Building and Rendering Planetary Bodies With Plate Tectonics

Justin Powell
Taylor University

December 2015

1 Abstract

For the generation and rendering of planetary bodies there are several things that need to be taken into consideration. These main points are the terrain geometry, the terrain textures, and the rendering of the atmosphere. This paper focuses on the terrain geometry portion of planetary rendering. This involves the handling of the model geometry and how it is represented in the data structure as well as the generation of varied terrain heightmaps. The terrain is generated through a combination of Perlin Noise and a simple tectonic plate approximation and rendered using dynamic tessellation through the OpenGL tessellation shader.

2 Introduction

Planet and terrain generation is a long studied field of the graphics industry, for use in applications such as film, games, simulations, and mapping software. With this varied background of research there are many different methods of generating terrain that have arisen over the years. Methods such as the scanning of real-world terrain are popular in mapping and simulation software. In the game industry, Perlin noise has made great waves over the past few years with the help of games such as *Minecraft*[cite] as a means to generate large and seemingly infinite terrain on the fly. Others such as *No Man's Sky*[cite] and *Elite Dangerous*[cite] are other examples that procedurally generate large planetary bodies. From these different and interesting examples, the inspiration for this project came to light. We are interested in the process and challenges that go with generating these planetoids and rendering them in real time.

The main problem this project targets is the generation and rendering of planetary bodies for use in a real time solar system generator. This includes the generation of terrain meshes, texturing, and atmospheric rendering. Over

the summer I worked on the atmospheric rendering portion of the project and ended with a method for producing realistic looking atmospheres both quick to produce, and easily modifiable [3]. The main portion of the project, however, lies in the terrain generation and rendering for planetary bodies.

Simply generating this detailed terrain, however, is only half of the equation. The project requires the terrain to be rendered at interactive frame rates and within a reasonable amount of time. This means that we must keep the generation simple and quick, and dynamically modify the level of detail of the tessellation to keep the performance within interactive levels.

3 Previous Works

Two projects that have different and interesting approaches to simulating plate tectonics in the process of generating terrain heightmaps. First is a paper by Benedetti and Minto titled *Tectonic Plate Simulation on Procedural Terrain*[1] which does a simple simulation using two plates on a 2D plane. The second paper is a more complex simulation by Viitanen titled *Physically Based Terrain Generation: Procedural Heightmap Generation Using Plate Tectonics*[6]. The two go about the simulation in differing ways that vary in complexity which helped to formulate our solution.

Benedetti and Minto's [1] paper does a simple simulation between two tectonic plates that interact either through a convergent or divergent boundary. The plates are represented through two flat polygons that are then either pushed together or pulled apart. Using the positive or negative overlap of the polygons, the previously generated heightmap of the plate boundary is modified. This paper gave me some initial ideas as to what we wanted to implement in this project, however it would need to be scaled up and applied to a 3D sphere.

Viitanen instead goes a more complex route in which multiple plates are generated and simulated. Each plate has several attributes that help in the simulation such as density, velocity, mass, and whether it is an oceanic or continental plate. The simulation runs through each plate and applies the different variables to produce the final terrain heightmap. Boundary types

between two plates are determined in real time via utilizing the direction vectors of the plates as well as density differences of the plates. This process takes quite a bit of time depending on the size of the map and the number of plates.

Again considering this simulation was done with only a 2D plane, all the calculations would need to be reconsidered and adjusted to fit to a 3D sphere. This also described some limitations that arise when doing complex simulations that could further impact the design of this project.

4 Plate Tectonic Theory

Plate Tectonic theory is the idea that the Earth's lithosphere is broken into different tectonic plates that move about on the surface of the mantle. This theory has been widely studied and verified since its introduction in the early 20th century and explains the occurrence of many of Earth's geological features such as mountains, trenches, and islands.

As plates move around the surface there exists several different type of boundaries between neighboring plates depending on how they interact with each other. These boundaries are: convergent, divergent, or transform type boundaries.

Convergent boundaries can vary depending on if the converging plates are made from light, thick, continental or dense, thin, oceanic crust. In the case that two continental plates of about equal density converge we find that the two compete for dominance in which gets pushed under the other. This results in the two being pushed together and upwards rather than down, creating large mountain ranges such as the Himalayas. In the case of a continental-oceanic or oceanic-oceanic convergence the process has a slight difference in that the denser plate is pushed under less dense plate and back into the mantle. Once it reaches the mantle it begins to melt under the intense heat and pressure which results in magma plumes that eventually rise back to the surface to produce either volcanic islands or mountains. Examples of these types of boundaries are along the Japanese Island arc or the Andes Mountains in South America.

Divergent boundaries are the result of two plates moving in opposite directions from each other. Along the boundary forms what is known as a rift, a zone where magma rises to the surface to fill in the space left by the diverging plates. These rifts can either form as rift valleys in the case of a rift located within a continent or as mid-ocean ridges in the case of sea-floor rifts. Volcanoes often form along these boundaries producing either volcanic mountains or islands. Examples such as the East African Rift and the Mid-Atlantic Ridge show the process of continental and oceanic rifts.

The final type of boundary, transform boundary, is caused in the case that two plates slide past each other in a horizontal motion. Unlike convergent or divergent boundaries, which produce prominent terrain features such as mountains or rifts, a transform boundary does not produce many noticeable terrain features. A rather prominent example of this type of boundary is the San-Andreas Fault along the western edge of North America.

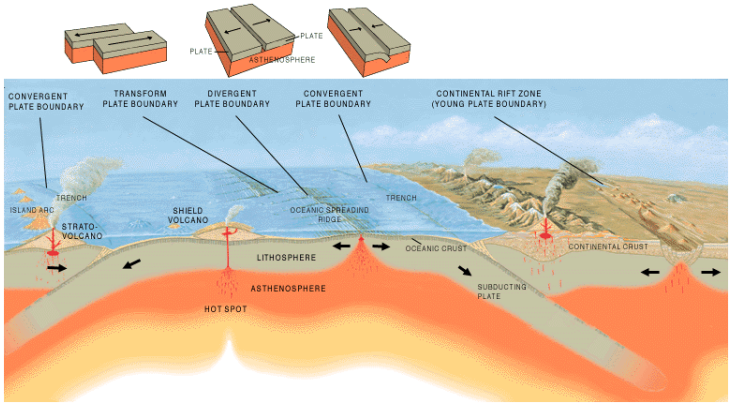


Figure 1: Tectonic plate interactions. [5]

5 Implementation

5.1 Considerations

Remember back to the goal for this project. We are looking to generate and render planetary bodies that can be used in a solar system simulation. Imagine we want to generate an analog to our solar system with 8 planets

and 168 moons. Each planet needs to have its own mesh and terrain generated and rendered in real time. Meaning when the program is started each planet's mesh is generated, tectonic plates assigned, and finally rendered to the screen. This is even before the actual simulation begins. Considering the large amount of bodies that need to be accounted for we don't have the time to get complex in our plate tectonic simulation.

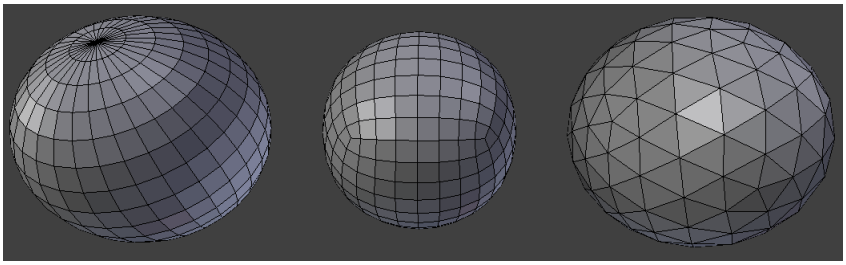
5.2 Planet Mesh Generation

When modeling a planetary body the most important thing to consider is the mesh used to represent the planet. How the data is stored in the model mesh will change how the mesh is acted upon and rendered. When creating the mesh to model our planet, that we will refer to as the "base mesh", there are three standard models from which our design was chosen. These three models are the UV sphere, the quad sphere (also known as a subdivided cube), and the icosphere. There were two main points that were considered when choosing a mesh design, first being the vertex distortion. Vertex distortion refers to the spacing of the vertices on the face of the sphere; how evenly they are spaced around the suface. The second point was how the faces are constructed and how that affects the adjacency information.

We settled on the quad sphere based mesh design for the terrain that can be seen in Figure 2. Of the many different ways you can represent a sphere this mesh provides both the simplest form for tessellation and neighbor traversal. For example: between a quad sphere design and a UV sphere design we find that the quad sphere has less distortion around the poles and around the face of the sphere in general leading to a more even distribution and spacing of vertices. This is important when we reach the stage of terrain generation. The distance between vertices governs the detail that can be packed into an area. If the faces of the mesh aren't roughly the same size then we are left with patches that are less detailed than others, leading to uneven terrain. An icosphere has even less distortion than a quad sphere, however it comes at the cost of a large increase in vertices and faces that make up the mesh.

To generate our base mesh we start with a simple cube. From there we tessellate the mesh multiple times before finally mapping each vertex to the

surface of a sphere. While tessellating this sphere increases the detail of the planetoid, it is also one of the greatest bottlenecks in our process. Each level of tessellation increases the number of vertices in our mesh by roughly four times, so extra consideration must be done before further increasing the tessellation.



(a) UV sphere mesh — Quad sphere — Icosphere

Figure 2: Note the concentration of vertices around the pole of the UV sphere and around the old corner of the cube of the Quad sphere.

5.3 Plate Assignment

Considering each planetoid is being generated at program startup, including base mesh tessellation and tectonic plate generation, we must make sure to keep load times short. For this reason we settled on a simple and fast method of generating tectonic plates.

Each planetoid is given a set number of tectonic plates. From here we randomly select a number of faces on the base mesh equal to the number of plates for that planetoid. From each starting face the tectonic plate grows outwards until every face has been assigned to a plate.

To increase the detail of each plate you simply increase the tessellation of the base mesh of the planet. This however will increase the base mesh generation time both because of the increased tessellation requirements and the increased number of faces which must be assigned.

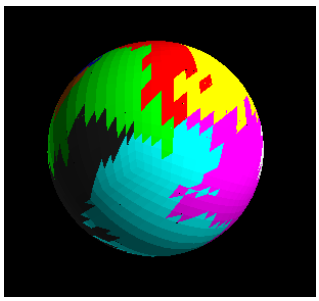


Figure 3: The base mesh after plate distribution.

Each plate is assigned an identifier which is in turn assigned to each vertex belonging to a face of that plate. This color is passed to the shader for use in generating the terrain detail.

5.4 Tessellating the Mesh

Our base mesh is not tessellated enough to accurately render detailed terrain. The closer to the surface of the planetoid we get the more we need the faces to be tessellated, however rather than relying on the CPU to tessellate our mesh we can offset the process to the graphics card to improve performance.

The geometry of the base mesh is passed to the graphics card in what are called "patches". This is to facilitate the tessellation of the mesh by the graphics card through the use of the Tessellation shader. In the OpenGL rendering pipeline there are several stages that execute one after another in rendering the objects sent to the graphics card. There are three important stages when tessellating geometry that are called the tessellation control, tessellation evaluation, and the geometry shader. These three work in conjunction to generate new primitives based on the parameters set in the control shader, modify the locations of these new primitives in the evaluation shader, and finally generate the new primitives in the geometry shader.

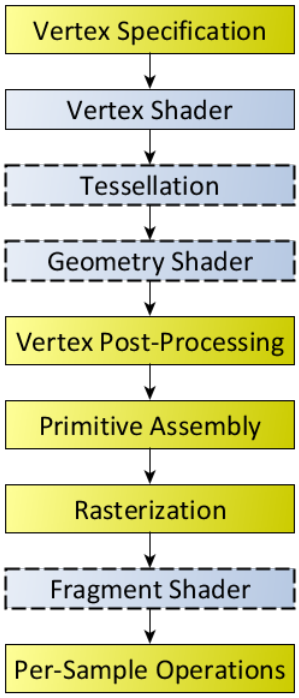


Figure 4: The OpenGL render pipeline. Note that the Tessellation shader and Geometry shader are optional steps.

Tessellation of the base mesh is handled by the tessellation control shader portion of the pipeline. It is in here that the tessellation factor is determined for each input patch based on the distance from the camera and whether or not it is visible.

Rather than rendering everything with the same level of tessellation a dynamic level of detail algorithm is used to reduce the amount of new geometry that must be generated and rendered each frame. When generating and rendering terrain the closer you get to the mesh, the more detailed you need to render it. However simply increasing the tessellation on all geometry arbitrarily will drastically reduce performance. To run at interactive speeds we need to render at least 60 frames per second. Thus the level of detail is modified during runtime based on the distance from the camera. The closer it gets, the more detailed it gets.

5.5 Visualizing the Planetoid

Finally, once we have our newly generated geometry we need to assign height values to each vertex to represent our terrain. This is done in the tessellation evaluation shader using a Perlin noise algorithm that is modified based on whether the vertex lies within a boundary zone or within the center of a tectonic plate. Each plate is seen by the shader as a specific color based on the id that is passed to the graphics card. To determine plate boundaries a simple blending function is used that takes colors of the plates and blends them together, producing a mixed area. The Perlin noise algorithm takes the colors of the plates into account when determining the variables that are input into the equation to produce the heightmap for the planet.

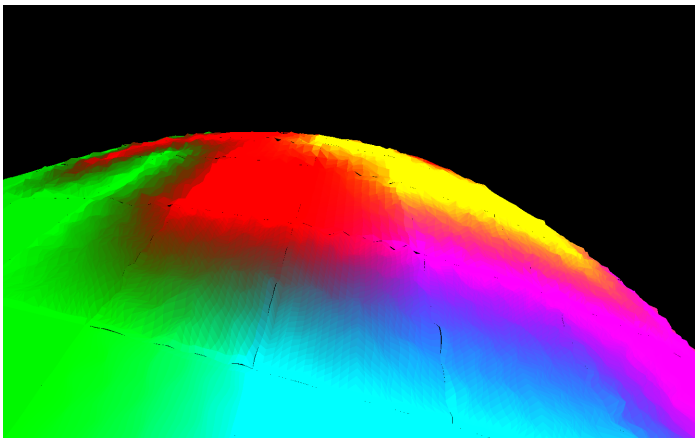


Figure 5: How the shader represents the tectonic plates. Blended area represent a boundary.

Along the blended border zones the algorithm increases the intensity of the generated terrain to produce mountains or rifts. However, within the center of the plates the intensity is rather smooth, which represents plains.

These newly generated vertices are finally passed to the geometry shader where they are assembled into new primitives to be sent off to the fragment shader.

6 Results

This project was run on an Intel i7 3610QM @ 2.3GHz, Nvidia 670m graphics card, and 16GB of memory. The planetoid generation times are within reasonable parameters as per the problem statement as can be seen in Figure 6. Due to the fact that the base mesh tessellation has the greatest effect on the time it takes to generate the mesh and assign plates it is recommended to keep it below 8 levels of tessellation.

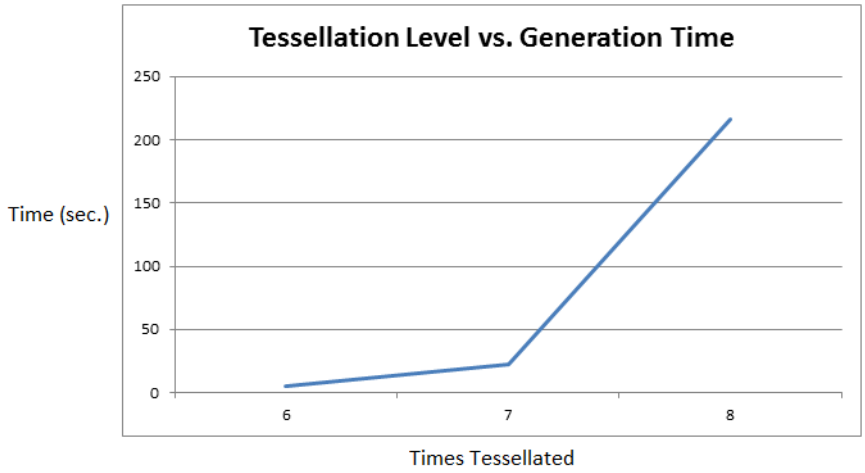
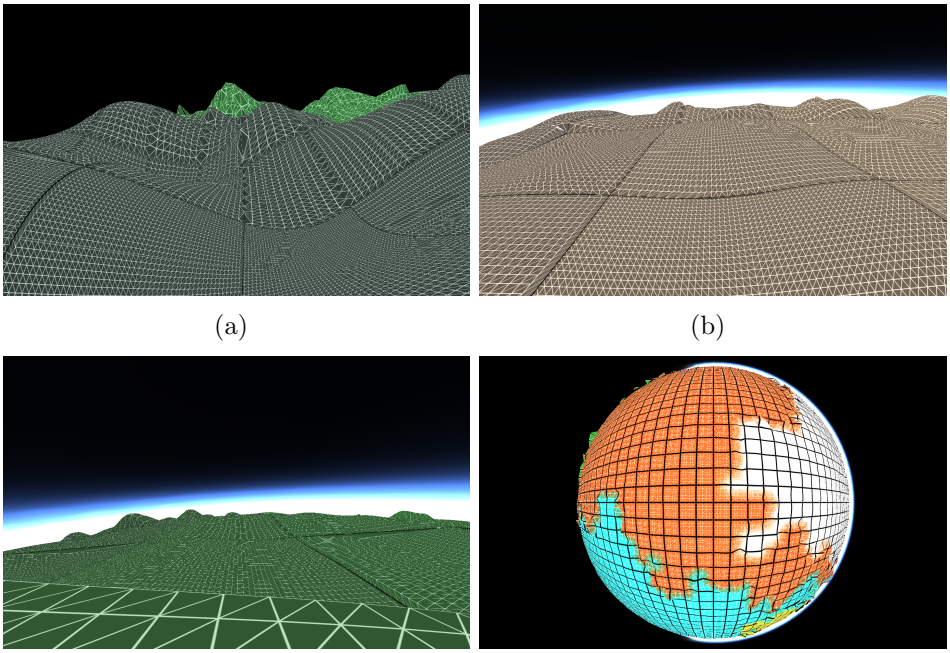


Figure 6: Base mesh tessellation level vs. planetoid generation time.

Utilizing this simple method of terrain generation a vast variety of planets can be generated quickly and on the fly. While at the moment the algorithms still need adjustment, they mark a good starting point for future improvement and refinement. Finally when used in conjunction with the atmospheric rendering code[3] the result is a decently realistic recreation of a planetary body that can be used in a solar system simulation as per the original goal of the project.



(a) (b)

(c) (d)

Figure 7: Examples of final results from both in atmosphere and in space.

7 Future Work and Conclusion

The are some limitations with this method of plate tectonics. Due to the structure of the mesh we only have a limited number of quad faces with which to generate tectonic plates. This leads to plates that look jagged and low detail. There is also a problem in the type of boundaries that are generated. At the moment all the work is done in the shader, meaning adjustments need to be made there to generate terrain that correlates to the different types of boundaries. Without careful tuning unnatural terrain can very quickly and easily throw everything into chaos. In the future it would be better to revisit both the structure of the mesh and the plate tectonics. Research into other mesh structures is needed.

Another limitation lies in the level of detail algorithm. Currently it needs fine tuning to further improve the performance gains as well as to fix several bugs that we've noticed. The algorithm does not work well with patch

edges and the dynamic changes in the mesh are quite noticeable. Either more tuning needs to be done on the algorithm to help eliminate inconsistencies between patches or another algorithm should be implemented.

The final limitation is on the load times. As the generation of the planetoid mesh is done on startup, generating enough to simulate our solar system could take several minutes. In the future we can reduce this time by storing pre-generated meshes to use as a base; this would allow us to cut out our largest bottleneck.

In conclusion this project succeeded in all its goals. As per the problem described, we have a method for generating planetoid bodies in real time and rendering them at interactive speeds. With some adjustment an infinite variety of planets and moons can be generated on the fly that can then be used in a simulation of a solar system. By integrating what was done in this paper with the research from over the summer we have a rendering of a planet including terrain and atmospheric scattering that was generated in real time and rendered in interactive speeds.

Bibliography

[1] Benedetti, Daniel and Evan Minto. *Tectonic Plate Simulation on Procedural Terrain.* Rensselaer Polytechnic Institute

[2] Boesch, Florian. "OpenGL 4 Tessellation." Codeflow. N.p., 7 Nov. 2010. Web. 16 Dec. 2015. <http://codeflow.org/entries/2010/nov/07/opengl-4-tessellation/>.

[3] Powell, Justin, 2015. *Selection of an Atmospheric Shading Model for Planetary Rendering* Taylor University

[4] Rideout, Philip. "Quad Tessellation with OpenGL 4.0." The Little Grasshopper. N.p., 6 Sept. 2010. Web. 16 Dec. 2015. <http://prideout.net/blog/?p=49>.

[5] Vigil, Jose F. Tectonic Plate Boundaries. Digital image.<https://commons.wikimedia.org/wiki/File:Tectonic_plate_boundaries.png>.

[6] Viitanen, Lauri, 2012. *Physically Based Terrain Generation: Procedural Heightmap Generation Using Plate Tectonics.* Bachelor of Engineering thesis, Helsinki Metropolia University of Applied Sciences.