Robert Inglin

Final Project Report

Megatexturing and Normal Mapping in Realtime

Introduction

The goal of this project was to create a large realtime environment that consists of a single large or mega texture. The 3d scene was portrayed using a single large scale texture that encompassed the entire environment. The height map data was procedurally generated using the diamond square algorithm. The normal maps were to be generated on load, And finally the user was to be able to traverse through the scene, while the texture sampled and reloaded the proper texture segments in real time. The portion of the graphics pipeline that this project focused on is the texturing, and pixel processing.

Approach

The approach to complete the project was to recompile a JPEG texture in to the streamable DXT5 texture format while creating an equal sized normal map of that texture through different algorithms an output the normal to a DXT5 texture also. The terrain is generated, then the portions of the texture where high detail are needed are sampled, and low portions are sub sampled ans streamed to their correct positions on the terrain. Finally the lighting is calculated based off of the pre-rendered normal maps.

Technical Details

DXT5 Format – 4X lossy compression algorithm. Compression is based on 4x4 pixel pairs allowing for real time streaming of texture positions.

Sobel Operator – Determines hard edges within images. These hard edges are converted into normals and are used for the normal map.

Mean smoothing – Smooths pixels based on the mean value of their surrounding parents. Used to make smoother normal maps.

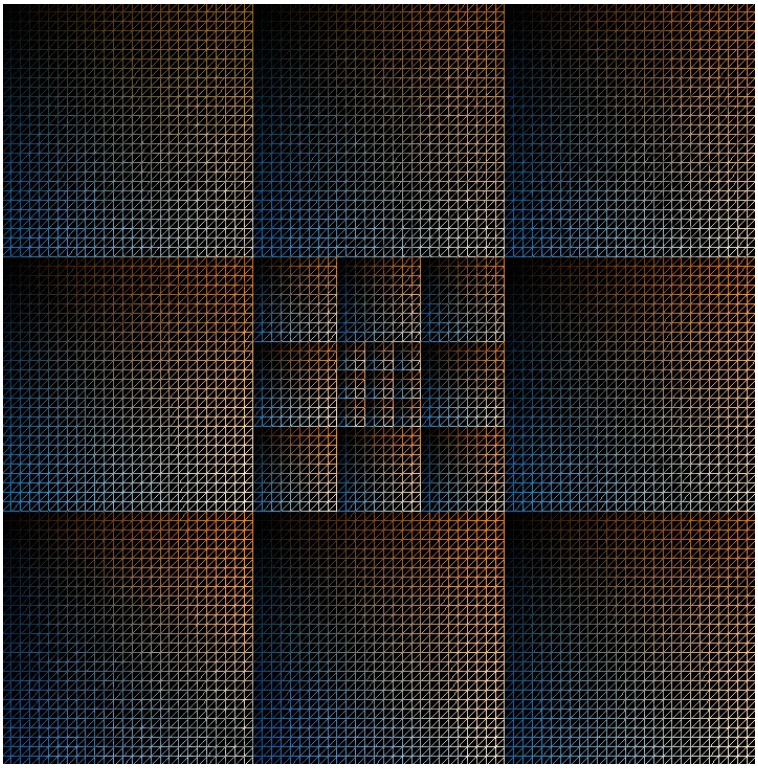
Diamond Square Algorithm – pseudo midpoint based terrain generation algorithm used to create realistic looking terrain.

Tessellated UV Mapping – Mathematic calculations to map tessellated quad to UV positions no matter how tessellated or where on the UV map the quad is.

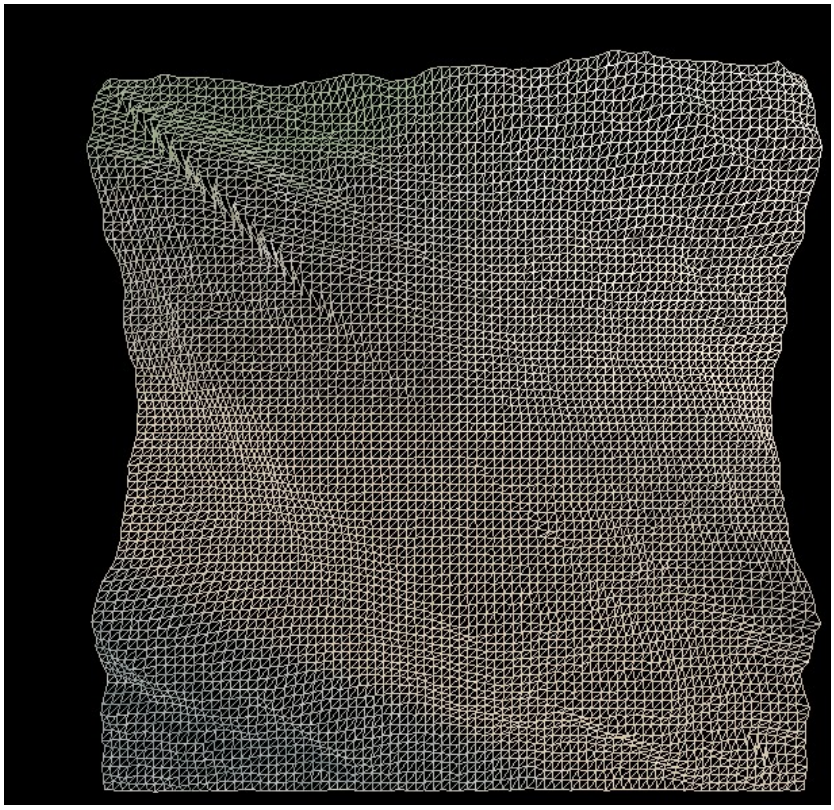
Sub file streaming – Streaming calculations that calculate binary pixel location and only read the required sub portions of a file.

Results

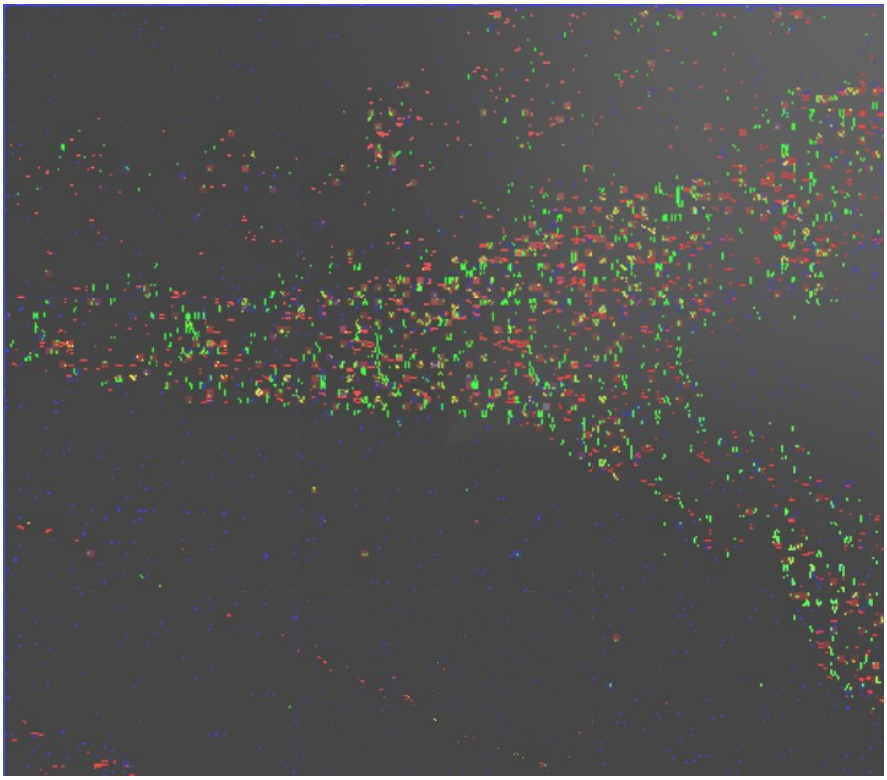
I was successfully able to complete everything from the texture recompilation up to megatexturing the procedurally generated terrain. Below are screenshots of the results.



UV Tessellation

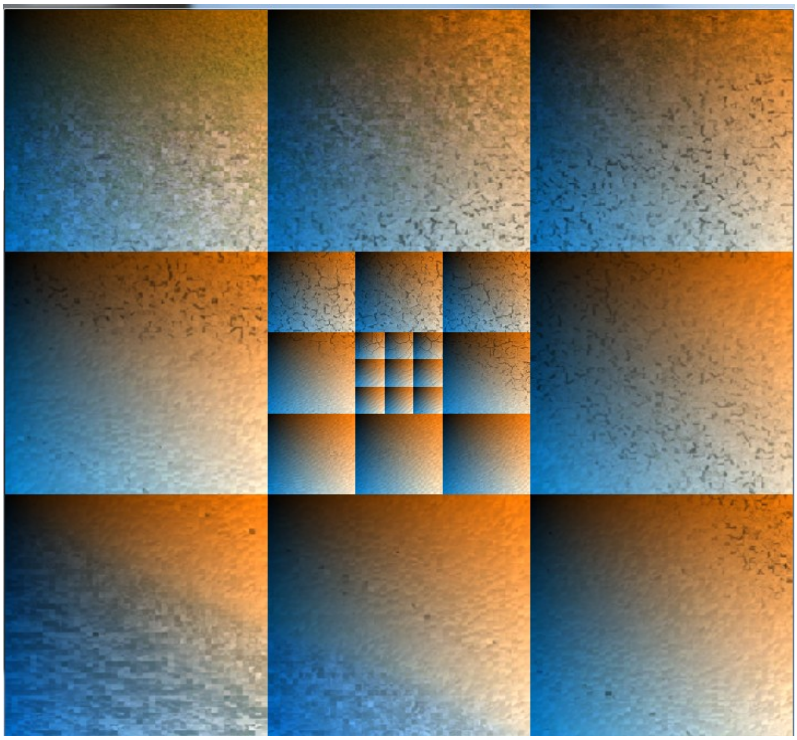


Diamond Square Terrain Generation

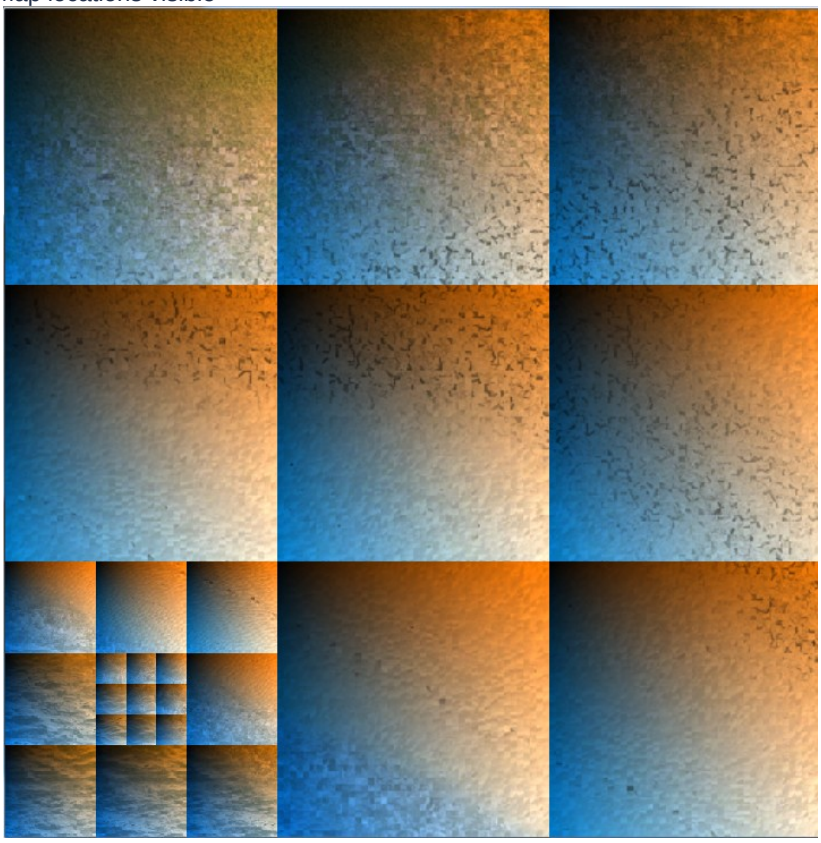


Normal Map Generation and Megatexturing

Normal Map Highspecular effects



Megatexturing with UV Map locations visible



Real time resampling of texture position and data.

Application Controls

WASD – Move around map.

X – zoom in

Z - zoom out

L – Increase specular component

K – Show megatexture sampled normal map

O – Resample previous quad section

I - Resample next quad section

P – Toggle Projection Matrix & Geometry

SHIFT - ~ - Toggle Wireframe

{ 0-1 }* + ` - Reseed the diamond square algorithm. Press ` after the number desired is entered.

Future Enhancements

Since modifying super large textures in paint programs is very memory intensive, I would like to add premerging of multiple texture images. I would also have liked to add a first person view camera that oriented it's height to the terrain it was on. And finally I would like to implement the multiMegaTexture system to allow for continuous high sample rate when moving across vast texture locations. This would be similar to a expanded bounding box where the megatexture objects would overlap eachother, but only the highest detailed one would be visible to the end user

Appendix A

System – Windows 7 Quad core 3.0ghz AMD Phenom processor, 1gb Nvidia graphics card, and 4gb ddr4 ram.

Compilation instructions – Run the code from the release folder or compile from scratch using the VS2010 project files. If ran from folder a glew32.dll needs to be included.