Initial Plan

Procedural Voxel Planet Generation

Author
George Kenny

Supervisor
Dr Frank C Langbein

Project Description **3**

Project aims and Objectives **5**

Aims 5

Objectives 5

Work Plan **6**

References **8**

Project Description

The majority of 3D games use a heightmap to store terrain data as they are efficient to manipulate and easy to render however they lack the ability to produce overhangs, cliffs, caves and many other terrain features (Greef 2009). One way to tackle this problem is to use voxel terrain instead. Voxel terrain allows you to create a volumetric terrain which can be used to enhance gameplay and can feature more interesting terrain.

Heightmap terrain systems typically use displaced planes which are fine for most games however for space themed and science fiction based games the ability to travel off world to other planets or just from planet to space is a sort after feature but if a game uses a heightmap terrain system then this typically involves loading in a separate level and trying to hide the loading through a variety of tricks. One way to have seamless transitions from planet to space would be to produce a spherical planet with terrain. This way a new level would not need to be loaded and it could all take place in a single world.

By combining planetary terrain with a voxel terrain system it is possible to create an interesting and realistic looking planet and create interesting gameplay features such as the ability to dig through the planet and flying from space down onto the planet seamlessly. This could be further extended by implementing more advanced terrain techniques such as erosion, rivers, caves and populating the terrain with vegetation. Finally, procedurally texturing the planet will give it a much more realistic look and this can also be further enhanced with different shader techniques. By making the planets procedurally rather than by an artist lots of time can be saved during development of a game as it would be possible to quickly generate new content. The downside to using voxel terrain for this is that a level of detail system is far more complex to implement (Lengyel 2010) which is vital for planets due to the large number of triangles.

Rendering large terrains or planets due to the number of triangles being rendered can quickly lead to performance issues so tackling these will be an important objective. There are a variety of techniques to handle this such as using chunks and compute shaders to speed up processing. One of the aims of the project is to be able to render the planets in real time at a high number of frames per second so that the planets will be usable in real time projects such as games.

I intend to create a tool that generates realistic voxel planets that can be rendered in real time and contain advanced terrain features. The main focus of the project is the realistic terrain generation rather than the isosurface extraction as the project isn't long enough to focus on both to the extent that I want to. However the isosurface extraction technique and how it is implemented is very important for the real time performance of the project.

The project will be developed using Unity game engine as it provides an easy way to render meshes and a simple to use API that will save time during development which would otherwise have to be spent on writing OpenGL or directX code. However, it is still very flexible so I can still write shaders and implement everything I wish.

Project aims and Objectives

Aims

- To create a tool for Unity game engine that can generate procedural voxel planets.
- To create a rendering system for Unity game engine that can render large planets at a fast enough frame rate for real time applications.

Objectives

- Planets should look earth like i.e. Have mountains, valleys, oceans etc.
    - Use water erosion to create more lifelike terrain
    - Layer different octaves of noise to create different terrain features
- Planets should be textured
    - Triplanar texturing
    - Procedural texturing
- Planets should have an atmosphere
    - Atmospheric shader
- Should be able to render a planet at 60 frames per second so they should be optimised.
    - Level of detail system
- Use of an appropriate isosurface extraction method
    - Marching cubes
    - Dual contouring
- The tool should have an easy to use editor UI to customise parameters.

Work Plan

Due to the relatively short length of the project, the work plan follows a waterfall approach for development with the essential features being worked on early so if any issues arise they can be dealt with early. This should mean that all the core features are completed by the end of the project, even if that means some of the more advanced but not as needed features are not complete. The two deliverables for the project are the initial plan and the final report with the projects code. Each feature of the project is a milestone and they are ordered in such a way that prerequisites are finished first.

Week 1: Beginning 27/01/2020
I will be working on the initial plan which is this document and doing some background research on what will need to be done to solve the problem.
Deliverable: Initial Plan
Milestone complete: background research

Week 2: Beginning 03/02/2020
Researching different isosurface extraction techniques and other background information needed to generate planets. Begin implementing the chosen technique to render simple voxel structures within Unity game engine.

Week 3: Beginning 10/02/2020
Finish implementing the chosen isosurface extraction algorithm and render a basic planet with basic terrain using basic noise. Create basic UI to edit parameters allowing fast generation of different planets.
Milestone complete: Isosurface extraction implemented.

Week 4: Beginning 17/02/2020
Research how to create a more advanced terrain and implementing this.
Milestone complete: terrain generation implemented.

Week 5: Beginning 24/02/2020
1st review meeting with supervisor.
Research different texturing and shading methods and implement one of these. One method that could be used is triplanar texturing. Texture based biomes and height of terrain from the ground.
Milestone complete: terrain texturing.

Week 6: Beginning 02/03/2020
Implement a basic ocean and water shader. Begin to implement rivers.
Milestone complete: oceans.

Week 7: Beginning 09/03/2020
Finish rivers.

Focus on optimization and making sure that once rendered there is a good enough frame rate of at least 60 fps. Decide on a set of benchmarks or a benchmark planet to see the effect that optimization has had.
Milestone complete: rivers on the terrain.

Week 8: Beginning 16/03/2020
Continue with optimization. Use chunks to only render nearby chunks and research into level of detail systems to increase performance.
Milestone complete: optimization.

Week 9: Beginning 23/03/2020
2nd review meeting with supervisor.
Research atmospheric shaders and begin to implement one.

Week 10: Beginning 30/03/2020
Finish the atmospheric shader.
Milestone complete: atmospheric shader.

Week 11: Beginning 06/04/2020
This week will be used for adding more complex terrain features such as erosion if there is any time leftover. Otherwise some features may take longer than planned so leaving this week free will allow me to finish them.

Week 12 - 14: Beginning 13/04/2020 - Ending 03/05/2020
During the Easter break the final report will be written and benchmarks from the tool I have created will be taken and will be used in the final report to see how well the tool fits the aims and objectives detailed in this initial report.
Milestone complete: final report written.

Week 15: Beginning 04/05/2020
Proofread the final report and make any final changes. Prepare the work for submission and finally submit the final report and code before 07/05/2020.
Deliverable: Final Report and code

References

Greeff, G. 2009. Interactive voxel terrain design using procedural techniques. MSc Dissertation, Stellenbosch University

Lengyel, E.S. 2010. Voxel-based terrain for real-time virtual simulations. PhD Dissertation, University of California Davis