**Student name**

Joseph Barber

**Blog URL**

<https://josephbarberfinalproject.wordpress.com/>

**Proposed Final Project title**

An investigation into techniques for procedurally generating realistic cities.

**Brief outline of the work**

To begin with I will be prototyping multiple techniques for procedural city generation. This will include techniques for road generation such as L-systems and template-based generation, techniques for terrain generation such as noise or mid-point displacement, and, if time allows it, techniques for procedural building generation such as shape grammars and fractals.

After prototyping these systems, I will pick one method to move forward with and modify / extend upon in order to improve the method and implementation.

**Rationale**

Previously I have worked with some procedural algorithms for dungeon generation in a dungeon crawler game and found this to be a fantastic approach to building the levels, it allowed for more re-playability as the game will be different every time it’s played. I’m interested in further researching procedural generation techniques and this project allows me to do so.

I’d like to research and test whether using rules and restrictions can control the output of procedurally generated content in such a way that the output is both realistic and suitable for gameplay, and research whether the use of global bounds and local constraints makes this a more viable approach to creating large and diverse cities when compared to manually creating levels.

Employability statement – how this project will demonstrate relevant specialist skills

This project will allow me to demonstrate the ability to procedurally generate “random” content in a controlled way.

Algorithms such as Drunkard walk are fully-random methods of procedurally generating content, however, fully-random methods such as these have little-to-no rules or condition checks in place, this means that they are not well suited for making realistic and believable worlds, nor do they ensure that they are appropriate for gameplay. This project allows me to demonstrate the ability to restrict and control the random elements in such a way that the content is created sensibly, and a realistic output is created.

I’ll also be demonstrating my ability to research, pick up and learn new systems and methodologies, ability to problem solve and ability to work with C++. These are all valued skills in many job listings I have researched.

Examples of specific organisations and/or jobs in the relevant industry sector in which these specialist skills will be of value:

|  |  |  |
| --- | --- | --- |
| **Company - Job title** | **Relevant essential or desired skills/experience** | **Link** |
| Games Programmer | * Solid C/C++ programming skills * Excellent problem solving ability | [http://jobview.monster.co.uk/ Games-Programmers-All-levels- AAA-games-studio-Job-North- West-NW-UK-188160257.aspx](http://jobview.monster.co.uk/Games-Programmers-All-levels-AAA-games-studio-Job-North-West-NW-UK-188160257.aspx) |
| Utswo - Junior Programmer (Games) | * Experience creating things in C/C++ or C# | [https://www.indeed.co.uk/ viewjob?jk=088e20a14735926a &q=Games+Programmer+ C&tk=1bqg26g55145r51m](https://www.indeed.co.uk/viewjob?jk=088e20a14735926a&q=Games+Programmer+C&tk=1bqg26g55145r51m) |
| Ubisoft - Game Systems Programmer | * Write clear and well-structured C++ code * Excellent C/C++ programming skills, with good knowledge of object oriented development * Able to plan and estimate their own tasks and ensure timely delivery of work * Self-motivated and pro-active with a strong work ethic | [https://jobs.smartrecruiters.com /Ubisoft2/743999659938169-game-systems-programmer-011-](https://jobs.smartrecruiters.com/Ubisoft2/743999659938169-game-systems-programmer-011-) |
| Ubisoft – Gameplay Programmer | * Proficient in C# / Java / C++ with a good understanding of object oriented development * Knowledge of common algorithms, design patterns and structures * Can breakdown and estimate tasks accurately | [https://jobs.smartrecruiters.com /Ubisoft2/743999656454889-gameplay-programmer?codes=1-INDEED](https://jobs.smartrecruiters.com/Ubisoft2/743999656454889-gameplay-programmer?codes=1-INDEED) |
| Splash Damage – Game Programmer | * Knowledge of C++ * Good sense of software architecture, design patterns, code quality | <https://www.indeed.co.uk> /viewjob?jk=590ce15437232e27 |

**What do I wish to be marked on for the final project?**

The quality of my written code:

* How well is the final technique implemented?
* Are there clear improvements that can be made?
* Is the code well commented/documented?
* Is the code efficient?
* Is the code maintainable?

Flexibility of my final generation technique / Scalability of the cities:

* Can the final system produce cities of varying sizes?
* Can the system be passed different sized information maps (height, population) and produce appropriately sized cities?

Diversity of the generated city:

* Is there a lot of repeating / reoccurring themes?
* Are buildings unique or are they often repeated?
* Is it obvious the city was generated and wasn’t man made?

**Project timeline and Milestone deliverables**

Deliverables

As a bare minimum, I would like to have a system that can generate and output a road map which abides by global bounds and local constraints. This road map would consist of major and minor roads, and would also contain information of where buildings would be placed.

By the end of the project I would like to have tested multiple methods of procedurally generating road networks and building placement and chosen one to move forward with and refine. User input may be fed into this in the form of maps such as population density maps and water boundary maps.

If I had infinite time, I would like to produce a system that could work with or without user input. If given no input, it could create its own maps using techniques such as noise or fractal displacement. Major roads would be further refined to use elevation maps to follow the path of least elevation. All buildings would also be procedurally generated. These would be exported in a suitable form (.obj). This system could be compiled and exported into existing engines such as Unity or the city could be rendered and viewed within the system itself.

Timeline

|  |  |  |
| --- | --- | --- |
| **Project Milestones** | | |
| **Week commencing** | **Week #** | **Milestone Deliverables and Tasks** |
| September 25 | Week 1 |  |
| October 2 | Week 2 |  |
| October 9 | Week 3 |  |
| October 16 | Week 4 | *Road Generation – Prototype 1* |
| October 23 | Week 5 |  |
| October 30 | Week 6 |  |
| November 6 | Week 7 | **Submission of Final Project Proposal: by noon, Friday 10 Nov 2017** |
| November 13 | Week 8 |  |
| November 20 | Week 9 |  |
| November 27 | Week 10 | *Road Generation – Prototype 2* |
| December 4 | Week 11 |  |
| December 11 | Week 12 |  |
| December 18 | **Mid-Winter Festival** | |
| December 25 |
| January 1 |
| January 8 |  | *Road Generation – Prototype 3* |
| January 15 |  |  |
| January 22 | Week 13 | *Reading in map data (water boundary, height map)* |
| January 29 | Week 14 |  |
| February 5 | Week 15 | **Seminar Presentations TBC** |
| February 12 | Week 16 | **Seminar Presentations TBC** |
| February 19 | Week 17 | *Procedural generation of terrain* |
| February 26 | Week 18 |  |
| March 5 | Week 19 |  |
| March 12 | Week 20 |  |
| March 19 | Week 21 | *Procedural building generation* |
| March 26 | **Spring Fertility Festival**  *(Code and implementation polish)* | |
| April 2 |
| April 9 | Week 22 | **Submission of Final Product and Blog: by non, Friday 13 Apr 2018** |
| April 16 | Week 23 |  |
| April 23 | Week 24 |  |

**Annotated bibliography**

Initial reading and research:

|  |  |  |
| --- | --- | --- |
| **No.** | **Item** | **Description / Annotation** |
| 1 | Brown, B. (2011). Prim's Algorithm: Minimal Spanning Tree. [Online] 15th May 2011. Available online: <https://www.youtube.com> /watch?v=YyLaRffCdk4 [Date of access: 02 October 2017]. | This is a video which explains Prim’s Algorithm which is an algorithm to create a minimum spanning tree from a weighted graph; this is one method I use of procedural road generation. |
| 2 | Chen G. Esch G. Wonka P. Mueller P. Zhang E. (2008). "Interactive Procedural Street Modeling" In Proceedings of SIGGRAPH 2008. ACM Trans. Graph. Article 103: 1-10. | This paper discusses the use of tensor fields and focuses more on user interactivity. It talks about a system which creates a tensor graph and allows the user to edit/adjust this graph to edit the generated city. |
| 3 | Eppstein, D. (1996). Design and Analysis of Algorithms – Minimum Spanning Trees. [Online] 6th Feburary 1996. Available online: <https://www.ics.uci.edu> /~eppstein/161/960206.html [Date of access: 29 October 2017] | This item details multiple algorithms for creating a minimum spanning tree including Prim’s algorithm and Kruskal’s algorithm. It includes pseudo code and explanations of each algorithm. |
| 4 | Evans, M. (2015). Procedual Generation For Dummies: Road Generation. [Online] 11 December 2015. Available online: http://martindevans.me/game-development/2015/12/11/Procedural-Generation-For-Dummies-Roads/ [Date of access: 19 May 2016] | This article talks about the use of tensor fields in road generation. It also talks about the use of global goals and local constraints when creating major and minor roads and shows how they created road networks using different road templates (radial, grid). |
| 5 | Goldman, R. Schaefer, S and Ju, T. (2004). Turtle geometry in computer graphics and computer-aided design. *Computer Aided Design* 36(14), pp.1471-1482 | This paper discusses and over views the turtle graphics system, implementations of turtle graphics are often used alongside L-systems to visualise procedurally generated content. |
| 6 | Greuter, S. Parker, J. Stewart, N. Leach, G. (2003). "Real-time procedural generation of 'pseudo infinite' cities". In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Astralasia and South East Asia* (GRAPHITE '03) Acm, New York, Ny, USA, 87 - ff. | This paper focuses on creating cities with a diverse range of buildings, and thus focuses on building generation and not generating the city layout. It proposes a method of building generation by splitting the city up into cells and using a hash function to create a number for each cell to be used as seed. This seed determines the properties of the buildings in the cell (number of floors, height, number of shapes, etc). The building is created by creating and placing several shapes together within the confined space of the building and then extruding each shape to the height of a specific floor. This creates a diverse range of buildings. |
| 7 | Havey, D. (2008). Tutorial #7: Voronoi diagrams\* [Online] May 4, 2008. Available online: <http://donhavey.com/blog/tutorials> /tutorial-7-voronoi-diagrams/ [Date of access: 30 May 2017] | This article talks about Voronoi diagrams, their use in creating real world maps and then goes on to talk about how these would be implemented, the logic behind how they’re created and what condition checking is necessary. |
| 8 | Ilangovan, K, P. (2009) Procedural City Generaror, MSc thesus, Bournemouth University. Available at: <https://nccastaff.bournemouth.ac.uk> /jmacey/MastersProjects/MSc09/ Ilangovan/Thesis\_i7834000.pdf (Accessed: 30 May 2017). | This paper talks about some techniques used for city generation such as L-systems, however this paper is useful to me as it talks about terrain generation; it takes grey scaled maps or contour maps as input and it uses these to create its own heightmap and can use this to create its own terrain. This is useful for me as I can use these techniques to find out where water will be in the world, and can also use these same techniques and treat them as a population density map which will define where my major roads are placed. |
| 9 | Kelly, G and McCabe, H. (2006). "A Survey of Procedural Techniques for City Generation". In *ITB Journal, No. 14*. | This paper starts by introducing multiple procedural techniques such as fractal, l-systems and noise. It goes on to evaluate these techniques in city generation based on criteria such as realism, scale, input, efficiency, etc. |
| 10 | Kelly G and McCabe H. (2006). “Interactive generation of cities for real-time applications” In Proceedings of *Siggraph ’06 ACM SIGGRAPH 2006 Research Posters* – SIGGRAPH ’06. |  |
| 11 | Martek, C. (2012). "Procedural generation of road networks for large virtual environments." Rochester Institute of Technology. | This paper proposes some new techniques for procedural road generation. It uses population data and places points on the highest populated areas, it makes connections between these points using a minimum spanning tree and then creates the connections using an A\* implementation that uses elevation as a cost. |
| 12 | Martz, P. (1997). *Generating Random Fractal Terrain*. [Online] 1997. Available Online: <http://www.gameprogrammer.com> /fractal.html [Date of access: 09 May 2016] | This article speaks about creating fractal two-dimensional and three-dimensional terrains, and later goes onwards to show how these can create height maps. |
| 13 | Muller, P and Parish, Y, I, H. (2001). "Procedural modeling of cities". In Proceedings of the 28th annual conference on Computer graphics and interactive techniques (SIGGRAPH '01). Acm, New York, NY, USA, 301 - 308. | In this paper, the authors present CityEngine which is a system capable of procedurally generating cities using user-controlled input data such as height maps and population density maps. They talk about using L-Systems and road patterns for creating the city layout and go on to talking about procedural building geometry (buildings) and textures. |
| 14 | Muller, P. Wonka, P. Haegler, S. Ulmer, A. Goo, L, V. (2006). "Procedural modeling of buildings" In Proceedings of SIGGRAPH 06' ACM SIGGRAPH 2006. Acm, New York, NY, USA, 614 - 623. | This paper focuses on the procedural modelling and texturing of buildings. It talks about the use of shape grammars in procedural building creation. When it comes to procedural texturing, it talks about using occlusion for checking for intersections between shapes and then uses snapping to avoid texturing errors that previous procedural texturing methods have faced such as having a window placed where an intersection of the building occurs. |
| 15 | Prusinkiewicx, P. and Janan, J. (1992). “Lindenmayer systems, fractals, and plants.” New York, N.Y.: Springer-Verlag. | The beginning chapters of this book explain many techniques of procedural generation useful to city generation including Re-writing systems, fractals, L-systems and Bracketed L-systems |
| 16 | Red Blob Games. (2014) *Introduction to A\*.* [Online] June 2016. Available online: <http://www.redblobgames.com> /pathfinding/a-star/introduction.html [Date of access: 29 September 2017]. | This page covers the Breadth First search, Dijkstra and A\* path finding algorithms. A\* is a path finding technique I will be using for road generation of an MST to follow the path of least elevation. |
| 17 | Sun, J. Yu, X. Baciu, G. Green, M. (2002). "Template-based generation of road networks for virtual city modeling". In *Proceedings of the ACM symposium on Virtual reality software and technology* (VRST '02). Acm, New Work, NY, USA, 33 - 40. | This paper talks about various road-templates such as population-based, raster and radial which are used in template-based generation. It explains each of these templates and goes onwards to talk about validity control in the creation of major and minor roads. |