InkStorm

[Company name] | [Company address]

[Document subtitle]

17BriceJ

[Year]

Contents

[Analysis 2](#_Toc134700162)

[Problem Identification 2](#_Toc134700163)

[Computational Methods 3](#_Toc134700164)

[Stakeholders 4](#_Toc134700165)

[Takeaways from stakeholder interviews 4](#_Toc134700166)

[Research 5](#_Toc134700167)

[Software 1 - Blender 5](#_Toc134700168)

[Website 1 5](#_Toc134700169)

[Software 2 – Awesome Bump 6](#_Toc134700170)

[Software 3 - Imogen 6](#_Toc134700171)

[Essential Features 7](#_Toc134700172)

[Limitations 7](#_Toc134700173)

[Hardware And Software Requirements 7](#_Toc134700174)

[Success Criteria 7](#_Toc134700175)

[Design 7](#_Toc134700176)

[Decomposition 7](#_Toc134700177)

[Explanation of the process 7](#_Toc134700178)

[Solution structure 7](#_Toc134700179)

[Iterative development 7](#_Toc134700180)

[Example 7](#_Toc134700181)

[Post development testing 7](#_Toc134700182)

[Testing 7](#_Toc134700183)

[Usability testing 7](#_Toc134700184)

[Evaluation 7](#_Toc134700185)

[Success criteria 7](#_Toc134700186)

[Unmet criteria 7](#_Toc134700187)

[Usability features 7](#_Toc134700188)

[Maintenance issues 7](#_Toc134700189)

[Limitations 7](#_Toc134700190)

# Analysis

## Problem Identification

In 3D design and modelling, Displacement maps are used to ‘displace’ 3 dimensional meshes.



Figure 1 - Wood displacement map

They add surface detail without requiring more vertices in the mesh. This means 3D models require fewer vertices to get the same detail.

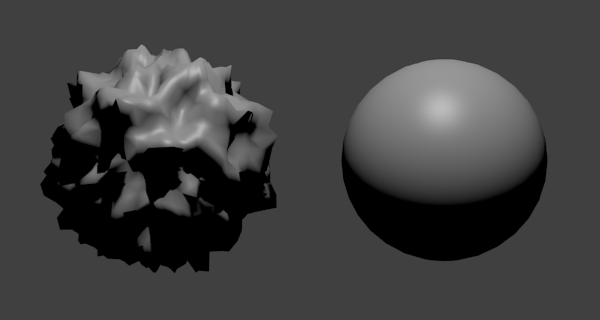


Figure 2 - Sphere before and after displacement

Perlin noise also never repeats, and is deterministic. This means the same seed input will give the same result every time. This means it can be used to add depth to 2d planes in open world games to generate terrain. This is much less labour intensive than manually sculpting terrain, and looks more natural.

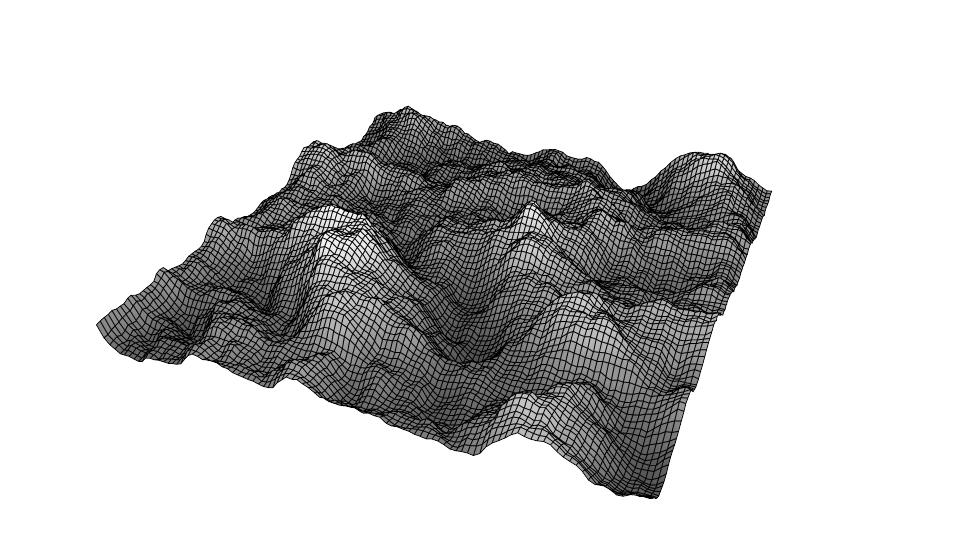


Figure - Terrain deformed by Perlin noise

Examples of games that use Perlin noise for terrain generation:

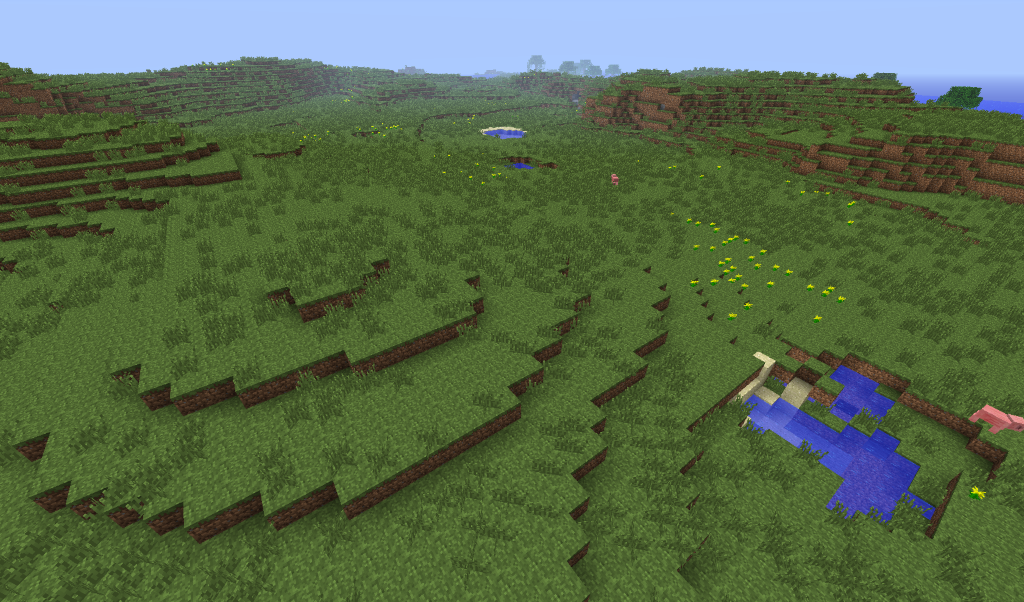


Figure - Minecraft Terrain

Minecraft uses Perlin noise to generate height maps, and uses them to place blocks on the surface. This gives the look of natural terrain.



Figure - No man's sky terrain

Perlin noise was vital in the generation of No Man’s sky, as the developers generated 18 quintillion planets. This would be totally unfeasible to do by hand.

3D models using this technique take less memory and storage space. These maps are often made with types of procedurally generated noise layered on top of each other. These are saved as bitmaps. Most 3D programs support some types of noise, but these mostly run on the CPU and cannot run in real time. Noise generation can be evaluated per pixel, so it is a good candidate for GPU processing, as each pixel can be evaluated on a different core. The most commonly used noise is Perlin noise due to its evaluation speed and its smooth gradients:

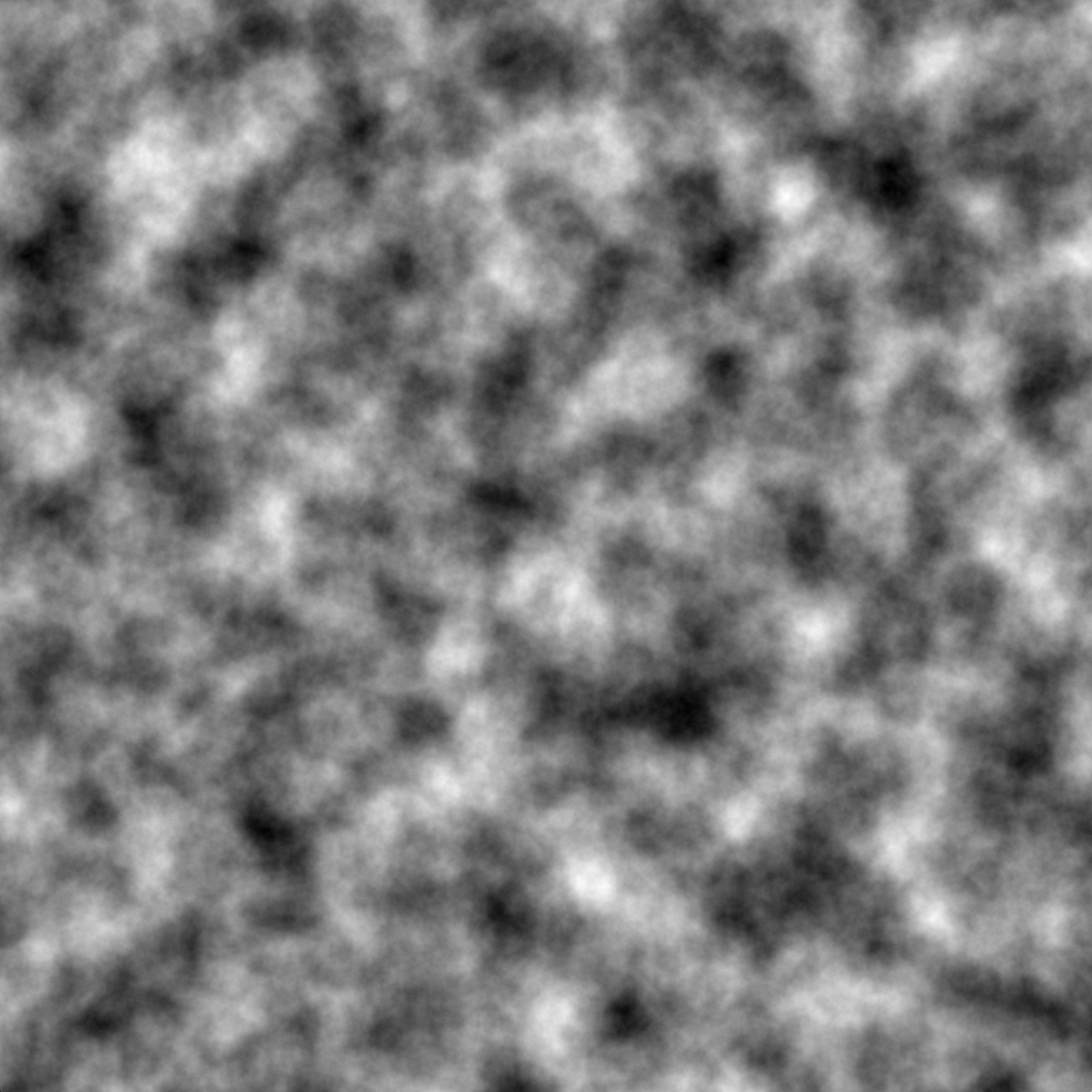


Figure 6 - Example of Perlin noise

## Computational Methods

Perlin noise is evaluated by splitting the viewing area into a grid, and assigning each a random vector of length 1:

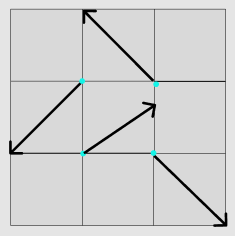


Figure - Example of vector generation

For every pixel, the distance from the surrounding vectors is evaluated:

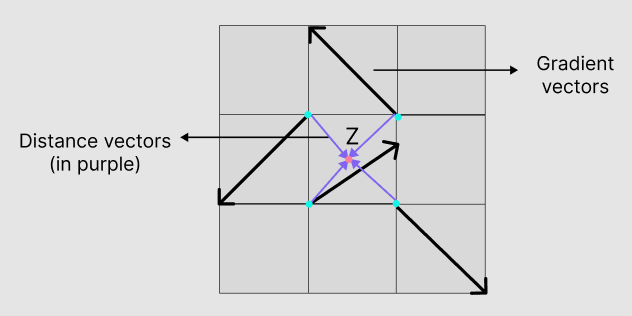


Figure - Distance vector visualisation

The dot product of the distance vectors and gradient vectors is then taken. These are added together to give the brightness of the pixel.

## Stakeholders

Ben Wigley:

Ben uses Blender to do his 3D modelling in his free time. He frequently uses displacement and noise maps while modelling. He is limited by his laptops CPU processing power, and needs the program to run efficiently on his NVIDIA graphics card. He needs a free program with customisable noise size and detail, and needs to be able to save the bitmaps in different resolutions. He also wants comprehensive online documentation in case he forgets how the software works, as he will not be using it every day.

Alfie Bacon:

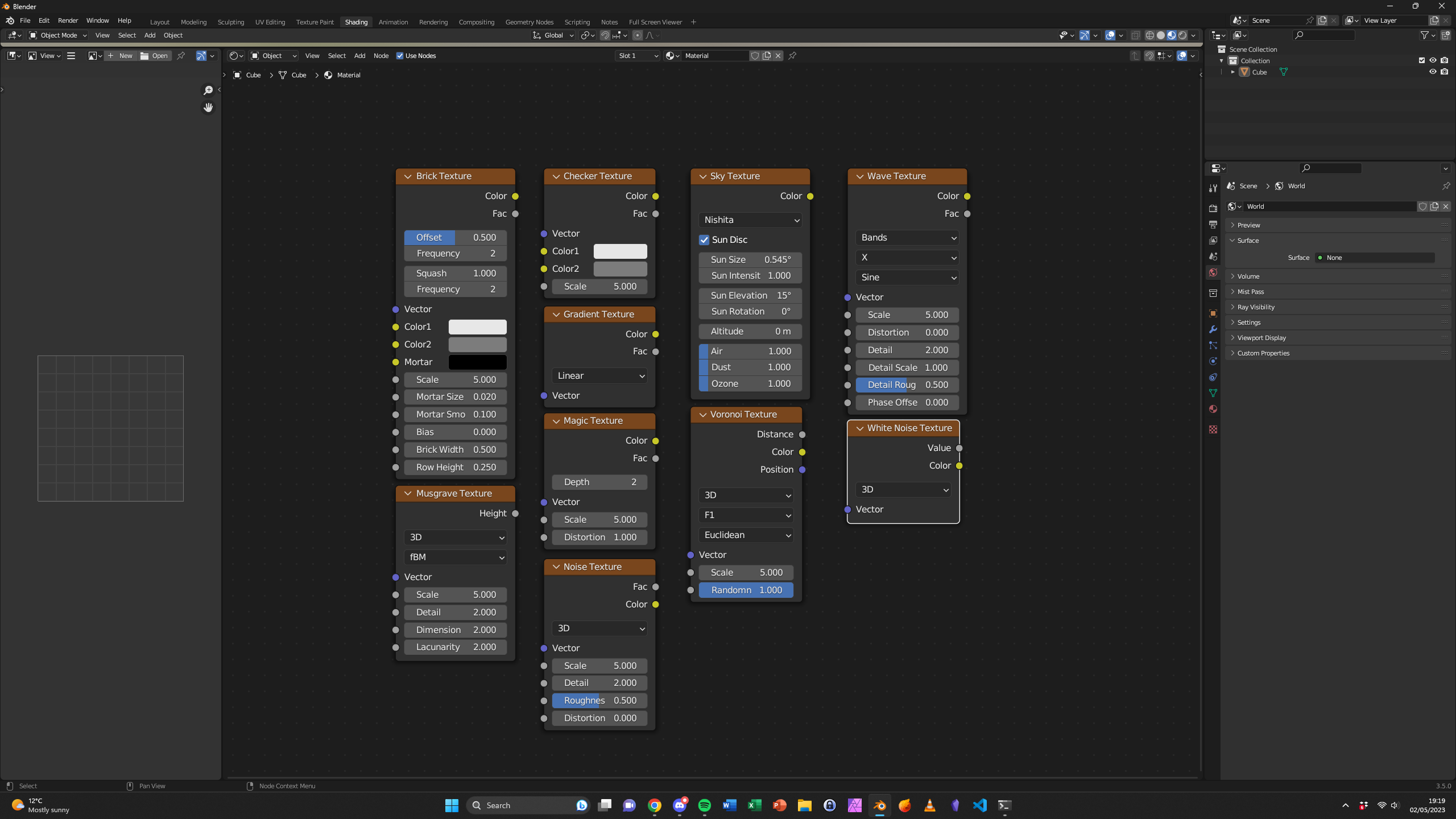
Alfie is a professional 3D animator and filmmaker at a local company. He mainly uses Blender and Unreal engine. Blender has comprehensive options for noise generation, but Unreal Engine lacks variety and customisability in its noise. Due to production workloads and deadlines, he needs to be able to efficiently produce and manipulate 8K textures. The software also needs to be able to utilize his NVIDIA graphics card. Due to the nature of his work, he needs to be able to save and load project files for customisation later. For best detail, he needs to be able to save files as 16-bit greyscale heightmaps. He uses an ultrawide monitor and would like to be able to adjust the user interface to fit.

### Takeaways from stakeholder interviews

* Must run in real time.
* Must have easily configurable noise settings.
* Must be able to save bitmaps in varying sizes.
* Must Utilise NVIDIA acceleration hardware.
* Must have a file format to save projects.
* Must be able to export to 16 bit PNG

## Research

### Software 1 - Blender



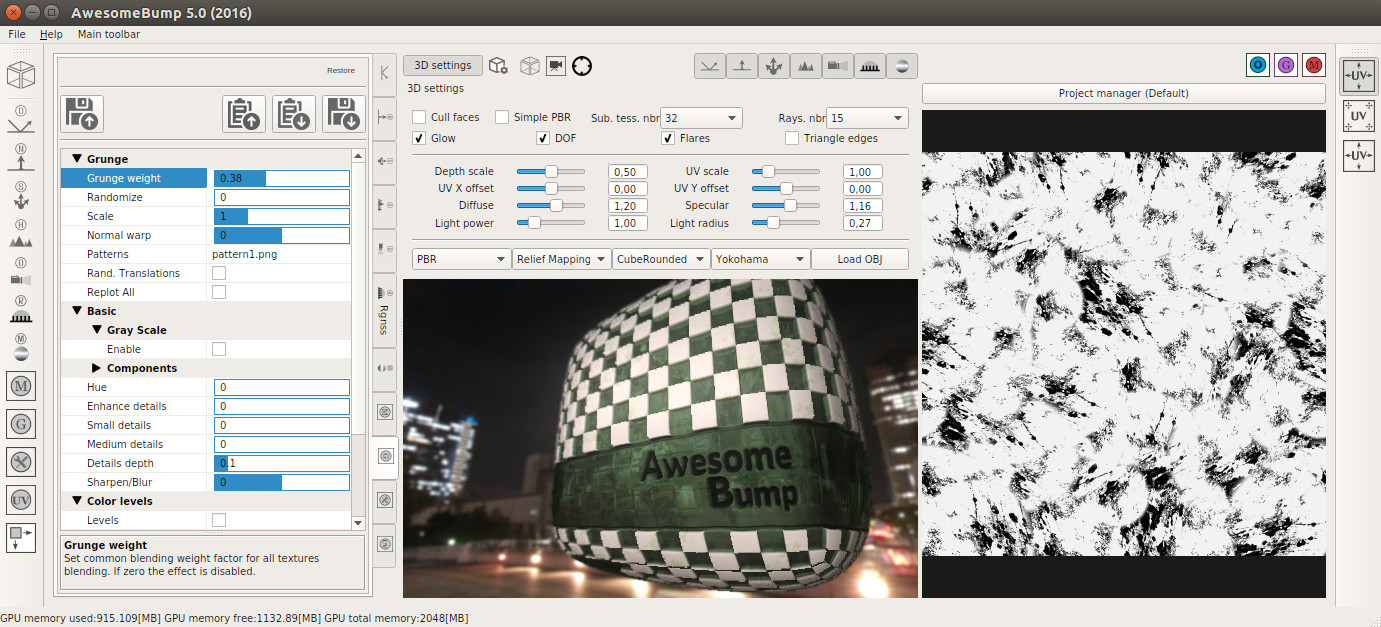
Blender is a free piece of software that has many different options for generating noise. The node-based user interface makes it very powerful but can be hard for beginners to master. Connections can be dragged between nodes, allowing quick development. The colour coded connections prevent any erroneous connections, making it quite user friendly. However, the online documentation is very scarce and non-intuitive. This is something I will strive to address with my project. The slider inputs also allow easy modification of input values. One problem with Blender is that it evaluates the nodes on the CPU, which can lead to slow response times on detailed models. I hope to address this by running my code on compatible NVIDIA graphics cards with the CUDA toolkit.

### Website 1

<https://core.ac.uk/download/pdf/250147208.pdf>

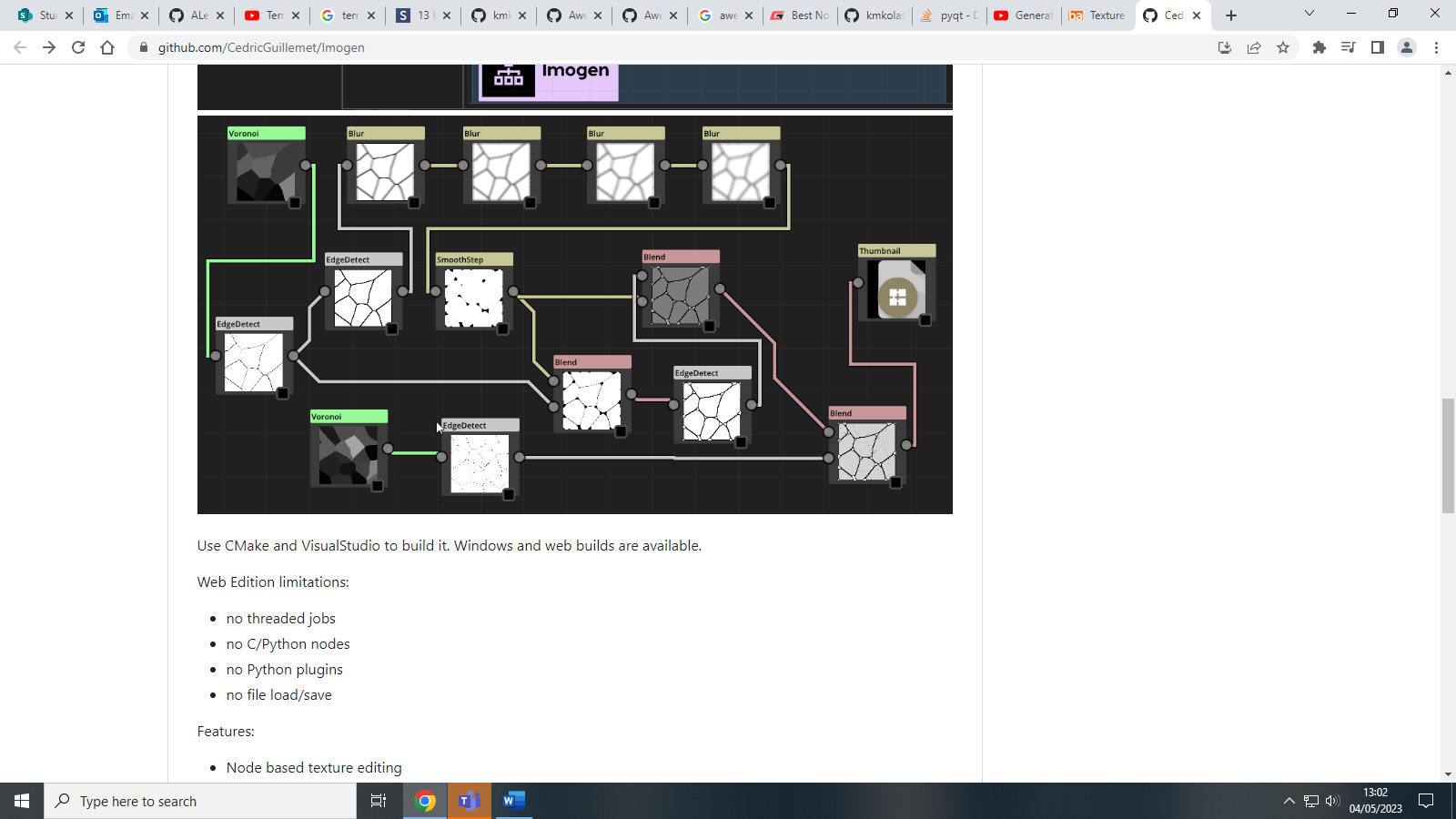
This paper talks about the methods behind generating noise, and how they can be used in video games to generate terrain. The methods explained in the paper will help when designing the back-end functionality of the program.

### Software 2 – Awesome Bump



AwesomeBump is open source and is similar to my goal. The sliders look intuitive to use, so I think I will use a slider-based approach to customisation. I think I will make the texture view bigger so that it is clearer. I don’t think I will be able to achieve the 3D view, so the texture view can take up more space. Unfortunately it can only manipulate existing textures, and does not have the tools to generate its own noise textures.

### Software 3 - Imogen



Imogen uses a variety of nodes to generate noise and combine it. The user interface seems easy to use, but the lack of customisation of the nodes looks limiting. The project has also not been maintained since 2019. The idea of having a small thumbnail for each layer looks helpful for productivity. I think I will add an option to show a preview for each layer.

### Essential Features

## Limitations

## Hardware And Software Requirements

## Success Criteria

# Design

## Decomposition

## Explanation of the process

## Solution structure

## Iterative development

### Example

#### Introduction

#### Algorithm

#### Variables and data structures

#### Identify test data

#### Coding

#### Testing

#### Failed test fixes

#### Evaluation

## Post development testing

### Testing

### Usability testing

# Evaluation

## Success criteria

## Unmet criteria

## Usability features

## Maintenance issues

## Limitations