used in the game engine. The reason for this is that the high polygon model would be used to bake onto the low polygon model inside Substance Painter later on. After experimentation, it was

determined that this only worked on assets similar to rocks, since their designs and patterns are unique. Having an asset such as a telephone pole with lots of small details made it stand out evidently, since the rest of the houses in the scene are clean and not finely detailed. One of the main mistakes that artists make is that they create 'N-Gons', which is basically a face with more than four vertices. This is not ideal in every engine because quad faces (four-sided triangles) are converted into triangular faces (three-sided triangles) and if the face has more than four vertices then the engine will not render the 'N-Gon' face correctly, causing issues when viewing the asset. The sizes of the models inside Blender are determined using a block scaled to be the size of a standard human, around 1.81 meters and using reference photos (Wobma & Bruggink, 2012). Sizes of certain models, like traffic signs, could be derived from online community documents. Faces underneath models which are not seen can be deleted to enhance performance. The less triangles in the models, the higher the performance (Powell, 2014). This was one of the goals in the Conditions of Satisfaction chapter. The website and environment model provider 'SpeedTree' (https://store.speedtree.com/store/italian-cypress/) provided an industry standard overview of mobile, desktop and developer polygon counts for environmental assets.

UV Mapping

A UV map is the bond between textures and models. It is the mapped and flattened topology of the model, as well as the groundwork for texture map baking (Bech-Yagher, 2018). This means that good UV mapping is absolutely crucial for every model. Errors in UVs provide bad texture bakes. Based on a 0 to 1 grid, with 0.5 as the middle coordinate, a UV map contains the 3D model's XYZ coordinates flattened out into a 2D space, also known as a tile. This tile is what the texture is overlaid on and then wrapped around eventually. A stretched UV means the texture would also be stretched. A UV map with overlaying islands (sections of the UV map), means the texture would overlay at certain parts, causing incorrect material placements with texturing later on. These two common mistakes must be avoided.

To get an understanding of how the developers at the company went about doing this, some of the models previously created were checked. The finding was that the models were created as simply as possible and UV mapping is sometimes not even done and/or using islands outside of the tile range. This was surprising and it was taken into observation as we were taught to keep UVs restricted to one tile and not overlap. It seemed that the final visual result would be more important than how it was constructed. Knowing this, the process was sped up by using an in-built feature. Blender has a 'Smart UV Project' tool which automatically arranges the UV map with user specific settings. An efficiently placed UV map with small margins in between islands meant the texturing phase of the project could be completed accurately.

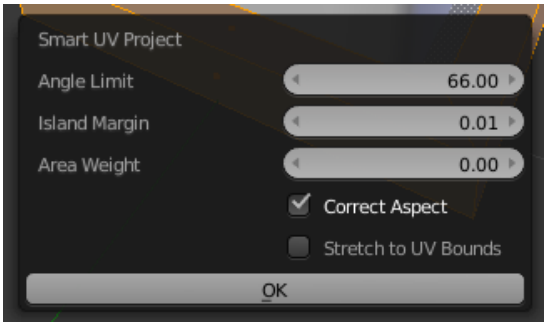


Figure 4: Blender Smart UV project settings for efficient UVs in the project.

The models are first cut with the 'Sharp edge' tool in order to define where the seams would be cut on the model. After the UVs were projected automatically, the student looked over each island individually and improved them if needed. Most of the time, the UVs would have been done

sufficiently, but occasionally certain parts of the model needed to be re-grouped and UV mapped manually.

Exporting the model

Once the model is completed, it is positioned at the coordinates (x-0, y-0, z-0), also known as the centre. Once the model is in position, then Blender has a function called: 'Apply the location', 'rotation', and 'scale' to the model data. This resets all transformation data when saving, in order to make the model accurately positioned, scaled and rotated once imported into the engine. In the export menu, the file format FBX is used currently at the company. Blender once again needs specific user settings in order to get the correct version in the engine. The *'!EXPERIMENTAL! Apply Transform'* option needed to be enabled and the 'Apply scale' setting needed to be set to 'FBX All', as having a different setting would mean the export process from Substance Painter to three.js would not work correctly.

b.) Texturing

Substance Painter, Substance Designer

Below you can see an illustration of the overall workflow used as a basis for creating assets:

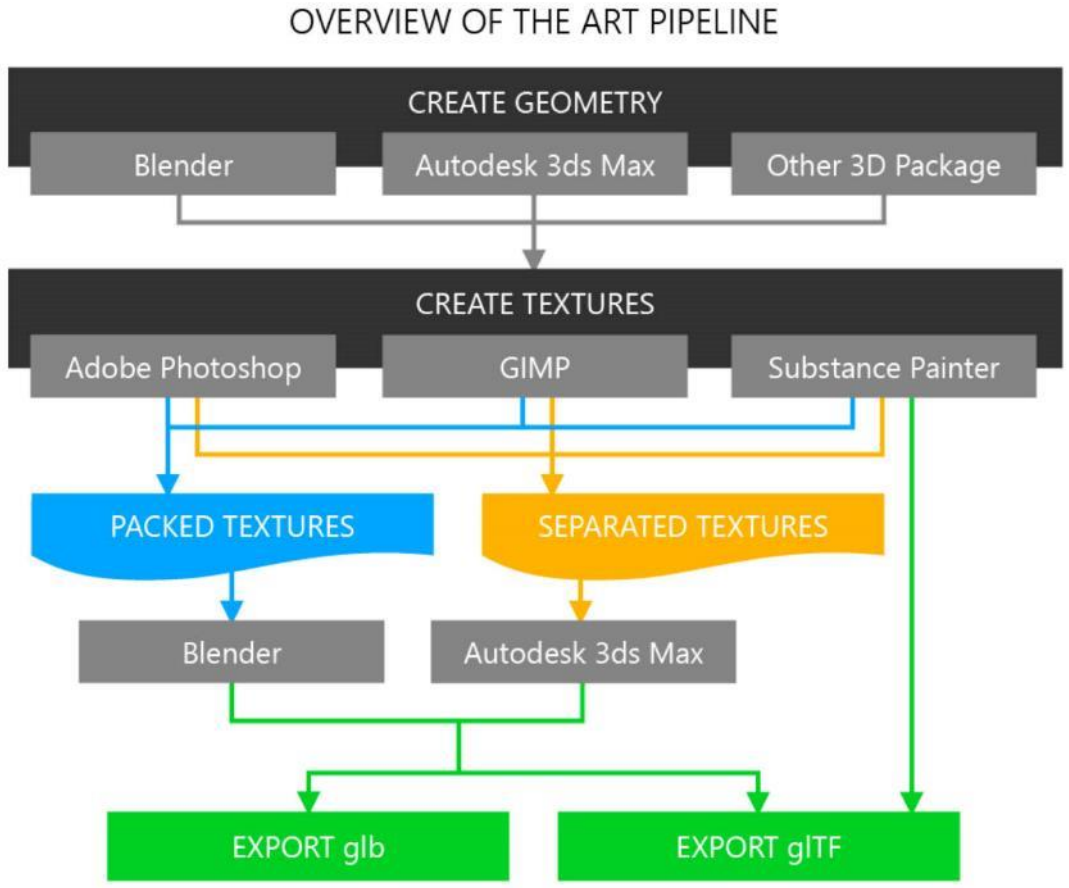


Image taken from the creators of the glTF and glb file formats (Ryan, 2018).

PBR (Physically based rendering) workflow

The Substance suite (https://www.allegorithmic.com/substance) is one of the industry's favourites when it comes to game creation. It was used to create the award-winning game Horizon Zero Dawn (Bosset, 2017). It is also used widely in architectural design since it is becoming more well-known and has partnered with firms such as Arte Factory Lab and HNTB (Sandrik, 2018). A lot of what makes a PBR workflow different from its predecessors is a more detailed calculation of the behaviour of light, surfaces and reflection. (Russell, 2015)

Benefits of using a PBR workflow:

1. PBR removes the unpredictable aspect of creating materials, such as realistic reflections, since its code and design are based on physically accurate calculations.
2. Assets will look realistic in all lighting set-ups.
3. It allows for a workflow to produce high quality textures, even between various artists and designers working on the same project.

(McDermott, 2018)

Simply put, a PBR workflow produces high quality realistic textures. It does this by creating maps that interact within the correct shader settings. The maps generally created are; Albedo, Normal, Metallic, Roughness and Height.

In order to confirm that the PBR workflow was being followed correctly, a PBR validate node will be attached to each material created. This node will justify that the materials created are confirmed to be PBR acceptable. Substance Painter was used to texture an entire individual assets and Substance Designer was used for procedural textures, such as road patterns, pavements and fences. It was chosen because Substance Designer produces seamless textures. A seamless texture means it can be placed right next to a duplicate and the seam of where the texture starts or ends cannot be seen. Sometimes a texture was needed that was not available in Substance Painter and then Substance Designer was used to create the desired texture. Substance Painter supports texture resolutions of 4096, 2048, 1024, 512, 256 and 128. The program can also easily and dynamically scale textures to any size at any time. The final results will be presented to the Innobrix team, to enable them to choose what they want to use.

Baking texture maps

Only Albedo maps are used in the current state of the Innobrix three.js version. All the maps in a standard PBR project will still be created during the graduation assignment, in case the project is eventually extended in the future. The project uses a standard resolution of 1024x1024, in order to maintain a base texture quality with regards to the rest of the scene. Higher texture resolutions will be tested later on to see the possibilities. Of course, a more optimized three.js engine will allow for a higher quality basis.

Texturing UV maps

Neatly organised layers and file structuring will be used throughout the project in order to provide a framework for future development. The very first few layers contain the base colours which are placed in a folder. Another folder is created which contains smaller details, such as scratches, rust and grunge. The Substance files are all saved and named accordingly, this allows for other employees to easily and efficiently make changes. Photoshop was used to texture certain assets. This was done in order for other employees of the company to work on the generic assets once the assignment was completed. An example would be the signposts, where each symbol on it can be enabled and disabled through layers in Photoshop to allow employees to make quick iterations.

Exporting the texture maps

Substance Painter has a custom export configuration for getting the correct textures for glTF and glb 3D models. Since these file formats are required for three.js, the student used this custom configuration. The exported maps were then named efficiently in the material and texture map name. In total, three maps were created for the project, i.e. the Normal map, Albedo and a combined map called the Occlusion, Roughness, Metallic map (or ‘occlusionRoughnessMetallic’). This is the same structure as used by the studio Dice, which produced the Battlefield franchise (Zakrisson, 2019). As mentioned before, these maps are not being used yet by the engine, but the reason for exporting these extra maps would be in order to future-proof the work done for later use. The maps are created at a 2048 resolution setting, since this is the highest quality the engine can handle. The maps are then

exported at a resolution of 1024 for the project, because they can be downscaled easily inside Unity to lower resolutions, thus decreasing the file sizes and improving the performance.

Figure 5: Texture maps created by using Substance Painters glTF export template.

c.) Implementation

Unity, GLTF export, exported projects, FileZilla server import

To start off, models are imported into the Unity project and placed into a structured folder layout. The materials are applied with their corresponding textures created in the texturing phase. The model with textures, now called an asset, is placed into the scene in a logical position, in line with the reference photos. Since the landscape '*kavelkaart*' was already created, the student had to place the assets into the scene. The *kavelkaart* itself needed quite a lot of optimisation since each group of models it contained had multiple materials. Once the layout was combined and the scene ready, a prefab of the entire environment was then created. This was done in order to easily distinguish what was done during the graduation assignment and what had already been completed.

The prefab was then added to custom 'GLTF Exporter' tool created by the company, which converts the scene into a format that was used by three.js. GlTF is a file format through which 3D assets are created by using JSON files. The *glTF* files are uploaded to the online Innobrix demo server via an FTP (File Transfer Protocol) program called FileZilla. The final product was then finally viewable online at https://configurator.innobrix.nl/hifi/. The files are uploaded to the test server, by adding '-test' next to 'configurator'. Once the files are checked, the project is moved to the production server. Since the engine was due to be getting upgraded, the graduation work done was future-proofed by including all the necessary textures that cannot be used yet, but perhaps at a later stage. These extra unused textures will tremendously increase the realism aspect of the assets. Photogrammetry was investigated in order to see if it would be a viable solution for creating assets, as it could possibly speed up the entire companys' workflow.

At one point during the project, the student decided to speed up the development process by adjusting a certain script in the exporter in order for the correct name to be used when exporting. Until that time, the exporter had created the incorrect names, but with the adjustments made, the script began to export the correct names, resulting in an easier workflow. This adjustment was done while under the guidance of an engineer at the company. A screenshot of the code can be seen in **Annex 2**.

e.) Configurable houses

Terms used by the company:

Detailed volume: The first house loaded into the scene, which is low in polycount for optimal performance.

Configurable house: Once a detailed volume is selected, the configurable house will load in its place. It is simply a high polycount house with various customizable options for the user.

Option list: A list of options every configurable house contains which is used to control what is enabled or disabled.

The company showcases houses that can be configured to whatever the user wants within the specifications of that certain house type, inside of a 3D environment. The process in order to get a

configurable house online inside the Innobrix scene is relatively long and not always straightforward. Firstly, a configurable house was imported with the help of another colleague at the company. The prefabs were then created and placed in the correct directory inside Unity. Scripts and details were attached to the prefabs to make sure that the configurable house worked correctly. Some configurables were mirrored, since the houses are under the same roof, so the mirrored versions also needed to be constructed from the beginning. The configurable houses were too performance heavy to be loaded directly into the scene, so detailed volume houses were placed in first. The detailed volumes are deconstructed versions of the configurables with all the unnecessary inside detail removed using Blender, in order to lower the polycount of the house. This is done to allow the houses to appear in the scene and run optimally. In total, three new configurable houses and detailed volumes were created for the scene. Once the configurable houses and detailed volumes were linked in the scene and looked alright inside Unity, they then needed to be exported.

Using a custom tool created by the company, the project was exported to the server on which the Innobrix scene files are stored. The files needed to be updated manually and the entire process of getting them all updated occasionally took up to 40 minutes, per export build. There were many times when one small issue meant the entire process had to be re-done. This workflow was not efficient; however, the company was in the process of removing Unity from the pipeline and implementing their own engine into the online three.js Innobrix version. This meant everything could eventually be built and constructed online with no need for third-party programs.

5. Definition of the problem

5.1 Main research question

Based on the client's requirements and the preliminary work done, the main research question was determined:

> How to enhance the visual aspect of the application 'Innobrix' keeping the target platforms' limitations in mind?

This was chosen to be the main question because it includes the main goals of everyone involved. Firstly, for a better looking environment to be created and secondly, for it to be made in such a way that the performances on other devices won't be affected negatively.

5.2 Sub-questions

In order to answer the main question stated above, the student had to resolve the following sub-questions. They are formulated on the basis of general feedback and the Conditions of Satisfaction:

a) Is 3D photogrammetry viable?

Photogrammetry is a relatively new method of creating assets that could possibly boost the workflow speed efficiency and production quality in the company. Base tests where completed in order to see if it would be a viable alternative to standard 3D modelling practices.

b) How can repetitive textures be fixed?

This was considered to be a major issue by the company. Using research and practical knowledge, a few methods for fixing repetitiveness in the Innobrix environment were explored and solutions were provided.

c) How can models and textures be optimized to improve the performance of Innobrix?

Innobrix is built on the three.js platform which is not widely used for creating environments. Assets needed to be highly optimized while maintaining the same style as the rest of the scene.

d) What are the performance results of the final Innobrix version on each device platform?

Different devices were tested in order to see anomalies and make justifications for the final product. The performance results will prove whether the research and optimisation methods were successful.

6. Scope

The scope of the student's graduation project was discussed thoroughly with the company supervisor. It included creating a sufficient work plan based on an asset deliverable list for the Innobrix web version. Roughly sixty assets needed to be modelled and textured, although the value could change depending on interruptions in the work schedule. The student worked with other employees, i.e.; two other artists and two technical engineers. The other members mentioned were only around to help the student if needed, otherwise they were engaged in other projects. The primary role was to fix any assets if needed, create new ones and be in charge of the overall environment scene design. The total timeframe would be the duration of the student's five-month graduation project.

7. Method of graduation project

Literature, iterative work and testing were used as the method of working. Challenges and issues were noted and solutions or alternatives were supplied to enable the company to continue working on the project efficiently. An asset list containing all the assets created was provided, containing the number of models, triangle counts and extra notes for each model. This was also a Condition of Satisfaction.

The structure of working was to begin with the first sub-question concerning the use of photogrammetry to create 3D assets. Two different programs were used to compare two different models. Based on base tests and their results, it was decided either to continue using standard methods of 3D modelling or use photogrammetry if it turned out to be the better option. The second sub-question was a request from the company CTO for a major issue to be fixed; repetitiveness in textures.

The methods tested were: use Substance Painter to manually paint over the landscape model, try a custom shader to fix the repetitive issue and the final method was to overlay certain models on top of other models. The third sub question focused on the theoretical framework section of optimisation. A few scenarios that arose during the graduation assignment are described, such as: creating flower and plant assets and the issues involved.

In the final sub-question, the student compared various devices and debugged various statistics of their performance running the final Innobrix version which the student produced. Their respective GPUs, RAM allocations, and CPUs specifications will be stated in **Annex 10**.

The results depicted the student's skills in using literature and iterative work. A couple of challenges presented themselves and were addressed in the 'Graduation Results' chapter. A schedule and asset creation pipeline were created and approved by the company supervisor (**Annex 7**). These were followed as a guideline for the duration of the graduation project. The most important aspects of the graduation assignment are highlighted below, i.e.:

1.) Create a library of assets that the company can access easily.

2.) Implement the assets into the Innobrix web scene.
3.) Make the overall performance optimal.

Taking into consideration the goals mentioned above, the student created two diagrams to illustrate the working structure, attached in **Annex 7**: *Schedule and asset creation workflow*. In the analysis phase the student was introduced to the company and was given the task of researching the business itself and information behind the software. The assignment task was properly formatted and the implementation plan was drafted, while being adjusted to the company and Saxion coaches' feedback. In the production phase the student worked in four-week periods for releasing new assets into the Innobrix configurator. The student used the asset creation workflow to create the assets in an orderly manner. Each asset can be found in **Annex 6**: *List of deliverables* and **Annex 8**: *Asset list*. In the evaluation phase the student evaluated and formulated his conclusions about the work completed. Recommendations were attached to the relevant chapter near the end of this report. The asset list also contained notes in order for both coaches to note updates and other important information.

Each stage was checked after completion to make sure it aligned with the overall assignment so that there wouldn't be any issues in the future. The company coach and student discussed each aspect and constructive criticism was applied to the assets. Research consisted of collecting useful resources which could be applied to the theoretical framework and project setup overview. The student went out occasionally to take original photographs of items on the asset list for reference later on in the modelling and texturing phases, as well for the implementation stage. This was done since assets needed to be logically placed. Assets were created by modelling them on the basis of the asset list. Once they were modelled, they were then UV mapped efficiently. Finally, the models were textured in line with the reference photos. This was used to clarify later on why certain colour schemes were chosen.

Lastly once the assets were approved, they were then implemented in the 'Innobrix' online scene. The reference photos were used as a base to logically place the assets. Of course, changes to the placement of assets were made as a result of feedback from all parties involved. The specifications of the selected devices are provided in **Annex 10**. The goal of doing these tests was to determine if the work done did indeed optimize the overall Innobrix performances. Once the tests were done, the model amounts, polygon limits and texture resolutions could be justified for later on.

The data results collected were filled into tables and the results were placed in the corresponding sub-questions regarding model and texture optimisation. It was not possible to optimize the three.js engine of the company in any significant way, because it was in its development stages. There was very little information available in terms of constructing an entire environment using three.js. This definitely affected the progress of work, since the scene being created only got greater in size over time and - with not many resources being available - only standard practices were used in order to optimize.

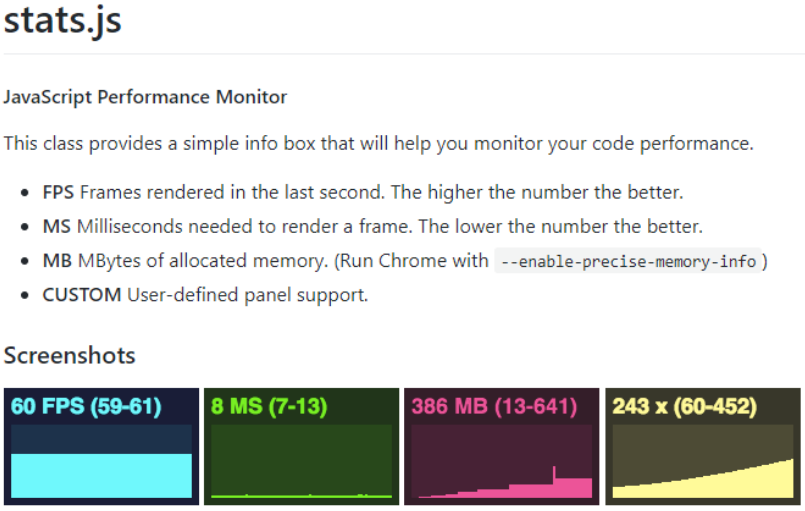
Towards the end of the assignment, in the last two months, an updated schedule was drafted based on the new knowledge the student had obtained (attached to **Annex 9**: *May June graduation plan*).

8. Graduation results

8.1 Answering the main question

How to enhance the visual aspect of the application ‘Innobrix’ keeping the target platforms’ limitations in mind?

The student completed tests in order to justify model density and texture resolution impact on performance. The goals were to obtain a stable 60 FPS, memory usage below 100 MB, and frame rendering speed to be below 15 MS. The definitions can be seen below, taken from the official three.js information page:



Screenshot taken from https://github.com/mrdoob/stats.js/

A baseline test was done in order to determine the capabilities of the three.js engine. A developer’s computer was chosen (this was the computer used during the graduation assignment), because if it was not possible to run the base tests, then they would be unlikely to work on almost any other device. Simply put, the developer’s computer was considered the standard test. Every time a device is mentioned (e.g. developer computer), the specifications can be found in **Annex 10**.

BASE TEST: TEXTURE SIZES AND MODEL DENSITY IN THREE.JS USING A DEVELOPER’S COMPUTER

	FPS (Frames per second)	MB (Memory)	MS (Milliseconds)
128 texture & 12 triangles	60 (stable)	28 - 34	0
128 texture & 768 triangles	60 (stable)	150 - 157	0
128 texture & 49152 triangles	60 (stable)	223 - 228	0
1024 texture & 12 triangles	60 (stable)	52 – 58	0
1024 texture & 768 triangles	60 (stable)	174 - 181	0
1024 texture & 49152 triangles	60 (stable)	259 - 255	0 - 1
2048 texture & 12 triangles	60 (stable)	77 – 85	0
2048 texture & 768 triangles	60 (stable)	199 - 205	0
2048 texture & 49152 triangles	58 – 60	262 - 281	0 - 2

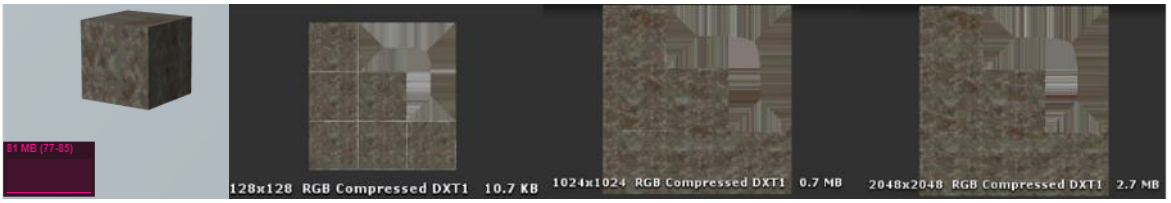


Figure 6: Testing in an empty scene with a simple model and texture.

The results show that the memory variable has the biggest variance in sizes and therefore, could not be defined as reliable information. Models seem to affect FPS, while texture sizes affect the memory, which is logical. Since the results above proved to be quite bland, a few smaller tests were done to further debug the scenes performance:

TESTING 1024 TEXTURE & 768 TRIANGLES USING A DEVELOPER'S COMPUTER

	FPS	MB	MS
10 Models	60 (stable)	298 - 303	0
100 Models	60 (stable)	323 - 350	0
500 Models	60 (stable)	474 - 495	1 - 2

This test shows that model amounts don't affect FPS significantly if they don't contain sub-meshes. However, having many models with lots of materials attached to sub-meshes would definitely impact the FPS negatively. The main issue that arose from the two tests above was having a high-density model. The scene will only cause poor performance if the user goes close to a model that is high in triangle count (above 49152) and contains many materials/textures (around 3+). The amount of materials also adds to the total draw call amount in the scene. As stated in the theoretical framework research: the lower the draw calls, the better the performance will be.

OLD INNOBRIX DEMO PROJECT BENCHMARK, TESTED ON DEVELOPER'S COMPUTER

	FPS	MB	MS
Landscape (*Kavelkaart*)	60 (stable)	56	0-1
Water	60 (stable)	80	0-1
Houses (Detailed volumes)	58 – 60	180 - 248	9 - 12
Houses, water, landscape (everything)	52 - 60	195 - 357	11 - 13

This proves that the base Innobrix demo version which the student was given - with the *kavelkaart*, water and houses - already had issues with performance. This meant that even if the student would have optimized their work, the actual engine was just not performance-friendly at the time of writing this report. A few options are provided in the recommendations chapter. This already leads to the conclusion that the final performance results will be inadequate unless the development team of Innobrix optimizes the three.js engine. Two detailed volumes contained too many draw calls for most devices to handle. Desktop computers could handle around 9 to 15 detailed volumes before the performance started to drop significantly.

The texture and model optimisations were then applied based on the research. The textures were adjusted in resolution based on their viewing distance. Models not seen from nearby have resolutions of around 512 or even 256, instead of the normal 1024 resolution. This was done in order to lower the projects total size, thus decreasing the loading times and increasing the frame rate. The improved results can be seen in the next test:

RESULTS AFTER OPTIMIZING ASSETS IN NEW INNOBRIX DEMO, TESTED ON DEVELOPER'S COMPUTER

	FPS	MB	MS
Houses, water, landscape (everything)	58 - 60 (stable)	42 - 170	10 - 14

After all of the optimisations the student applied, the FPS increased by roughly 6 frames and the performance became stable. The memory also improved dramatically, it went from a 276 MB average, to a 106 MB average. Lowering textures which were not seen up close from 1024 to around 512 substantially decreased the loading times.

However, it was noted that having too many draw calls in the scene would eventually cause the MS variable to spike and therefore cause mobile devices to crash. This occurred when the MS reached around 18 to 22. This is not an issue of the student's work, but is caused by the assets already implemented, such as the detailed volumes and car assets. Once the detailed volume houses are optimized the scene will run smoothly. Another quick test was done and the maximum number of detailed volume houses in the scene was determined to be around 10, if the project was aiming for a 60 FPS goal on desktop computers.

The final Innobrix project had 24 configurable houses. The company supervisor was notified about the issues regarding the amount of configurables and after consultation with the student, gave the all-clear to continue working on the project, despite the performance loss. Since the Innobrix web version was being upgraded in the next few months, the issues encountered would most likely not reoccur in the future pipeline.

The issue with the detailed volumes was discovered as a result of testing. With 17 detailed volumes, the scene lost roughly 50% of its performance. This is illustrated below:

Figure 7: Without detailed volumes (left), with detailed volumes (right).

After testing, 10 – 14 detailed volume houses turned out to be the maximum that could be added to the scene before the performance dropped substantially on desktop computers. The environment had roughly 5% impact on the performance, since the assets created were optimized. This can be seen on the testing server, with 0 detailed volumes and only the environment enabled. This is further discussed in the recommendations section, with options provided. Further testing needs to be done by the company as this performance cost was a major issue.

After extensive testing the student selected a few of the assets created and analysed the process. To create the flower assets, the student took pictures of flowers from around the neighbourhood as reference photos and cropped out the necessary parts using Photoshop in order for them to be placed on planes inside Blender. They could then be exported and placed into the scene. All the textures mentioned in the next few paragraphs were created during the graduation assignment.

Figure 8: In order: reference photo, Photoshop work, plane construction in blender, final result.

Flowers and grass assets in typical game engines are optimized to be planes that are vertically positioned in a small patch as illustrated above *Figure 9, final result*. However, this did not work for the project. After testing, it was discovered that the result was not visually appealing. This was because double-sided rendering was not enabled in the engine.

Double-sided rendering shows both sides of a material on a plane instead of hiding the side of a plane that is not facing the camera. It gives the effect of having more visible sides in order to fill up the area and hide gaps. The cut-out shader used on the materials also needed to have anti-aliasing enabled to smooth out the rough edges.

To counter this, the student created the planes as flat as possible in order for the camera to not see the other side of the plane and hide it. In other words, the planes are facing up towards the viewer.

Figure 9: To fix this new problem, the student created some 'box pots' for the flowers to fit into.

This bought up a new issue. The flat planes didn't fit well into the scene since they now stood out even more than before. This would not be the best asset in the scene and would be used rarely in order for it to not stand out. The only way to fix this was to make the flowers viewable from a top-down view with no side angles. The result was then accepted by the company coach and the student could move on to creating other assets.

The next task that was problematic, was creating the new hedge '*beukenhagen*' (common Dutch hedges). The hedges that were used previously contained 1774 triangles, which was too many and it also did not look visually appealing.

Figure 10: In order; Old hedge, second hedge and final hedge.

The original hedges were too rectangular and looked fake. With the second attempt, the hedge UV seams of the model could be seen easily and this ruined the visual immersion. The student aimed for 300 triangles and kept experimenting. In the end, after extensive testing, the student created a realistic looking hedge containing 328 triangles.

In order to achieve this low triangle count, the student used a method that was discovered by experimentation. The Sims, as stated in the theoretical framework, was used as a reference with regard to hedge design.

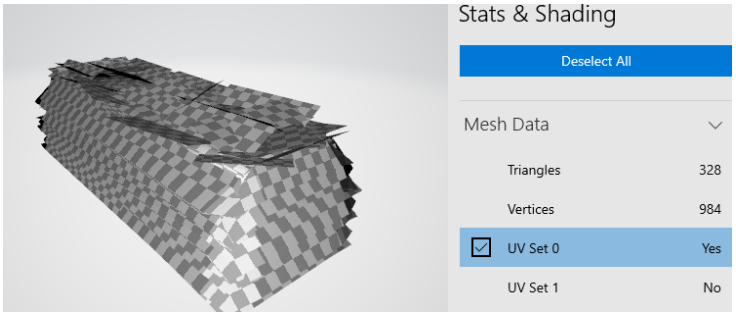


Figure 11: Detailed look at the final hedge result, showing the optimized UV mapping.

The UVs of the hedge were lowered to a 45-degree angle, in order for the camera to not see the seams. The vertices cut around the outside of the hedge perimeter, which decreased the overall number of polygons needed and thus improved the performance. The decrease in triangle count while improving the visuals of the asset meant this method was a success. The final result can be seen in Chapter 11.

8.2 Answering the sub questions

a) Is 3D photogrammetry viable?

This is a valid question because many of the top triple-A game studios use this method to create realistic assets for environments, such as Dice in Star Wars: Battlefront (Lievendag, 2016). In order to do a simple base test to see if it would work and be worthwhile to pursue, a few programs were considered to be best suited in order to create the models from photographs, i.e.:

1.) 3DF Zephyr Free

2.) Meshroom (from AliceVision)

After following tutorials on setting up the program and gathering suitable 360-degree photographs, the results can be seen in **Annex 4**. Photographs were taken of two assets in order to compare the differences. From what can be seen in the base tests results, it did not turn out well. Small sections and details were not processed efficiently when using photogrammetry. Further in-depth testing

needs to be done to determine if it is suitable for larger models. Another problem with photogrammetry was lighting conditions. Luckily, the Netherlands is a relatively cloudy country, which is beneficial in this scenario because it removes the shadows. This is a requirement due to the fact that shadows in Innobrix are updated in real-time and constantly changing. Having shadows baked onto a texture inside the environment would give an unrealistic impression. This also applies to reflective assets, as seeing the incorrect reflections and shadows on the texture would take users out of the immersive aspect (Lindquist, 2015).

It might be viable for natural assets such as rocks, because they are large models which don't have any small gaps or high reflective values. The priority of the graduation assignment would be to create models for the environment. This sub-question was calculated to take up too much time and since the base tests failed, it was discontinued. Overall, the results were insufficient in order to be used for the any project further.

b) How can repetitive textures be fixed?

A few options were noted during iteration in order to solve this sub-question:

1. Paint over the model's texture in Substance Painter.
2. Use a custom shader to fix the repetitive issue.
3. Overlay other models on top of the landscape.

Option 1 was attempted, but the results were not of an acceptable standard because textures can only be repeated so many times in a single UV space until they become blurry and unusable. The next option, number 2, worked well inside Unity, but since the company is moving to three.js, a custom shader will need to be created specifically to solve this issue. The name of the shader used is a procedural stochastic shader and more information can be found about it on the official Unity website found in the references (Deliot & Heitz, 2019).

Lastly, the final option for breaking the repetitive pattern was for models to be placed above and over the repetitive areas.

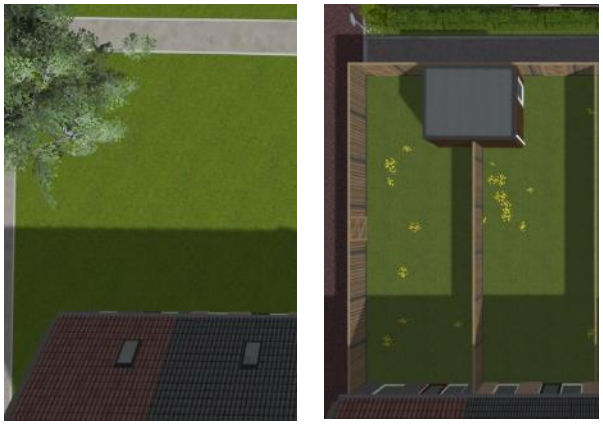


Figure 12: Before overlaying models and after.

This turned out to be the best option and it was also discovered that shadows played a huge role. Therefore, the creation of large assets was prioritized with regard to solving this question. Further down the line, flowers turned out to be a positive addition to remove repetition from the ground texture. This was tested and the results definitely proved worthwhile as seen in the image above.

c) How can models and textures be optimized to improve the performance of Innobrix?

The limits of Innobrix were discovered after testing assets in the pipeline and implementing them in the engine. This is because three.js is a relatively unknown framework with limited resources when it comes to constructing environments, therefore trial and error was the best solution at the time.

Based on the theoretical framework and the student's own knowledge, a few methods were selected in order to optimize the assets. Firstly, the model creation process was done by making the models have as few polygons as possible while maintaining realism from a distance. Unnecessary geometry that could not be seen was deleted. The models were between 200 and 3000 triangles, which was more than acceptable based on the results of the main question above. While testing and experimenting, an effective approach to reducing texture file sizes was discovered. This was done by changing the export settings of 'dilation' inside of Substance Painters custom glTF exporter, which is described in more detail below.

Figure 13: Without Padding, with padding and the new export settings.

The new export settings decreased the base colour value file size by 62%. Overall, this decreased the loading times of the project since the more memory (MB), the longer it would take to download and start-up on the user's device. The appearance also remained the same, thus maintaining the quality while reducing the load on the server. The export settings are listed in the appendix at **Annex 5**. UV set structuring inside Blender was also crucial for joining models together. If this was not done, then UVs would be deleted and lost when combining models. This was a problem that took a long time to fix, because there was no obvious solution at first. However, changing all the 'UVset' names into one general name and then joining them fixed this issue.

An issue that transpired while modelling smooth-shaded surfaces was that the models, once imported into Unity, looked visually unattractive. This is a bug in Blender when it comes to exporting normals in models that use smooth-shading. A normal, in basic terms, is the direction that a polygon face points in on a model. It was relatively simple to fix this bug. When selecting the model inside of Unity, the '*Normals'* setting was changed from '*Import'* to '*Calculate'*. A few other assets, such as the playground slide encountered this same issue. When exporting from Unity to the three.js server, the normals stayed consistent with the Unity settings and thus the problem was solved.

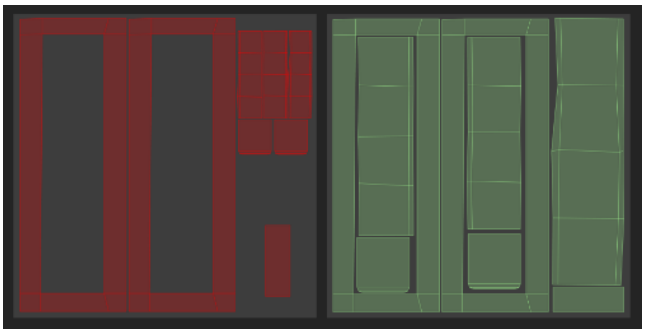


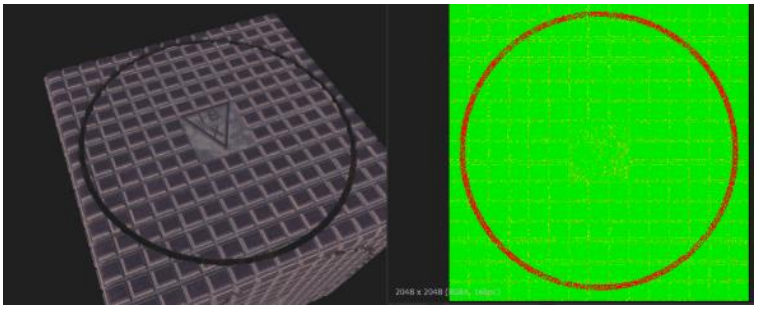
Figure 15: Un-optimized UVs on the left and optimized UVs on the right.

For some models such as the hedges, their UV space was not calculated efficiently by Blenders smart UV tool, therefore it was manually arranged. As per the figure above, the space is completely filled in on the optimized side. Using as much UV space as possible also enhances the performance as explained in the theoretical framework. Combining many materials into a single one reduced the load on the memory used for the application. This is also useful for future artists who will continue to work on the assets, because they are neatly organized in a structured library.

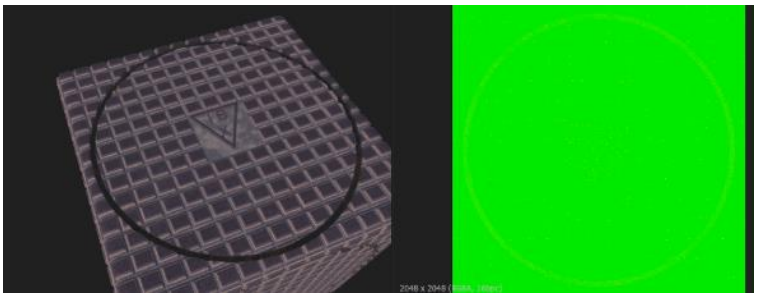
Figure 16: Original bicycle (left) 12000 triangles, optimized bicycle (right) 6620 triangles.

The original bicycle model was bought online but contained too many triangles and had a large texture file size. Therefore, the mesh, texture resolution and file type were optimized. Blender's decimate tool was found to be extremely useful in lowering the number of triangles. It works by averaging the distances between vertices and removing those which it considered to be unnecessary, while maintaining the shape of the original mesh. The result was that the triangle count went from 12 000 triangles to around 6000 triangles, decreasing the amount by roughly 50%. The texture was originally a 62 MB TGA file format, this was compressed to a JPG file format and the resolution was downscaled from 2048 to 512. This decreased the total texture size to 513 KB. As seen in the image above, the appearance remained the same. In the end, the student created three variations, as shown in the Graduation results chapter. As stated earlier on in this paper, all textures were PBR validated in order to justify them for the environment.

Below you can see a PBR invalid material due to the red colours:



After optimizing the texture by using a levels node inside Substance Painter, the final result was validated as successful. Another example can be seen in **Annex 3**. If compared, the result is almost unnoticeable, but the light will interact with the asset realistically in the fixed version:



This texture was created inside of Substance Painter using a 2048x2048 resolution (for the highest detail) and inside Unity it was downscaled to 256, thus improving the size from 22 MB to 384 KB. These improvements to textures decreased the overall loading times of the project significantly. In terms of model density optimisation, the main achievement was downscaling the Volvo car model from 26000 triangles, to around 9000 triangles. This was done by deleting all of the unnecessary geometry manually inside of Blender.

Throughout the duration of the graduation assignment, the student received constant advice and support from an employee nearby to export the projects to the three.js server. This process was extremely complicated and many issues arose due to faulty tools, scripts and outdated programs being used. Since the company was in the process of upgrading their work pipeline to a new system, the issues where left alone and the student had to keep working with what was provided.

d) What are the performance results of the final Innobrix version on each device platform?

The student completed benchmarks on a few different devices in order to gather the performances on each platform. This was done as a final test to evaluate the performances of the new environment. The main limiting factor consisted of the detailed volume houses provided by the company, because the draw calls of each detailed volume averaged around 200. This had a major impact on the performance during the entire project. Since there were around 24 detailed volumes in the final version, the performance was beyond fixable. However, the 24 detailed volumes were considered a requirement by the company and the impact on performance was therefore disregarded.

Detailed volumes were disabled during the tests. This was done because they would be optimized by the company at a later date.

Optimisations were applied to the assets created, as mentioned in the question above. The results would determine whether or not the methods chosen were effective. The devices chosen are based on what the company determined to be the standards in the industry. Of course, the more devices, the more testing could be done, but these where predetermined variables. Adding additional devices would slow down the performance testing schedule, which is why using the most popular devices was chosen as the approach of testing (Oxborrow, 2018).

What the company could offer and what the student could manage to get hold of were the determining factors. If more devices were needed, then they would be added to the recommendations section of this report. The scene tested can be found here: https://configurator-test.innobrix.nl/bruce/

*Note: Each device specification is attached to **Annex 10**.*

SMARTPHONE DEVICE PERFORMANCE COMPARISONS

	Mobile 1: **Samsung S7 Edge**	Mobile 2: **Apple iPhone XS**	Mobile 3: **Huawei P30 Pro**	Mobile 4: **Razer phone**
FPS (Frames-per-second)	6 FPS	60 FPS	35 – 60 FPS	5 – 8 FPS
Memory (RAM %)	10 MB	*Hidden*	10 MB	10 MB
Milliseconds MS	83 – 113 MS	13 MS	30 MS	55 – 120 MS

When Innobrix crashed on the Apple device, the cache had to be reset in order to load the scene again. They also seemed to lock the FPS to 40 when the device warmed up as a safety precaution to save battery power. A discussion with the team came to the conclusion that Apple performs better due to them having custom GPU and CPU drivers made to make rendering graphics a priority.

With Huawei, the FPS recording was impressive and in the same league as Apple. The result is due to it having a newer GPU and more CPU power than most devices. An interesting discovery was that using the 'performance ultra-mode' setting disabled the FPS from dropping below 60 FPS. Memory could not be extracted from mobile devices as they displayed incorrect results or were hidden. This was most likely done due to security and/or privacy reasons.

MID-RANGE DEVICE PERFORMANCE COMPARISONS

	Tablet 1: **Samsung Galaxy Tab S2**	Tablet 2: **Apple iPad Pro 2017**	Laptop 1: **Asus ROG GL552VW**	Laptop 2: **Asus UX305C**
FPS (Frames-per-second)	2 FPS	31 FPS	18 – 24 FPS	7- 11 FPS
Memory (RAM %)	10 MB	*Hidden*	47 MB	36 MB
Milliseconds MS	90 – 155 MS	30 MS	38 – 42 MS	96 - 122 MS

The mid-range devices turned out to be mediocre in performance. This is most likely due to older hardware being used that has not been adjusted to run three.js well. The iPad Pro 2017 uses a *PowerVR* GPU that was specifically designed for 2D/3D rendering, DirectX, OpenVG, OpenGL ES, and OpenCL acceleration. Apple is much further ahead in terms of providing better performance to intensive applications, such as Innobrix. Samsung performed poorly all-round, due to their low CPU processing speeds and average GPU cards. Once again, the memory results were incorrect and should be disregarded.

HIGH END DEVICE PERFORMANCE COMPARISONS

	Computer 1: **Low-End PC**	Computer 2: **Developer PC**	Computer 3: **Gaming PC**	Computer 4: **High-End PC**
FPS (Frames-per-second)	55 FPS	56 – 60 FPS	80 – 83 FPS	60 FPS
Memory (RAM %)	41 MB	38 – 56 MB	52 MB	40 MB
Milliseconds MS	11 MS	14 - 16 MS	11 MS	10 MS

The results for high end devices (desktop computers), produced logical results with comparison to the hardware and performance outcomes. Having a computer with a strong GPU and CPU proved to have better results. This can be seen in **Annex 10** when comparing the different hardware set-ups of each device. An interesting note is that the gaming PC used a 144 Hz monitor, as most computers use a 60 Hz monitor. This meant that the FPS of the scene was not capped to 60 FPS and therefore, it went to much higher values providing for a smoother viewing experience.

9. Conclusion

Photogrammetry could be a viable option for irregular-large models, while using better camera equipment might also improve the final outcome. Since it is still a relatively new technology and the time that it takes in order to get a decent result, make it likely that the outcome will not be acceptable. Shadows and reflections also caused issues in terms of removing the realistic aspect. Therefore, using traditional methods to create 3D assets was the preferred choice. The PBR workflow created visually appealing assets since it used more texture maps.

This was a problem for the graduation assignment, because the company only used albedo maps at the time of writing this report. However, these maps are available for every asset created in case they want to implement PBR maps in the future. As mentioned in the project setup overview chapter, the glTF PBR Metal Roughness export configuration inside Substance Painter turned out to be extremely useful in exporting the necessary file types quickly and efficiently. In terms of device performance of the students Innobrix demo, Apple proved to have the best low and mid-range device results. Huawei also had great results for their mobile device. In regard to these results, Samsung devices should be used as a benchmark version for the lowest possible graphics in order to run. The results are logical; the better the CPU and GPU, the better the performance. Apple has custom GPU drivers that are dedicated to graphic intensive applications, hence why the results are impressive.

The first Innobrix demo containing the *kavelkaart*, configurable houses and water (excluding all the work the student did), already performed poorly on most mobile and mid-range devices. This meant that no matter how much optimisation was done to the assets, the Condition of Satisfaction of having Innobrix run on every device would never be achievable. The only way to fix this would be for the development team to optimize the engine as well as the detailed volume houses.

Draw calls of the detailed volumes were determined to be the main performance cost. Therefore, combining as many models, sub-meshes and materials as possible into one would improve the overall performance and loading times substantially. This would however mean the functionality would be lacking, since having one combined model has 0 options, meanwhile a detailed volume with 20 options would provide more customization for the client. A decision will need to be made at the company on what they should focus on: scene performance or house customizability. Another possible solution is for the company to research the effects of using a LOD (level of detail) for each detailed volume model.

10. Recommendations

The scene that was provided for the graduation project was extremely unorganized and overloaded with unnecessary files that were used in previous discontinued projects. Using the correct folder structuring was not easy to grasp and took a considerable amount of time, since the code was hard-coded in a lot of cases for individual projects. Due to the Innobrix engine being moved to an online version it would have been unnecessary to reorganize everything in the Unity version at this stage. This issue was mentioned because it slowed down the process of development significantly by causing lots of script issues. It is recommended that future projects follow a strict folder structuring guideline. On mobile devices, the FPS dropped at various random points, but mainly when showing

many objects on the screen at the same time. This is because the device's graphical specifications are not able to handle the sheer amount of draw calls. It is recommended to lower the asset count as much as possible and also to limit the user to not be able to go up close to assets, while on mobile devices. It is also recommended to keep the framerate rendering speed below 30MS in order to avoid crashes.

The student used a 144-hertz monitor at home and noticed the web version ran a lot smoother due to the higher refresh rate. This can be seen in the results with the FPS performance recordings of the gaming computer. It is recommended that the company invests in a high quality 144-hertz monitor to give new clients the smoothest and best possible experience of Innobrix.

The company should force the Innobrix web version to use a built-in auto-detect function and have a few variations that can be used per device type. A mobile device with high-end hardware (such as Apple) that can handle a lot will be given a higher quality 3D version, while a lower-quality phone (such as Samsung) will be able to access a low-quality version of Innobrix. It is suggested for the company to also install a better stats viewer for Innobrix. The '*threex.renderstats*' (https://github.com/jeromeetienne/threex.rendererstats) is a great tool that would benefit every employee, as they would be able to view performance costs such as draw calls and vertices in real-time. Due to the finding of detailed volumes costing a huge amount of performance, some options were given to the company in order to try and mitigate the issue.

Option 1: have one detailed volume house per area. The other houses are empty, non-clickable, low polygon models (fake houses), as displayed in Figure 15 below:

Figure 14: Blue circle = detailed volume house, Red circle = fake house.

This would mean that the final count of detailed volume houses would be around five in total.
Option 2: remove detailed volumes and just have one house model mesh, so that it can't be customized once out of the configurable mode.

Option 3: optimize the three.js engine to handle detailed volume houses so that they do not affect performance as much. This can be done by using a LOD (level of detail) system for displaying models. The company should have no more than 10 to 14 detailed volumes in any project. Three.js was not made to handle thousands of objects in a scene instantaneously.

11. Graduation products

The most successful products created are listed below. The final product can be viewed on the production server here: https://configurator.innobrix.nl/hifi

Reference photo on the left, final 3D Hedge model textured on the right.

Reference photos above and 3D models textured below.

Garden assets, all 3D models and textures.

Bus stop 3D model with texture.

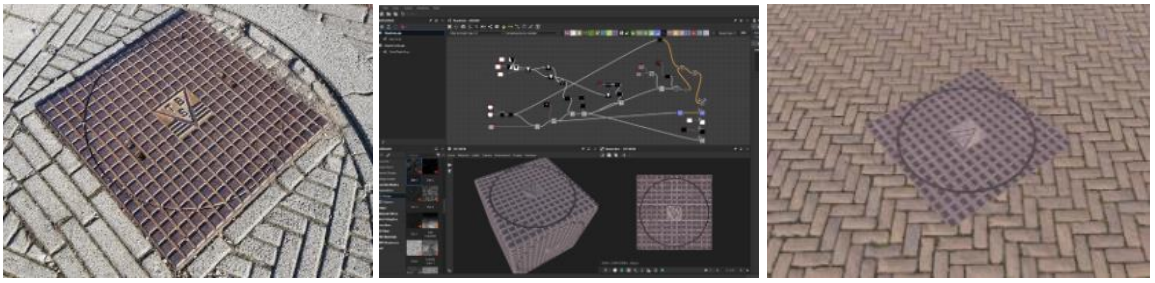
Recycling bin variations, 3D models and textures.

Post-box 3D model and texture.

3D Models (left side), Reference Photos (right side).

Creating the Fence: Reference photo, Substance Designer building of the fence texture, applying the fence texture to a cube model inside Blender, the final result.

The three bikes textured.



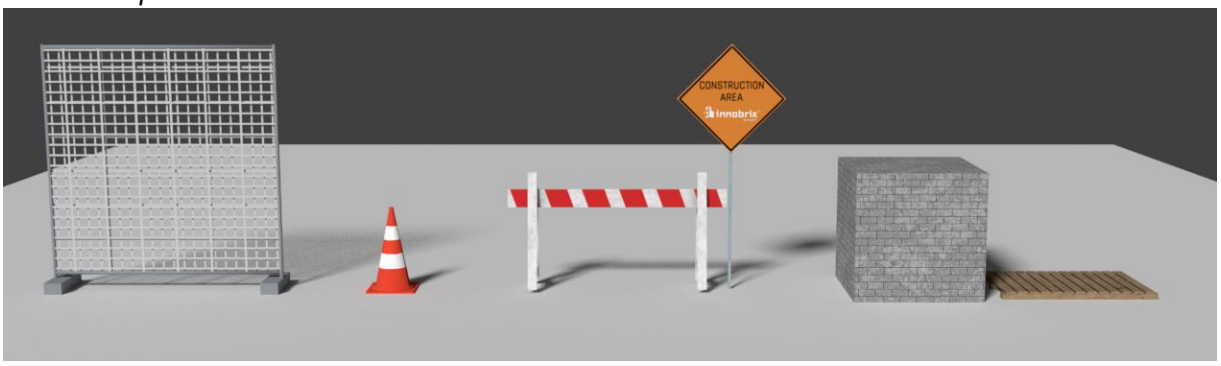
Manhole asset: Reference photo, Substance Designer work and final Innobrix implementation

Left = Old Innobrix version, Right = New Innobrix version that the student created.

Left = Old Innobrix version, Right = New Innobrix version that the student created.

Screenshot from the new version.

Screenshot from the new version.



Construction assets created, for the company to use when there is a new building being placed:

References

Bech-Yagher, C. (2018, October 23). *UV mapping for beginners*. Retrieved from CB Creative Bloq: https://www.creativebloq.com/features/uv-mapping-for-beginners

Bosset, P. (2017, March 1). *HORIZON ZERO DAWN: IT'S GORGEOUS, IT HAS SUBSTANCE!* Retrieved from Substance3D: https://www.substance3d.com/blog/horizon-zero-dawn-it-s-gorgeous-it-has-substance

Cozzi, P., & Riccio, C. (2012). *OpenGL Insights.* CRC Press. Retrieved from https://books.google.nl/books?id=CCVenzOGjpcC&dq=texture+atlas+increase+performance+game+&lr=&source=gbs_navlinks_s

Deliot, T., & Heitz, E. (2019, February 14). *Procedural Stochastic Texturing in Unity*. Retrieved from Unity 3D Blogs: https://blogs.unity3d.com/2019/02/14/procedural-stochastic-texturing-in-unity/

Kolenberg, A. (2014, September 8). *De Sims 4 Review*. Retrieved from XGN: https://www.xgn.nl/review/58935/de-sims-4-review

Lievendag, N. (2016). *How 3D Scanning Was Used To Create The Worlds Of Star Wars Battlefront*. Retrieved from 3D Scan Expert: https://3dscanexpert.com/3d-scanning-star-wars-battlefront/

Lindquist, B. (2015). *Game Environment Texturing.* Finland: Metropolia Ammattikorkeakoulu. Retrieved from https://www.theseus.fi/bitstream/handle/10024/93856/Lindquist_Benjamin.pdf?sequence=1&isAllowed=y

Mader, P. (2005, December 2). *Creating Modular Game Art For Fast Level Design*. Retrieved from Gamasutra: https://www.gamasutra.com/view/feature/130885/creating_modular_game_art_for_fast_.php

McDermott, W. (2018). *THE PBR GUIDE BY ALLEGORITHMIC - PART 2*. Retrieved from Substance Academy: https://academy.substance3d.com/courses/the-pbr-guide-part-2

Nazarov, R., & Galletly, J. (2013). *Native browser support for 3D rendering and physics using WebGL, HTML5 and Javascript.* Retrieved from https://pdfs.semanticscholar.org/1f8e/52394a4d4d9de9da110c036ccace5bb97221.pdf

Oxborrow, I. (2018, December 2018). *World's most popular smartphone brands, Q3 2018*. Retrieved from The National: Business: https://www.thenational.ae/business/technology/world-s-most-popular-smartphone-brands-q3-2018-1.799408

Powell, R. (2014). *Optimized Graphics for Handheld Real-time CG Applications.* Uppsala University, Department fo Game Development. Uppsala University. Retrieved from https://www.diva-portal.org/smash/get/diva2:722454/FULLTEXT01.pdf

Rooney, P. (2012). *A Theoretical Framework for Serious Game Design: Exploring Pedagogy, Play and Fidelity and their Implications for the Design Process.* International Journal of Game-based Learning. doi:10.4018/ijgbl.2012100103

Russell, J. (2015, November 1). *BASIC THEORY OF PHYSICALLY-BASED RENDERING*. Retrieved from Marmoset: https://marmoset.co/posts/basic-theory-of-physically-based-rendering/

Ryan, P. (2018, January 8). *Art Pipeline for glTF*. Retrieved from Khronos Group: https://www.khronos.org/blog/art-pipeline-for-gltf

Sandrik, D. (2018, January 16). DAVID SANDRIK'S INVENTED HOMES: BALANCING ART AND TECHNIQUE. (V. Gault, Interviewer) Retrieved from https://www.substance3d.com/blog/david-sandriks-invented-homes-balancing-art-and-technique

Tavares, G. (2012). *WebGL fundamentals*. Retrieved from html5rocks: https://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals

Wobma, E., & Bruggink, J.-W. (2012, December 4). *Dutch population taller and heavier*. Retrieved from Centraal Bureau voor de Statistiek: https://www.cbs.nl/en-gb/news/2012/49/dutch-population-taller-and-heavier

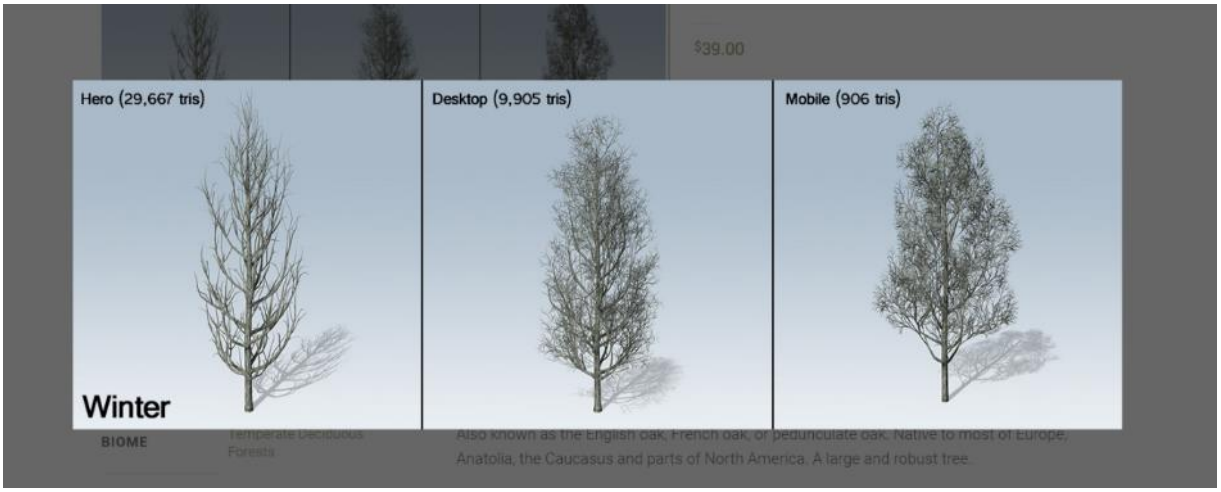
Wu, Q. Y. (2016, June 23). *Three.js pros and cos*. Retrieved from Chewy.ninja: http://chewy.ninja/blog/2016/6/19threejs-pros-and-cos/

Zakrisson, J. (2019, February 18). DICE: Behind the Art of Battlefield and Battlefront. (K. Tokarev, Interviewer) Retrieved from 80 Level: https://80.lv/articles/001agt-dice-behind-the-art-of-battlefield-and-battlefront/

Annexes

Annex 1. Polycount amounts per device.

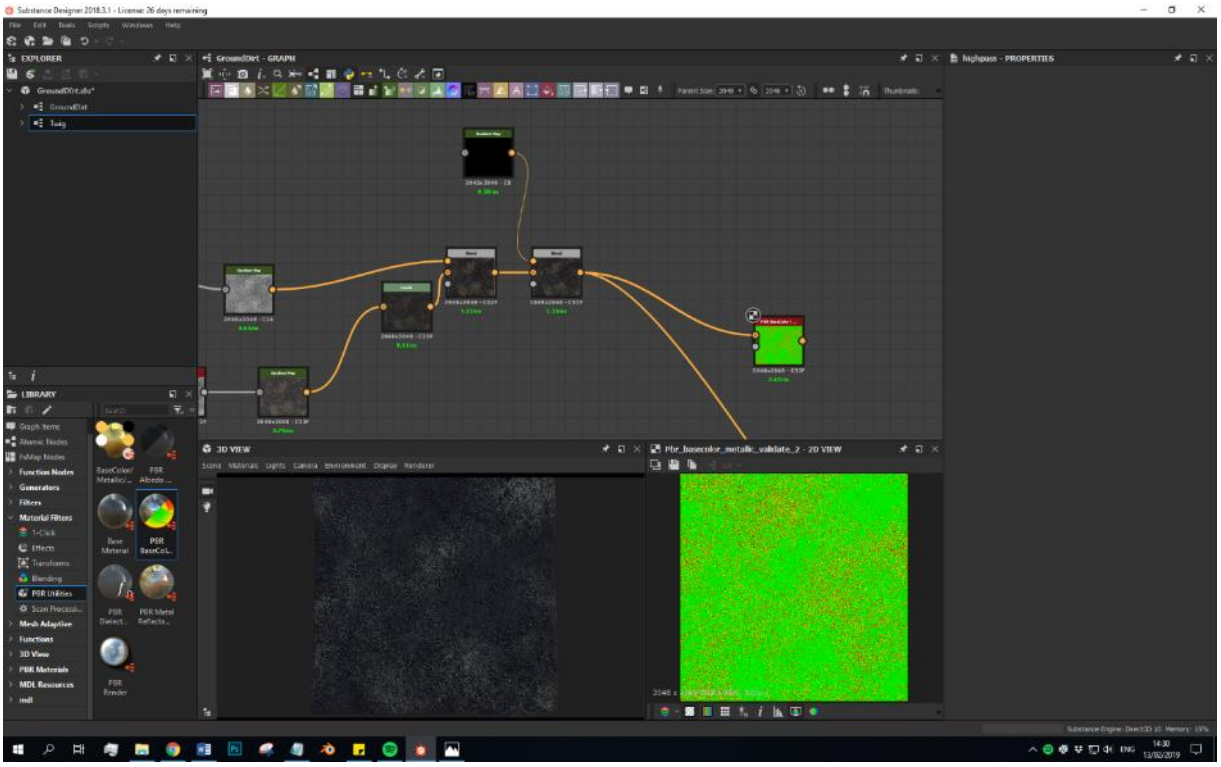
Add reference photos in small screenshot form, show overview, highlight best



Annex 2. Optimizing the export script.

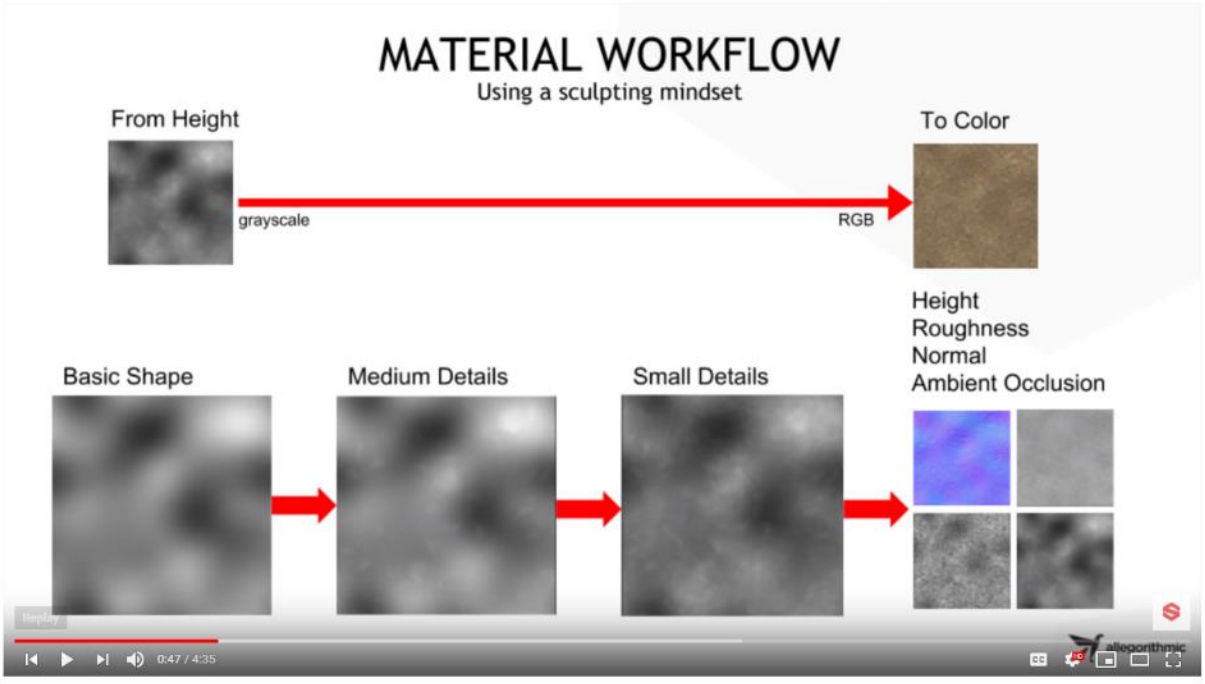
```
        startInfo.Arguments = "/C gltf-pipeline -i \"" + Directory.GetFiles(_exportRootPath, "*.gltf")[0] + "\" -b -d --secure";
        startInfo.RedirectStandardOutput = true;
        startInfo.RedirectStandardError = true;
        startInfo.UseShellExecute = false;
        startInfo.CreateNoWindow = true;
        Process processTemp = new Process();
        processTemp.StartInfo = startInfo;
        processTemp.EnableRaisingEvents = true;
        processTemp.Start();
        processTemp.WaitForExit();

        System.IO.File.Move(_exportRootPath + "/parcelMap-processed.glb", _exportRootPath + "/parcelMap.glb");

        Debug.Log("Done");
    }

    private List<Transform> GetTransformsRecursive(Transform parent)
    {
        List<Transform> transforms = new List<Transform>();

        var children = parent.GetComponentsInChildren<Transform>().ToList();
        children.Remove(parent);

        transforms.AddRange(children);
```

Annex 3. PBR Validation using Substance.



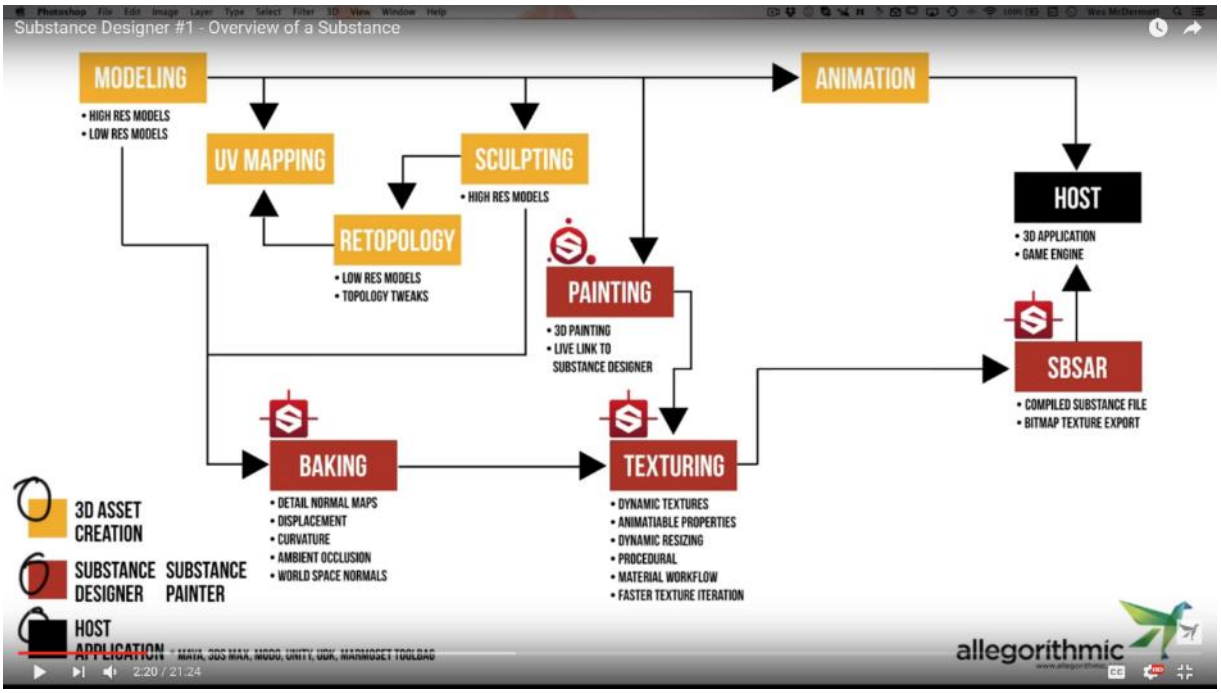
Substance Designer PBR Validation node, red dots mean it is not optimal, a full green node result would be acceptable.

Annex 3A. Workflows.



01-03: Understanding the material workflow

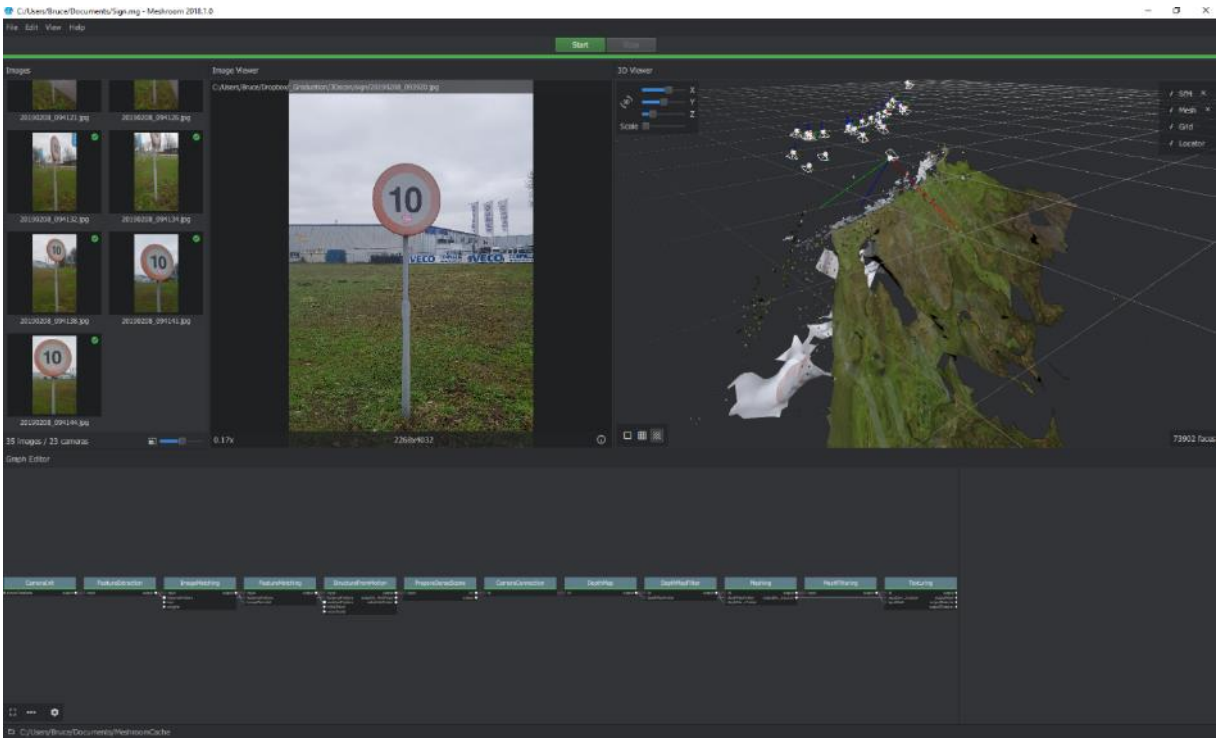
Substance Designer workflow breakdown.



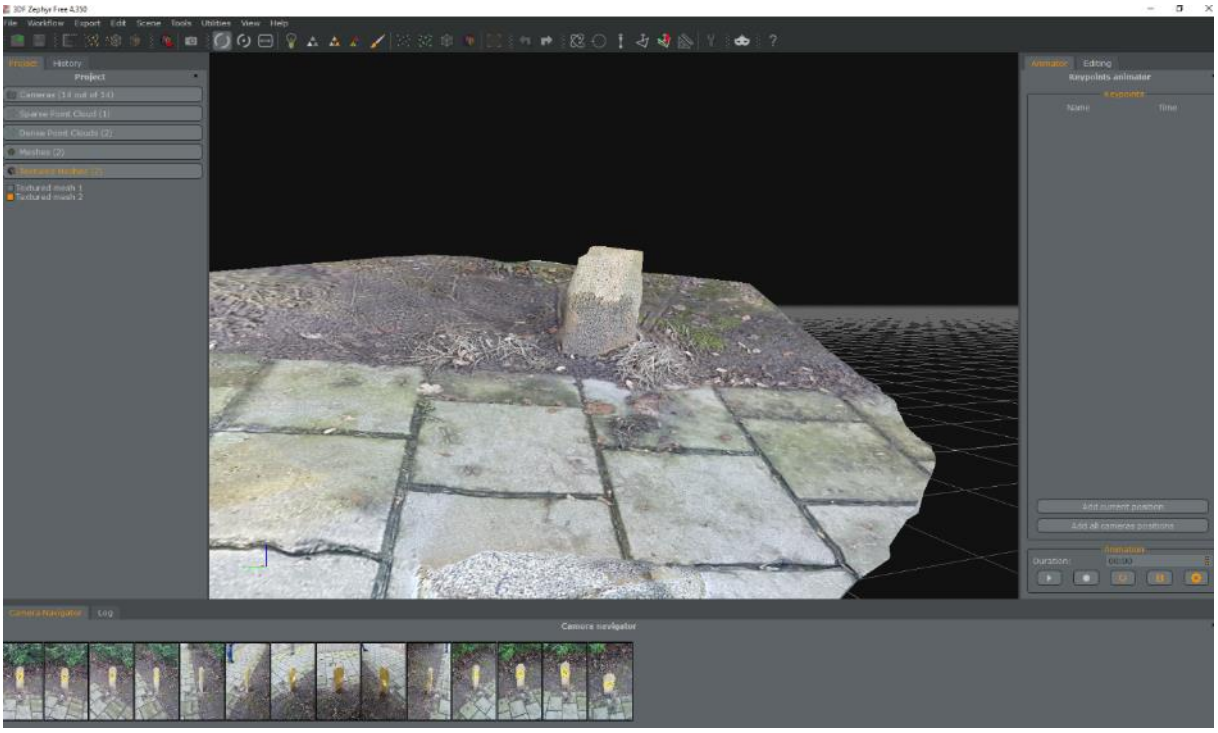
The overall workflow pipeline used as an industry standard for creating games.

Annex 4. 3D Photogrammetry results.

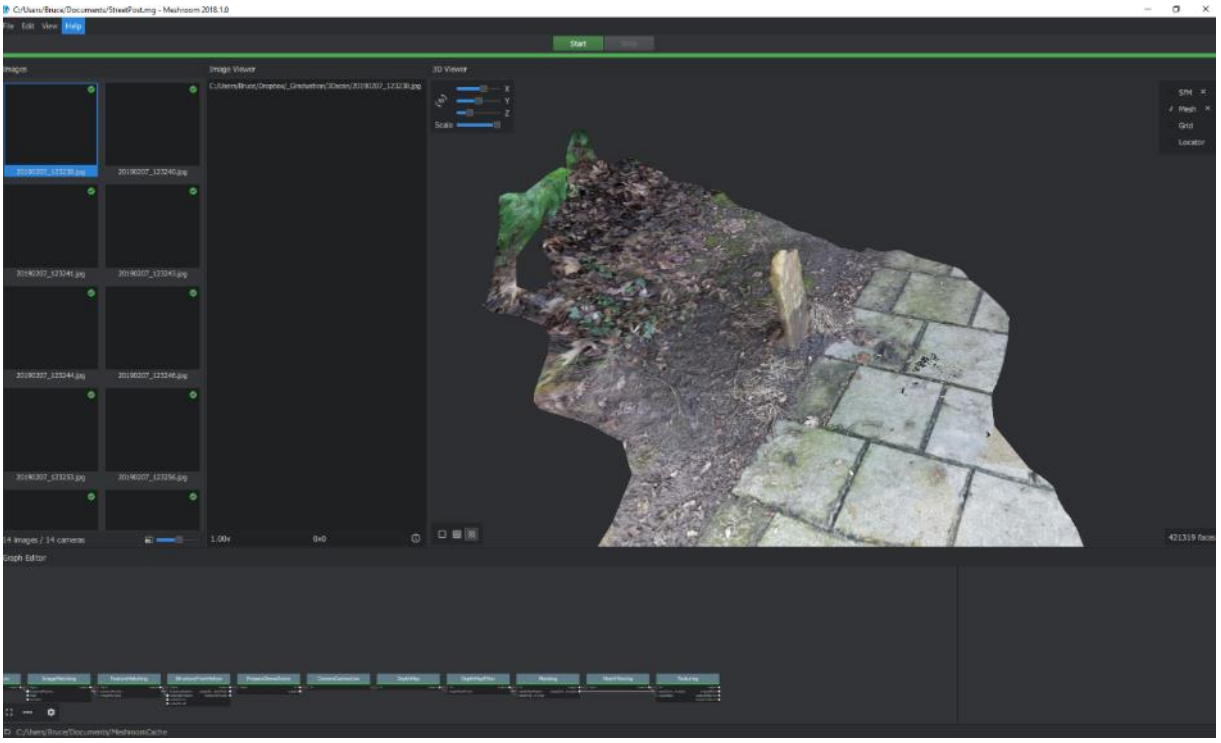
- Sign Post result; using 3DF Zephyr.



- Sign Post result; using Meshroom.

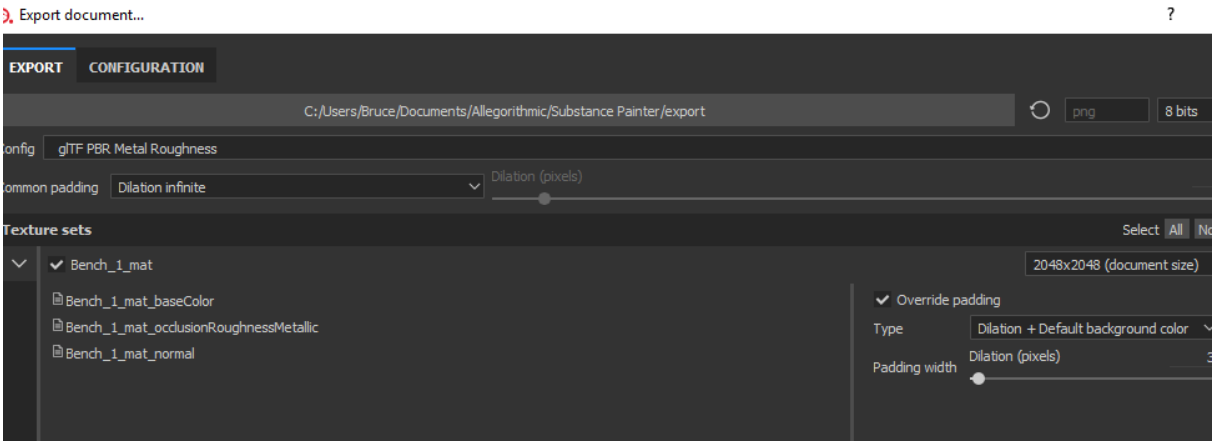


- Pole result, using 3DF Zephyr.



- Pole result, using Meshroom.

Annex 5. Substance Painter export glTF settings.



Annex 6. List of deliverables

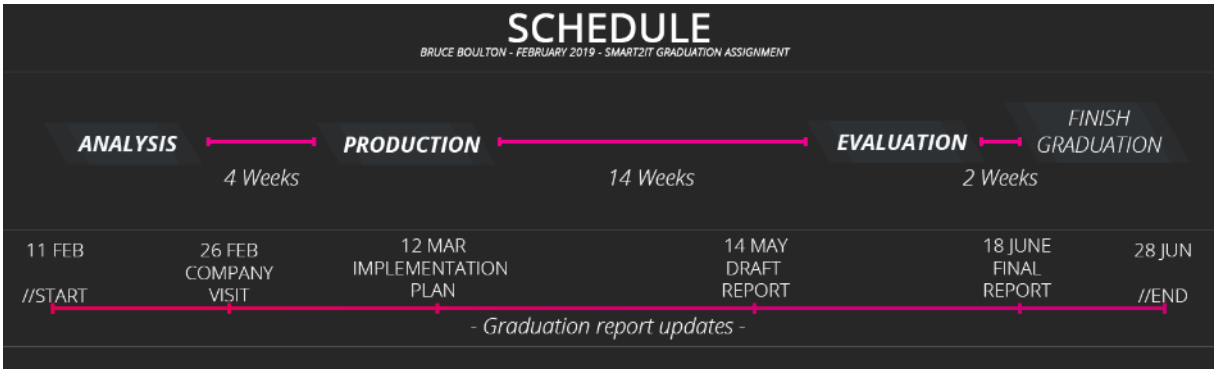
This list is derived from the ‘asset list’ document. It contains assets the student has been approved to make. It falls into the schedule of 4-week iteration cycles of the student’s ‘asset creation workflow’ diagram inside of the schedule illustration (under ‘Production’). Asset list [42 total models/textures]. This number varies between documents as it was constantly under iteration. This document should be seen as a rough indication to what the student will produce. The number in brackets indicates the level of importance, with 10 being the highest.

- Houses (10)
- Add 3 new variations
- Create the detailed volumes from scratch.

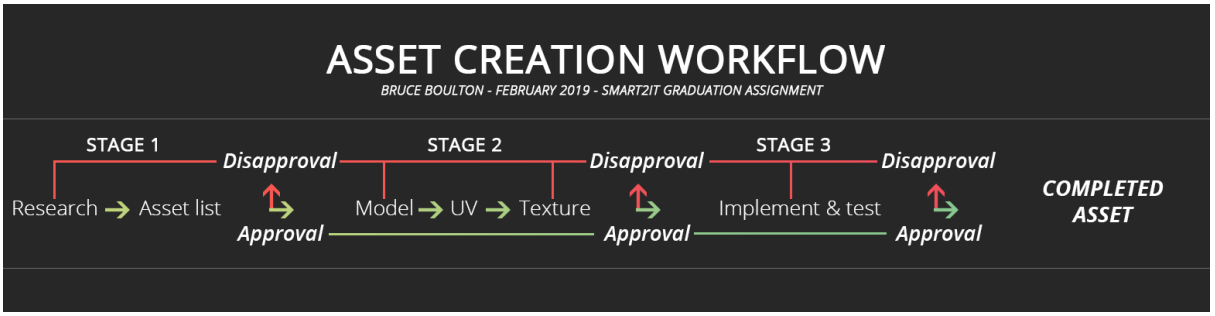
- Place them in the scene
- Create configurable houses
- Build and test the project to see if it works.
- Roads (4)
- Textures will be created.
- Black tar
- Cobblestone
- Red brick
- Pavement (4)
- Textures will be created.
- Standard grey bricked pavement
- Grass variances (3)
- Purchase from online and then make final adjustments.
- Dirty grass
- Muddy grass
- Normal grass
- Weeds
- Wild grass
- Fences (5)
- Textures and Models will be created.
- House fence.
- Black fence
- Playground fence
- Street lights (6)
- Textures and Models will be created.
- Old streetlight
- Modern streetlight
- Traffic lights (6)
- Textures and Models will be created.
- Modern traffic light
- Street signs (4)
- Textures and Models will be created.
- Stop sign
- Bicycle sign
- Parking sign
- Pedestrian sign
- Recycling bins (8)
- Textures and Models will be created.
- Plastic, Glass, Paper, Waste
- Standard blue bin
- Dustbin
- Big recycling bins blue and orange
- Bus stop (5)
- Textures and Models will be created.
- Standard bus stop

- PostNL mail box (1)
- Textures and Models will be created.
- Standard PostNL mail box
- Electricity box (1)
- Textures and Models will be created.
- White electricity box
- Green electricity box
- Picnic table (4)
- Textures and Models will be created.
- Standard wooden picnic table
- Bicycle stand (4)
- Textures and Models will be created.
- Standard Dutch metal bike stands
- Car stop poles (2)
- Textures and Models will be created.
- Standard car-stop pole variant 1 (red and white)
- Standard car-stop pole variant 2 (black with red stripe)
- Trees (10)
- Textures and Models will be bought or re-used from previous projects.
- Birch Tree
- Pine Tree
- Willow tree
- Oak Tree
- Cedar Tree
- Cypress Tree
- Flowers (9)
- Textures and Models will be created.
- Yellow flowers
- Rocks (10)
- Textures and Models will be created.
- Rock Big
- Rock Medium
- Rock Small
- Ducks (1)
- Textures and Models will be created
- Dutch white call ducks
- Reeds (3)
- Textures and Models will be created
- Boat (5)
- Textures and Models are already created
- Speedboat model
- Cars (3)
- Textures and Models will be bought from online
- 3 different variations (Small, medium, big)

Annex 7. Schedule and asset creation workflow



Schedule diagram illustrating the student's plan during the graduation assignment.



Asset creation workflow diagram illustrating work structure.

Annex 8. Asset list

https://docs.google.com/spreadsheets/d/1MlWM7bQQ9Jz3iqz6mjOTIPP98pZz5Aba48L_vntVWaE/edit?usp=sharing

Annex 9. May-June plan

Overview

With the final two months of my graduation coming up, I created a plan to adjust to the time left with the most important aspects.

Goals

- Release the library of assets I created.
- Make a document showcasing the assets and triangle counts, with notes.
- Finish hifi demo and release the final version to the production server.
- Make new 2D parcel map.
- Do performance tests for my report.

List of deliverables

- Adjust the option list for the Vorm middle villas.
- Variation in ground textures, planes
- Adjust the option list for the standard row of houses in the front.
- Create new 2D parcel map
- Add two new configurable houses to scene
- Create new detailed volume house

- Fix prices of option lists for new houses
- Create the park
- Playground models and textures
- Create gardens per house type (4 variations
- Pathway
- Plants
- Brick wall
- Create bike stands
- Create backyards for some houses
- Table and chairs
- Barbeque
- Outside couches
- Add more to water area
- ~~Ducks~~ (suggested)
- ~~Boats~~ (suggested)
- Reeds
- Add more cars to the scene
- Expensive car
- Standard car

Annex 10. Device specifications

SMARTPHONE SPECIFICATIONS

	CPU	GPU	RAM
Mobile 1: Samsung S7 Edge	Samsung Exynos 8 Octa 8890 - 2.3 GHz	Mali-T880 MP4	4 GB
Mobile 2: Apple iPhone XS	Hexa-core (2x2.5 GHz Vortex + 4x1.6 GHz Tempest)	Apple GPU (4-core graphics)	4 GB
Mobile 3: Huawei P30	Octa-core (2x2.6 GHz Cortex-A76 & 2x1.92 GHz Cortex-A76 & 4x1.8 GHz Cortex-A55)	Mali-G76 MP10	8 GB
Mobile 4: Razer Phone	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)	Adreno 540	8 GB

MID-RANGE DEVICE SPECIFICATIONS

	CPU	GPU	RAM
Tablet 1: Samsung Galaxy Tab S2	Qualcomm Snapdragon 652, 8 core, 1.8 GHz	Mali-T760 MP6	3 GB

Tablet 2: **Apple iPad Pro 2017**	Hexa-core (3× Hurricane 2.34 GHz + 3× Zephyr)	Apple A10X Fusion GPU	4 GB
Laptop 1: **Asus ROG GL552VW**	Intel Core i7-6700HQ 3.5 GHz	GeForce GTX 960M	8 GB
Laptop 2: **Asus UX305C**	Intel Core m3-6Y30 1.51 GHz	Intel HD Graphics 515	4 GB

HIGH-END DEVICE SPECIFICATIONS

	CPU	GPU	RAM
Low-end PC	Intel Core i3-8350K / 4 GHz	Nvidia GeForce GTX 960	8 GB
Developer PC	Intel Core i5-6700K / 3.30 GHz	Nvidia GeForce GTX 970	16 GB
Gaming PC	Intel Core i7-4590 / 4 GHz	Nvidia GeForce GTX 1070	32 GB
High-end PC	Intel Core i7-7700K / 4.20 GHz	Nvidia GeForce GTX 1080 Ti	16 GB

Annex 11. Weekly graduation report updates

https://docs.google.com/document/d/1EiEl5L0s44tg05KTGan3JcRNoX3McHjVhDNmBQ7rA3U/edit#

Every week, a small update was provided to what had been done. This would be useful for all parties to see the overall progression of the graduation assignment.