Requirements

Team
Space Blender

Team Members
Andrew Carter
Eric Ghazal
Jason Hedlund
Terence Luther

Due Date:
12/9/13

[This document represents a high level, rough draft of our requirements document that represents the details and functionality of our software system.]

Table of Contents

Introduction

We are developing a software system to automate a process that already exists but is too cumbersome for the average user to complete with the desired results. Our end goal will resemble something along the lines of a small package that any user, that has blender, can download, install, and execute our software.

The system revolves around taking an input, Digital Elevation Model (DEM), and outputting a 3D rendered landscape through Blender. With that the user will be able to do flyovers to make small videos.

We'll accomplish this process by using techniques that have already been proven to work when completing sections of this process. Using the open source Geospatial Data Abstraction Library (GDAL) , we will process the DEMs into Python data formats which can then be rendered in Blender and be used to generate flyover videos of the landscape. We have already tested these processes with third party Python scripts that complete some of these tasks. Our job is to combine all of it and make the package as easy to use as possible.

Some issues that could arise during our development stage is preserving data relationship, defining non-standard data types, and packaging everything in one utility. The data that is received from the DEM file is represented in a binary floating point representation and it's our job to preserve the accuracy and adjust the scale of the image for Blender to render without difficulty. Unfortunately a DEM may have some undefined areas in the map because of the way the DEM is generated or along the collar of the image. It is our job to ensure that these non-data areas are identified and marked so the user doesn't get an incorrect surface as defined by the DEM. We may also have difficulty packaging everything together so Blender would be able to use it. If we are unable to do so then we'll have no choice but to add some other steps to installing and using our software.

Problem and Solution Statement

Problem Statement

The sponsor of our project is Trent Hare from the USGS in Flagstaff, AZ. Trent is the Cartographer/GIS Manager of the USGS Astrogeology Science Center that is responsible for mapping planetary surfaces using DEM image data. Mapping planetary surfaces is an extremely important part of

landing spacecraft on planets. Knowing the exact terrain that spacecraft will be landing on, and having accurate models of a planet's surface, helps ensure that the spacecraft is able to land safely and without damage.

The tools that are needed to process DEM images are available through the GDAL utility package. GDAL can create hillshades and color-relief maps that can overlay on the DEM image to visualize the planetary surface. The program Blender can be used to create 3D models and animations of the planetary surfaces from DEM images, and overlay the hillshade and color-relief-map images that GDAL produces. This process currently has to be done in a series of steps, in two different programs, making the processing of these images a manual task.

The software system we are developing will be a pipeline between GDAL and Blender that will streamline the processing of DEM images. This system will greatly simplify and automate many of the processes that are currently being done manually. Automation of these processes will lead to faster image processing, and possibly, off-loading processing tasks to a central clustered-server. 3D animations and flyover animations of images are currently features of Blender but these features are unintuitive and have many intermediary steps involved to achieve the desired result. The system will simplify the creation of 3D animations and image flyovers that Blender produces.

System Objectives:

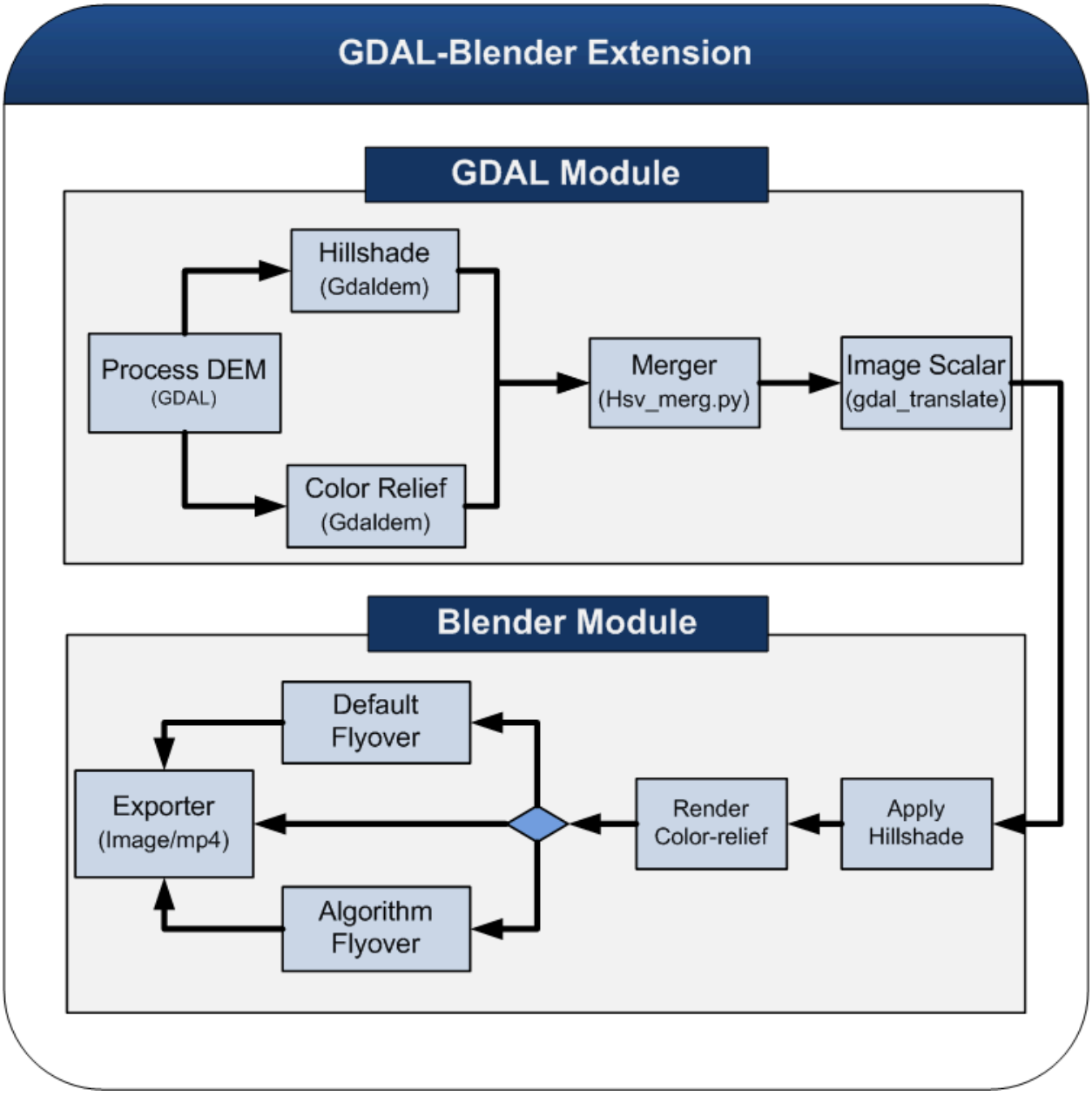
- Integrate GDAL raster library to load DEMs into Blender.
- Help optimize the model into a usable file size and vertical scale.
- Run color hillshades from Blender using existing GDAL binaries. These layers become the material or texture for the DEM surface.
- Method to help define a fly-over path for the 3D model for rendering an animation.
- An integrated method to then export the scene to a WebGL model for Chrome.
- Make this whole method automated or batch-able for many DEMs.

Solution Statement

The system uses Digital Elevation Models (DEMs), which are collected originally from satellites or websites (e.g. USGS). Once this data is processed it will be output as a .blend, .MP4, or X3D file depending on user choices. These files can then be deployed on a website or kept as a viewable file on a computer. A functional system for DEM processing could result in educational viewable and publishable .MP4s. Another step could lead to automated decision making about finding safe landing locations. Streamlining the DEM process will make DEM formatted images more usable and accessible by other institutions, research organizations, and mapping groups.

The whole system strives to automate the DEM computational process and has a possible advantage of being executable on a cluster. There are no foreseeable tradeoffs to the automation of this system. The following describes contains a pipe-and-filter architectural diagram which helps describe how this system will be possible.

USGS Pipe-and-Filter Architecture



Each of the components (filters) and data connections will be discussed in the near future.

Requirements

1. Functional Requirements

1.1. System Requirements (User Application)

1.1.1. The system shall provide a simplified method for processing DEM images.

1.1.2. The system shall be able to process a single DEM image at a time.

1.1.3. The system shall have an automated method to batch pipeline to process DEM images in GDAL and pipe them to Blender.

1.1.4. The system shall provide a simple method to help define an optimized fly-over track for the 3D model for rendering an animation.

1.1.5. Simplified installation kit that installs all necessary software (GDAL, Blender, etc…)

1.1.6. All coding should be done within Python with limited shell execution (if possible).

1.2. GDAL/OGR Script

1.2.1. GDAL shall be able to process .img, .tif, or .dem formatted images (DEMs)

1.2.2. GDAL shall use existing GDAL binaries to create hillshades.

1.2.2.1. Shaded areas (black regions from hillshade) should have specific handling for blender. Should be zeroed out in shading or marked in some special way with Blender.

1.2.3. GDAL shall be able to use existing GDAL binaries to create color-relief maps

1.2.3.1. Color choices shall mostly be colorblind friendly.

1.2.3.2. Color scales shall allow for user choice from a set of color patterns that will be provided by sponsor.

1.2.3.3. Color patterns should include, but are not limited to:

1.2.3.3.1. Grayscale (8-16 bit grays, Lunar)

1.2.3.3.2. Blues & Whites (default)

1.2.3.3.3. Brown & Blue (Earthlike)

1.2.3.3.4. Brown & Red (Mars)

1.2.3.3.5. Rainbow (not colorblind friendly)

1.2.4. GDAL shall be able to scale images to a size that is usable in Blender.

1.2.4.1. If the size exceeds 10,000 or -10,000 then rescale the image by a factor of 10 or an aesthetically appropriate value.

1.2.5. GDAL shall be able to integrate the GDAL raster library to load Digital Elevation Models (DEMs) into Blender.

1.3. Blender

1.3.1. Blender shall be able to render the image processed through GDAL/OGR script.
1.3.2. Blender shall be able apply the hillshade to the DEM image.
1.3.3. Blender shall be able to apply the color relief map to the DEM image.
1.3.4. Blender shall not render 'non-data' areas
1.3.5. Blender shall be able to render a 3D flyover animation of the DEM image.
1.3.5.1. Will have preset flyover animation that exists in a straight line.
1.3.5.2. Height of flyover will be preset to 100ft-1000ft over the highest rendered object in a simple linear direction by default at an attractive skewed angle.
1.3.5.3. (Optional) Pretty paths algorithm, which will create a flyover path that focuses on hotspots or areas input by user or hotspot algorithm.
1.3.6. Blender shall be able to export an .mp4 video file of the 3d flyover animation.
1.3.7. Blender shall be able to export an .x3d 3D model for WebGL-capable browsers

2. Environmental Requirements

2.1. The system shall be cross platform compatible with OSX, Windows, Linux.
2.2. The system shall require Python 3.2 installation on the host computer.
2.3. The system shall require Blender 2.68a installation on host computer.
2.4. The system shall require GDAL package to be installed on the host computer.

3. Non-Functional Requirements

3.1. Compliance
3.1.1. Open source standards.

3.2. Usability
3.2.1. Color schemes should be supported for appeal and colorblind.
3.2.2. Should have either a GUI, Command-line, or Blender Extension interface.
3.2.3. Should be usable by scientific community (USGS).

3.3. Performance
3.3.1. Processing shall not take longer than 24 hours to perform.
3.3.2. Efficiency should not be a concern as it is gated by GDAL and Blender.
3.3.3. Should print out status of process execution to the chosen UI.

3.4. Extensibility
3.4.1. Support additional filters being added to the pipeline.
3.4.2. Must be well-commented, so future coders can add additional functionality in seamlessly.

3.5. Scalability
- 3.5.1. Use single threaded multi-tasking instead of threading.
- 3.5.2. Support future cluster (for Linux due to USGS)

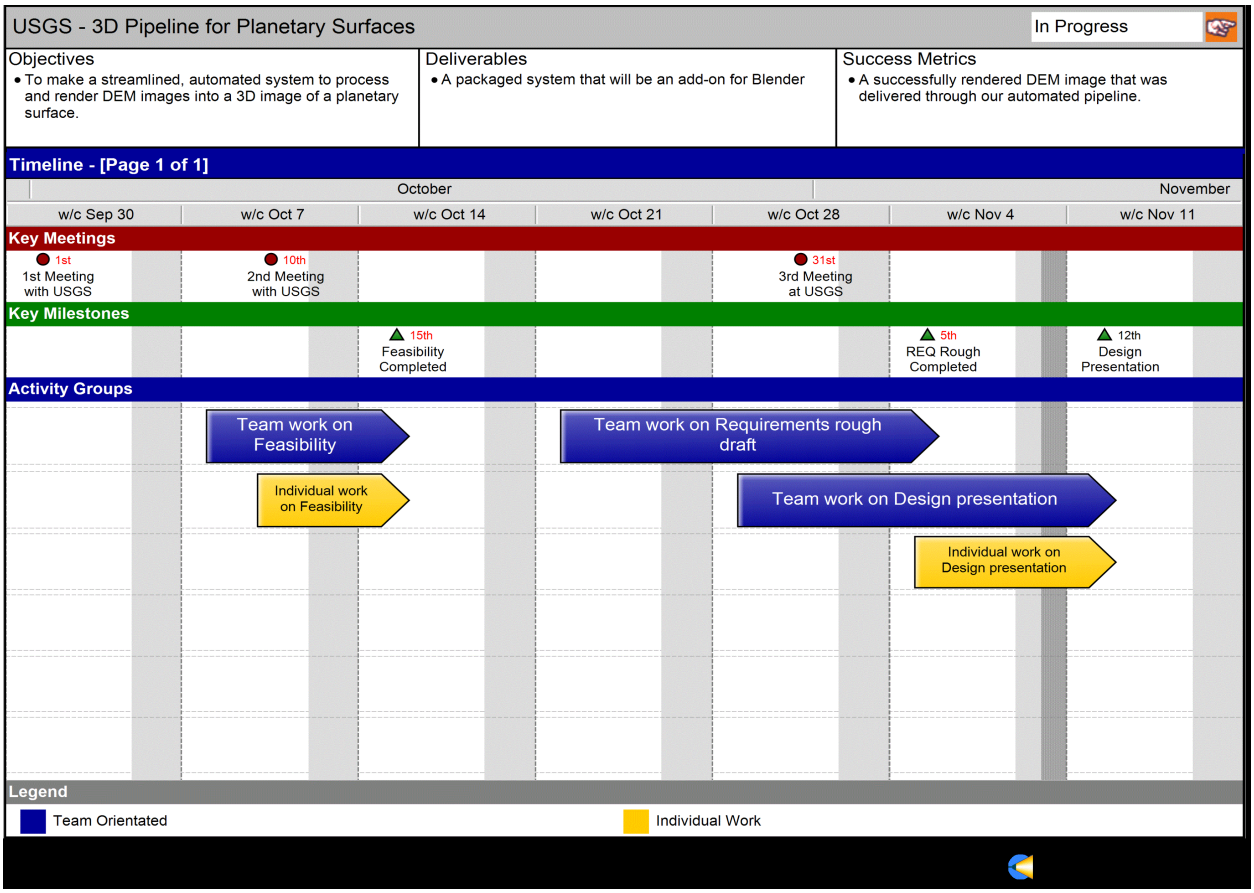
3.6. Portability
- 3.6.1. Operating System Portability
  - 3.6.1.1. Cross platform support (Linux, OSX, Windows)
  - 3.6.1.2. Python powered for additional portability of code.
  - 3.6.1.3. At least Linux support should be provided.
- 3.6.2. Data Interface
  - 3.6.2.1. Input data must conform to what GDAL supports for DEMs
  - 3.6.2.2. Input data scaling must conform to Blender.
  - 3.6.2.3. Data must be able to be piped to other filters in system.

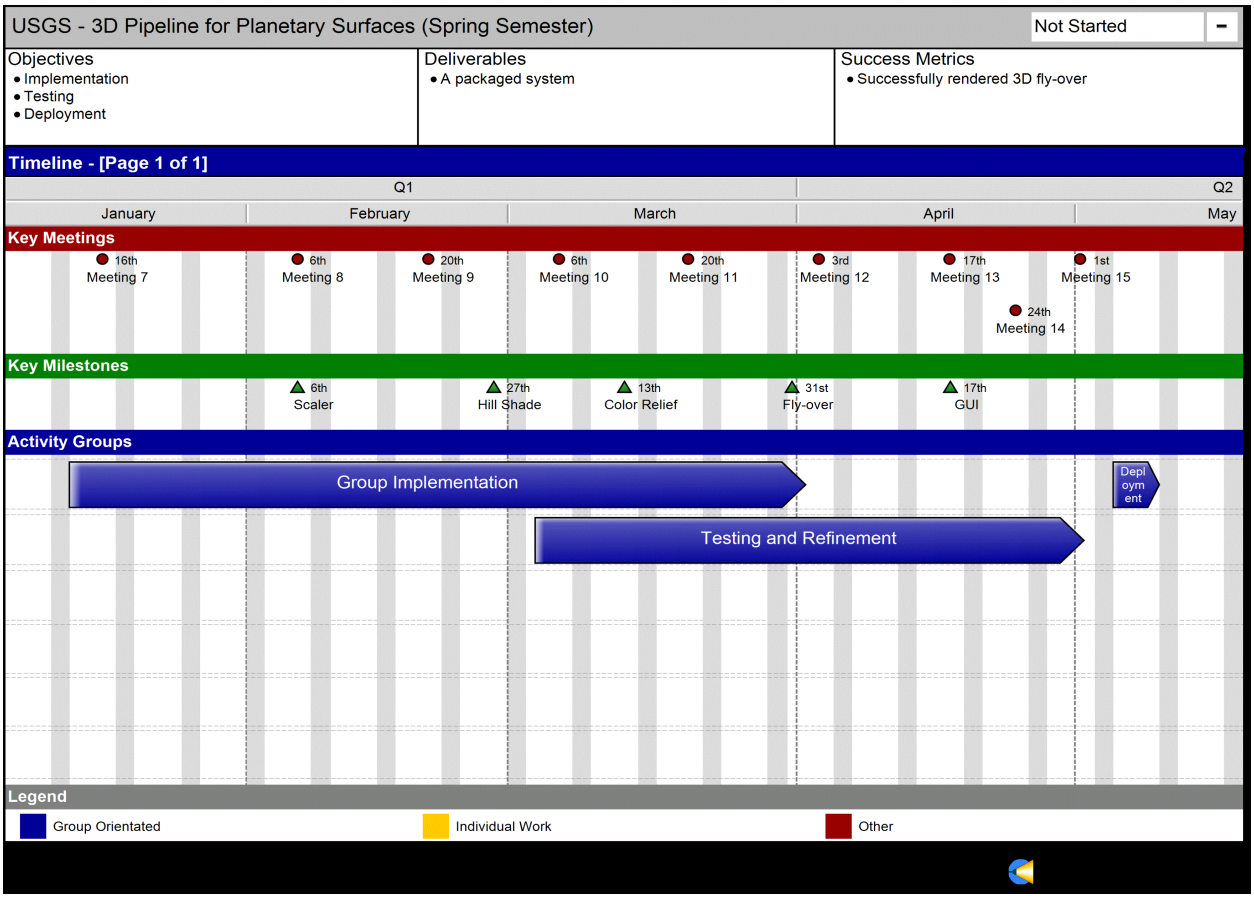
3.7. Documentation
- 3.7.1. All changes to requirements should be documented. This will not have a traceability matrix due to time cost.
- 3.7.2. All documentation will conform to standards laid down in the team foundations document.
- 3.7.3. Meetings with the sponsor will be stored in a drop box folder labeled meeting_minutes_[date].
- 3.7.4. All documents should be stored in the .doc or .docx format.

Potential Risks

Hazard	Primary Cause	Control Strategy	Verification Strategy	Sev	Like	Rating
No data handling	Failure to handle no data regions	Notify user about no data regions	No voids or sudden terrain shifts in Blender	5	3	15
Future: inaccurate landing site	Algorithm for determining landing site is faulty	Verify potential landing zones manually	Compare landing sites with known landing sites	5	3	15
Security vulnerability	Code regions allow for code injection	Perform code analysis	White box penetration testing	5	3	15
Unable to load in Blender	Data scaling is not correct	Failure to load in Blender, run scaler method	Test for scaling consistency	5	1	5
Automated flyover out of bounds	Flyover algorithm does not check bounds	Check scaled boundaries	If camera does not fly outisde terrain	2	2	4
Automated flyover too high	Flyover algorithm does not check highest point	Check for max height	Camera nevers passes through terrain	1	4	4

Project Plan





The progression of this project as displayed in the Gant chart outlines the milestones that will be completed in the spring 2014 semester. Implementation will begin in January and should be complete in late April. Testing and refinement will be an ongoing process that will be started in the beginning of March. The deliverable product produced by this project will be ready to present the beginning of May.

Glossary

Blender – Open-Source Graphics and Animation Application
DEM – Digital Elevation Model
GDAL – Geospatial Data Abstraction Library