Project Plan

DV1478 Kandidatarbete i datavetenskap

2017-04-10

Thesis	Tentative title	Procedural city generation viable in games
	Classification	Theory of computation, Randomness, geometry and discrete structures, Computational geometry
Student 1	Name	Elias Frank
	Email	elias.frank@gmail.com
	Social security number	19940327–4179
Student 2	Name	Niclas Olsson
	Email	ohlsson.niclas@gmail.com
	Social security number	19930421–3797
Supervisor	Name	Hans Tap
	Email	hans.tap@bth.se

1: Introduction

Exploring a huge open world environment is a desirable feature in games. But creating a big open city such as in the *Grand Theft Auto*[1] series, and *Batman: Arkham City*[2] involves years of work for a lot of people. Making big open cities in games is simply not feasible for most game companies. These games all have massive success with their big open worlds making a feasible generated city an attractive technique for smaller companies to be able to compete with big open worlds of their own.

To create massive amounts of content without a big workforce there exists algorithmically based solutions, *Procedural Content Generation* (PCG). PCG was in the past used to minimize the disk space required for games. *.kkrieger*[3] is an excellent example of this. It has since evolved into a method to minimize workforce required for content. *No man's sky*[4] is an excellent example of a game using PCG to minimize workforce while maximizing content.

In this work, we want to explore the possibility of procedurally generating a city viable to use in games. When measuring the performance of procedurally generating content, impact on the following resources will be measured:

- Primary and secondary memory (RAM, disk space).
- Video memory (VRAM).
- Loading times.
- Repetition in patterns.

After the initial research into what PCG technique(s) to use, a budget will be specified for each resource. This budget will match system requirements for contemporary games.

There are many techniques to achieve PCG such as ray marching, squarified treemaps[5], perlin noise, fractals and many others. Part of this work will be to research previous work on the subject[6][7][8][9][10][11] and evaluate what technique(s) works best for our purpose.

2: Aim and objectives

- Research and explore what techniques to use when procedurally generating a city.
- Find a way to generate a city viable to use in games.
- Explore what constraints a city must have to be viable in games.
- Research what parameters and steps to use in the procedural generation of cities.
- Collect relevant data for analysis to answer the research question.

3: Research question

- Can PCG techniques be combined in a hierarchical manner to procedurally generate a city that is viable in games according to set constraints a real-time application such as games have?

4: Method

The method used to answer these questions will be studying and researching of PCG techniques along with an implementation of said techniques to generate a city viable in games. This implementation will be able to generate cities according to a set of variables. The performance of the implementation will be measured with different input resulting in generation of different cities. The result of these measurements will be compared against the specified resource constraints. If a result is within limits, it is considered viable.

4.1: Constraints

The focus of this thesis will be on the PCG techniques and if they produce viable results for games, with this in mind we have the following constraints.

- Do not create our own models for any property in the implementation.
- Do not implement optimizing techniques for the rendering pipeline.
- Limit content generation to the city (i.e. no terrain generation etc.).
- Do not implement property generation (i.e. no cars or street signs etc.)

4.2: Early implementation plan

To generate a city viable to use in games, three different generation stages have been recognized. The city will contain houses that will be arranged in blocks connected with roads.

4.2.1 District generation

For more variety and logical placement of the houses in the city districts will be generated. A city may have several district areas covering the whole city. Districts may have different kinds of houses in them. For example, there may not be big skyscrapers in a poor district.

4.2.2 Block generation

Blocks and roads needs to cover the entire city. The roads need to connect the whole city and the blocks may not have anomalies such as the block is in no way connected to any road. The blocks may have some constraints in their form and size.

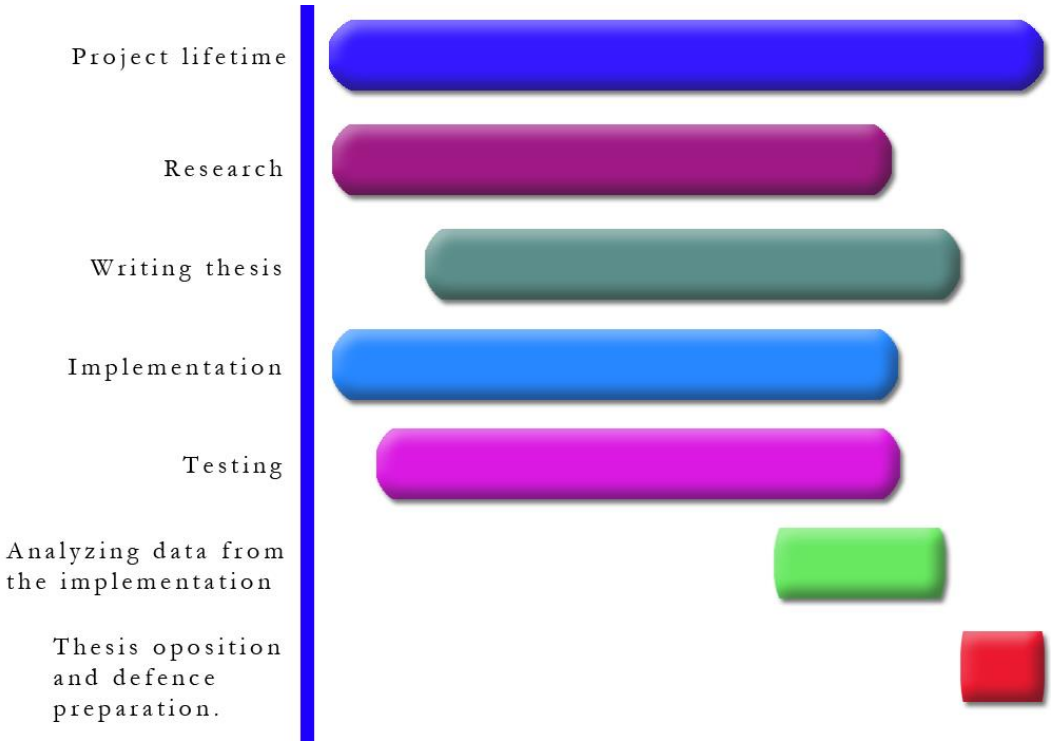
4.2.3 House generation

The houses will be generated to avoid repeating houses all over the city. This generation system will add different house-parts together creating a seemingly new house. This system will decrease the repeatability of the city creating a more organic city that hopefully is viable in games.

5: Expected outcomes

From the implementation, we will gather data to analyze and answer the research question.

6: Time and activity plan



The implementation and research will be the first things to be done and they will be done simultaneously. Some parts of the implementation require research before we have the knowledge to implement it, but all the parts of the implementation, such as the rendering pipeline, that do not require any research, will be done in conjunction with the research.

The writing of the thesis will begin shortly after we have some basic research and base implementation done. The thesis will be worked on every week until completion. When the implementation has enough features, data gathering and analysis of this data will begin. This data is crucial to answer the research question and conclude the thesis.

7: Risk management

Risk	Probability (1-5)	Severity (1-5)	Mitigation
Procedurally generating a city is difficult and consumes too much time.	3	5	Only implement the most crucial features to answer the research question.
Failing to communicate with thesis partner.	2	4	Have daily meetings and work together when possible.
Technical issues	2	5	Make sure all work is on several hard drives. Use git as source control.
Collect data from the implementation wrong, twisting the results.	3	4	Carefully decide what parameters in the implementation to collect data from and in what form to collect this data.
Defective construction of the implementation.	2	4	Before coding the implementation make sure there is a good plan to follow.

Referenser

[1] Rockstar Games. (1997-2013). *Grand Theft Auto Series.*

[2] Rocksteady Studios. (2011). *Batman: Arkham City.* Warner Bros, Square Enix.

[3] Farbrausch. (2004). *.kkrieger.*

[4] Hello Games. (2016). *No man's sky .* Hello Games, Sony.

[5] Fernando Marson, S. M. (2010). *Automatic Real-Time Generation of Floor Plans Based on Squarified Treemaps Algorithm.* Brazil: International Journal of Computer Games Technology.

[6] Carlos A. Vanegas, I. G.-D. (2012). *Inverse design of urban procedural models.* New York: ACM Transactions on Graphics.

[7] Markus Steinberger, M. K. (2014). *On-the-fly generation and rendering of infinite cities on the GPU.* Computer Graphics forum.

[8] Ubisoft Montreal. (2007-2015). *Assassin's Creed series.* Ubisoft.

[9] Muller, P. (2006). *Procedural Modeling of Cities Part VI.* Boston, Massachusetts: ACM SIGGRAPH.

[10] Müller, P. (2001). *Procedural modeling of cities.* New York: SIGGRAPH '01.

[11] Stefan Greuter, J. p. (2003). *Real-time procedural generation of `pseudo infinite' cities.* Australia: Graphite.