SOUTHAMPTON SOLENT UNIVERSITY

BSc (Hons) Computer Games (Software Development)

Academic Year 2017 – 2019

Terrain Generation Report

Flavio Fiori

Supervisor: Philip Alassad Date: February 2019

Contents

[Background 3](#_Toc1295033)

[The reason why I want to undertake this project 3](#_Toc1295034)

[How this project is going to challenge me 3](#_Toc1295035)

[Feasibility and Risk 3](#_Toc1295036)

[Software 3](#_Toc1295037)

[Research and Prototyping 3](#_Toc1295038)

[Project Aims and Objectives 4](#_Toc1295039)

[Project Specification 5](#_Toc1295040)

[Potential Solution 5](#_Toc1295041)

[Project Management 5](#_Toc1295042)

[Initial Gant Chart 5](#_Toc1295043)

[Initial Work Break Down Structure 6](#_Toc1295044)

[Appendix 6](#_Toc1295045)

[A) Prototype using the 2D Perlin noise function that I created 6](#_Toc1295046)

[B) Prototype using the Unity Perlin Noise Function 6](#_Toc1295047)

[C) Perlin Noise from my function 7](#_Toc1295048)

[D) Perlin Noise from unity 8](#_Toc1295049)

[E) Ken Perlin’s original gradient function 8](#_Toc1295050)

[F) Riven Calculate Perlin Noise twice as fast 8](#_Toc1295051)

[G) Noise Displaced super quadric with old interpolants 9](#_Toc1295052)

[H) Noise Displaced super quadric with new interpolants 9](#_Toc1295053)

[Reference 10](#_Toc1295054)

# Background

The goal is to create a procedural generation of terrain, with a feasible scale. Randomly generate a smooth transition from mountains to valleys, simulating a terrain and applying textures in relation of the type of terrain. This process will be done using heightmaps.

## The reason why I want to undertake this project

Terrain generation is something that I find fascinating. Due to this reason I find that undertaking this project not only is going to drastically improve my knowledge in Unity and C# but also is going to increase my job opportunities in the game industry. This project can also increase the probability to work for *Jagex*, this is a company that I want to apply after my studies.

## How this project is going to challenge me

Programming a random terrain generation using high maps, is going to be one of the biggest challenges for me. Since I do not have an idea where to start. Long hours of study and programming are going to be required for me to complete this project.

## Feasibility and Risk

To prevent any risk from occurring I am going to use time management techniques like pomodoro and Kanban (Cirillo 2006; Cummings 2018; Cirillo 2017; Marcus Hammarberg and Joakim Sundén 2014; Anderson 2010; Peterson 2015).

I believe that creating a terrain generator with high maps is a feasible project, for avoiding any risks I will make sure that I work at least 30 hours per week in the project.

## Software

The required software needed is Visual Studio.

## Research and Prototyping

When developing a prototype of the procedural terrain, the premade Unity Perlin Noise function was used to generate a noise map and after converting it to a colour map ([Appendix B](#_Prototype_using_the)), this allowed to get an idea of the result of the prototype.

During the research was discovered that the original Perlin Noise created in 1980 had two deficiencies in the algorithm. One of them was the gradient computation, to fix this problem Ken Perlin discovered that was not necessary the G equation to be random at all. In the journal Improving Noise the G function was replaced by 12 vectors defined by the directions from the centre of a cube to its edges instead of the original G equation that take each grid point and assigns a pseudorandom vector of length 1 in R^2 (Perlin 2002). The other deficiency was found in the function to ease the curve, see ([Appendix G](#_Noise_Displaced_super)) ([Appendix H](#_Noise_Displaced_super_1))

“The above algorithm is very efficient but contains some deficiencies. One is in the cubic interpolant function's second derivative 6-12t, which is not zero at either t=0 or t=1. This nonzero value creates second order discontinuities across the coordinate-aligned faces of adjoining cubic cells.”

-Ken Perlin

With the prototype in hand, now was time to implement the Perlin Noise function. This function was based on the java reference implementation of improved noise function with slightly modifications.

When creating the noise function, few problems raised, one of them is the original Ken Perlin’s gradient function using complicated, and confusing bit-flipping code to calculate the dot product of a randomly selected vector and the 8 location vectors ([Appendix E](#_Ken_Perlin’s_original)) (Perlin 2002).

An easy alternate way of writing the code and even fast to compile in many languages, is to write a switch statement with all the 16 possible cases from the 8 location vectors ([Appendix F](#_Riven_Calculate_Perlin)) (Riven 2010).

Is possible to see the first result of the 2D Perlin noise function that was implemented in the ([Appendix A](#_Prototype_using_the_1)). While this function was far from perfect due to the fact that sometimes the output was negative, the result was satisfactory. To improve the function it was necessary to add the third axis (Z), to create a the correct output value ([Appendix C](#_Perlin_Noise_from)) similar to the original Unity function ([Appendix D](#_Perlin_Noise_from_1)).

## Project Aims and Objectives

The purpose of this project is to create a procedural terrain generation, using a terrain synthesis technique to create height maps. When the terrain generation is complete, if time allows me, I want to implement a method for synthesis of eroded terrain (Olsen 2004).

* Research the best algorithms to create procedural terrain.
* Implement one of the three algorithms to create procedural generated terrain:
* Perlin Noise
* Diamond-Square
* Simplex Noise
* Create a 3D wireframe in unity using the algorithm selected
* Generate a 2D height map using one of the algorithms above.
* Implement an algorithm for eroded terrain
  + Thermal erosion
  + Hydraulic erosion

## Project Specification

The product is going to be a complete program created in Unity to create a procedural terrain generation. The goal is to create a terrain with simulations of nature to create a realistic outcome.

For this project the following objectives were defined:

* Implement a 2D terrain generation prototype.
* Implement a full 3D terrain prototype
* Implement a method to simulate erosion on the terrain.

### Potential Solution

When implementing the first prototype, the Mathf.PerlinNoise function from unity was used (Appendix B).

The next solution was to implement the Perlin Noise function from scratch. If this function does not perform well for the procedural terrain generation, two algorithms can be implemented as an alternative. The simplex noise or Diamond-Square Algorithm.

## Project Management

Planning a project beforehand is a critical process to ensure that the project is going to be completed successfully. To ensure that everything goes as planned, a methodology named “Time-Box” is going to be applied to the project.

At the start of each week, a Gant Chart and a work break down structure is going to be created, this will be called a sprint. The risks will be managed at the end of each sprint.

### Initial Gant Chart

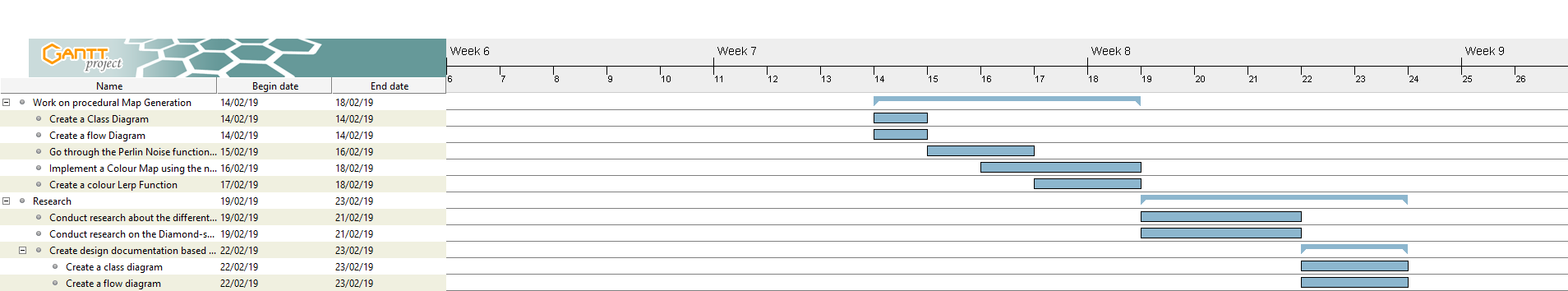


Figure 1 – Weekly Gant Chart using the Time-Box methodology

### Initial Work Break Down Structure

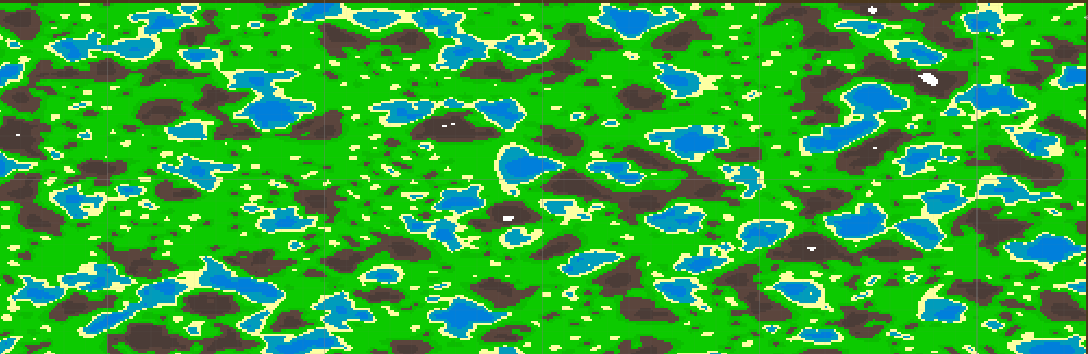
Week 13/02/2019 – 23/02/2019

WBS

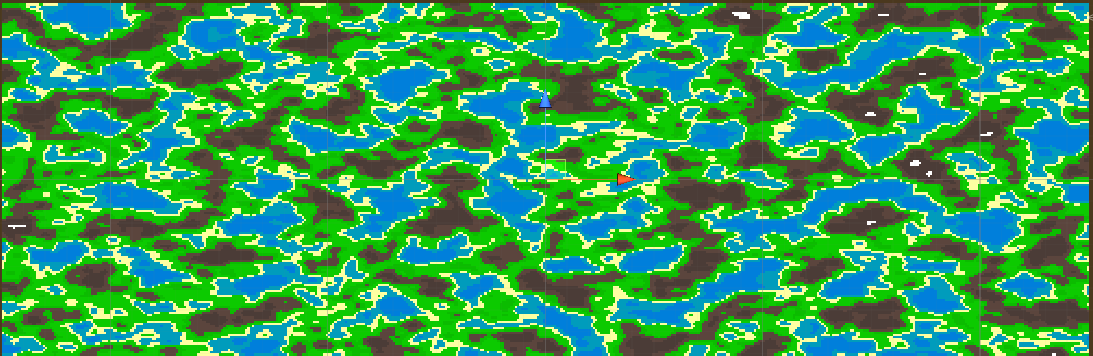
1. Work on the Procedural Map Generation (50%)
   1. Create a Class Diagram (5% Design)
   2. Create a flow Diagram (5% Design)
   3. Go through the Perlin Noise function done last week, and check for improvement (10% Programming)
   4. Implement a Colour Map using the noise map (20% Research/Programming)
   5. Create a colour Lerp Function (10% Research/Programming)
2. Research (50%)
   1. Conduct research about the different techniques to improve Procedural Terrain Generation (20% Research)
   2. Conduct research on the Diamond-square algorithm (20% Research)
   3. Create design documentation based on the research (10% Design)
      1. Create a class diagram (5% Design)
      2. Create a flow diagram (5% Design)

# Appendix

## Prototype using the 2D Perlin noise function that I created



## Prototype using the Unity Perlin Noise Function



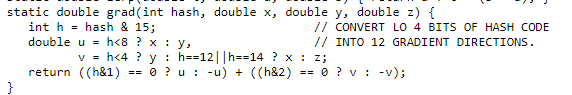
## Perlin Noise from my function



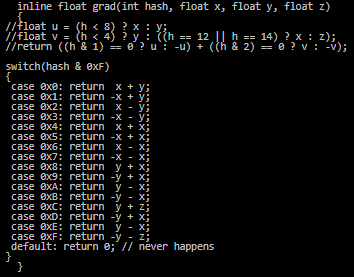
## Perlin Noise from unity



## Ken Perlin’s original gradient function



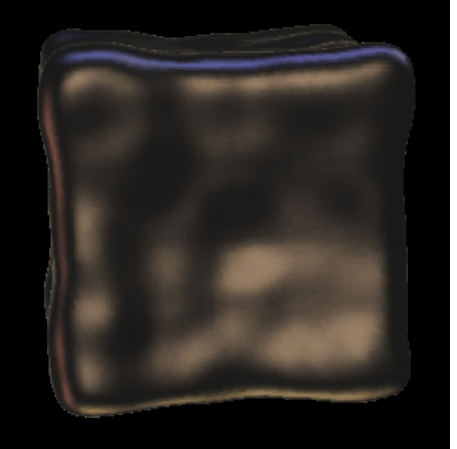
## Riven Calculate Perlin Noise twice as fast



## Noise Displaced super quadric with old interpolants



## Noise Displaced super quadric with new interpolants



# Reference

ANDERSON, D.J., 2010. Kanban. Sequim, Wash: Blue Hole Press

CIRILLO, F., 2006. The Pomodoro Technique. [viewed 02/05/ 2018]. Available from: <http://baomee.info/pdf/technique/1.pdf>

CIRILLO, F., 2017. Do more and have fun with time management. [viewed 02/05/ 2018]. Available from: <https://francescocirillo.com/pages/pomodoro-technique/book/>

CUMMINGS, T., 2018. The Pomodoro Technique: Is It Right For You? [viewed 02/05/ 2018]. Available from: <https://www.lifehack.org/articles/productivity/the-pomodoro-technique-is-it-right-for-you.html>

MARCUS HAMMARBERG and JOAKIM SUNDÉN, 2014. Kanban in Action. Manning Publications

OLSEN, J., 2004. Realtime Synthesis of Eroded Fractal Terrain for Use in Computer Games. Department of Mathematics And Computer Science

PETERSON, D., 2015. What is Kanban? [viewed 02/05/ 2018]. Available from: <http://kanbanblog.com/explained/>

PERLIN, K., 2002a. Improved Noise reference implementation [viewed 13/02/ 2019]. Available from: <https://mrl.nyu.edu/~perlin/noise/>

PERLIN, K., 2002b. Improving noise. ACM Transactions on Graphics (TOG), 21(3), 681-682

RIVEN, 2010. Calculate PerlinNoise faster with an optimization to grad(). In: August, Available from: <http://riven8192.blogspot.com/2010/08/calculate-perlinnoise-twice-as-fast.html>

ZUCKER, M., 2001. The Perlin noise math FAQ [viewed 10/02/ 2019]. Available from: <https://mzucker.github.io/html/perlin-noise-math-faq.html>