CS 4961 Senior Design

Planetary Surface Flyover Movie Generator

Software Design Specification

Document Prepared by:

Shawn Anderson
Fidel Izquierdo Jr.
Angel Jimenez
Khang Lam
Christopher Omlor
Hieu Phan

02 December 2016

CALIFORNIA STATE UNIVERSITY LOS ANGELES

Los Angeles, California
Planetary Surface Flyover Movie Generator

Section 1: Introduction
1.1 Purpose
1.2 Scope
1.3 Intended Audience
1.4 References
1.5 Definitions, Acronyms, and Abbreviations

Section 2: System Overview
2.1 Description
2.2 User Interface
2.3 Hardware Requirements
   2.3.1 User Hardware Requirements
   2.3.2 System Hardware Requirements
2.4 Software Requirements
   2.4.1 User Software Requirements
   2.4.2 System Software Requirements
2.5 User Characteristics

Section 3: Design Considerations
3.1 Operating Environment
3.2 Architectural Design
   3.2.1 Existing Software Solutions
   3.2.2 Programming Languages
3.3 User Interface
   3.3.1 Expected Input
   3.3.2 Output

Section 4: Architectural Strategies
4.1 User Interface
   4.1.1 Viewer
   4.1.2 Add
   4.1.3 Edit
   4.1.4 Delete
   4.1.5 Load in Json
   4.1.6 Preview
4.2 Backend Processing
   4.2.1 Blender - FilePaths
   4.2.2 Blender - Import_OBJ
Planetary Surface Flyover
Movie Generator

4.2.3 Blender - BuildScene
4.2.4 Blender - RenderStills
4.2.5 Blender - AnimateScene

4.2.6 Job Management
4.2.7 Job Management - CZMLParser
4.2.8 Distributed Processing Scripts

4.3 Sun Data Module
4.3.1 spkFiles
4.3.2 center_marsToSun
4.3.4 spherToCart
4.3.4 cartToSpher
4.3.5 marsToSun

Section 5: Other Nonfunctional Requirements
5.1 Performance Requirements
5.2 Safety Requirements
5.3 Security Requirements
5.4 Software Quality Attributes
5.5 Business Rules

Section 6: Human Interface Design
6.1 Overview of User Interface
6.2 Screen Images

Section 7: Appendices
Appendix A: Analysis Models
Planetary Surface Flyover Movie Generator

1. Introduction

1.1 Purpose
This document will provide a detailed outline of the software architecture for the Planetary Surface Flyover Movie Generator (PSFMG). This document will provide readers the details necessary to understand the system and its functionality.

1.2 Scope
This document intends to show how the design will accomplish the functional and nonfunctional requirements described in the Software Requirement Specification (SRS). This document will provide the details required to implement the software system by describing the architecture, subsystems, and interfaces.

1.3 Intended Audience
This documents is intended for the software engineers who will implement the designs described in this document as well as the clients who will deploy the software system.

1.4 References
N/A

1.5 Definitions, Acronyms, and Abbreviations
- PSFMG - Planetary Surface Flyover Movie Generator
- UIM - User Interface Module
- SDM - Sun Data Module
- PVM - Preview Video Module
- GJSBM - Generate Json Start Backend Module
- STMM - Server Task Management Module
- ISSRM - Individual Server Side Rendering Module
- UI - User Interface
- OBJ - Wavefront Object File
- PNG - Portable Network Graphic
- CZML - JSON format with special tags
- Json - JavaScript Object Notation
- 3D - 3 Dimensional
2. System Overview

2.1 Description
The Planetary Surface Flyover Movie Generator provides a simple user interface that allows a user to create a camera path that will be rendered and animated with a video file as output.

2.2 User Interfaces
The users of the software system will be using a website running Cesium hosted by nodejs for the user to enter points to generate camera paths along with other features to control the quality and details of the video output.

2.3 Hardware Requirements

2.3.1 User Hardware Requirements
N/A

2.3.2 System Hardware Requirements
Webhost requires a basic webserver with public accessible address and ports. Hadoop cluster for backend processing of data for movie creation.

2.4 Software Requirements

2.4.1 User Software Requirements
HTML5 compliant web-browser that fully supports WebGL and a graphics card that supports 3D.

2.4.2 System Software Requirements
The software system has several different requirements for the various components of the backend processing.
- Cesium current version for hosting UI
- Nodejs for running webserver to host the Cesium UI
- Blender version 2.78 for creating and rendering the movie
- Python 3.5 for scripting automation of blender and running server for backend
- Spice Framework for calculating the position of the sun relative to position on planet
- Spiceypy Python wrapper for Spice Framework
- Hadoop for distributed processing of rendering tasks and animation
2.5 User Characteristics
The software systems intended users are as follows
- General public - little to no experience, able to use a website
- JPL Engineers - Understanding of CZML and Json file types as well as geopositioning data on Mars.
3. Design Considerations

3.1 Operating Environment
The Planetary Surface Flyover Movie Generator is intended to be run on a web server and server cluster.

3.2 Architectural Design
The Planetary Surface Flyover Movie Generator is split into two distinct parts, a front end UI the user interacts with and a backend server for processing the data. The front end will be hosted by a web server and the back end server will reside on a server cluster for distributed rendering of the video.

3.2.1 Existing Software Solutions
- Cesium - open source web-based globe and map for visualizing dynamic data.
- Blender - open source 3D creation suite.

3.2.2 Programming Languages
- Javascript - Cesium and UI
- Python - Automation Scripting of Blender processes and distributed processing.

3.3 User Interface
The user interfaces with the system via a website that hosts a google earth like 3D model of the planet mars to create a camera path.

3.3.1 Expected Input
The user is presented with a 3D globe of the planet Mars. The user is then expected to click on the planets to place points that are linked together to form a camera path. The user will have the option to control camera angles, altitude, and time between points.

3.3.2 Output
The user will receive a link to download the rendered movie file.
Planetary Surface Flyover
Movie Generator

4. Individual Classes of System

4.1 User Interface

4.1.1 Viewer
Cesium provided viewer that is used inside of HTML 5 canvas tag that creates a Globe of earth as default where the cesium library is built upon.

4.1.2 Add
Create and save point data to Linked List and add a Cesium entity for each point on user double click to an area on the globe.

4.1.3 Edit
Updates the data in the Linked List associated with the point being edited with the new information. Removes all point and line cesium entities and remake each of the point entity in the Linked List and line in between to redraw the points.

4.1.4 Delete
Removes the node in the Linked List that is associated with the point being deleted and remove all point and line Cesium entities. Remake each point entity in the Linked List and line in between to redraw the points.

4.1.5 Load in Json
Takes in a path to a .json file. Loads in the .json file and input data from .json file to Linked List and adds point and line entities to draw out path.

4.1.6 Preview
Has two ways to preview path being run with interpolation between points. First way is first person view where camera is the object and camera view is locked to point data specified. Second way is third person view where camera is a certain distance away from object while always focused on the object. Camera view can be freely controlled by user in this preview mode. The preview mode are toggled.
4.2 Backend Processing

4.2.1 Blender - FilePaths

4.2.1.1 Constructor
Configuration data for program. Contains variables for absolute file path directories where core files are located as well as rendering options for video quality.

4.2.1.2 get_camera_preset
gets the type of camera to simulate on the camera path. Blender has various camera presets that can be used

4.2.1.3 set_blend_file
Sets the name of the blend file used for the job

4.2.1.4 get_blend_file
Gets the name of the blend file used for the job

4.2.1.5 obj_file
Gets the absolute file path for the OBJ file with filename

4.2.1.6 get_temp
Gets the absolute file path for the temp directory

4.2.1.7 get_output
Gets the absolute file path for the output directory

4.2.1.8 get_obj
Gets the absolute file path directory where OBJ file is stored

4.2.1.9 get_relative_path
Gets a relative file path to a specified directory from a specified starting directory.

4.2.2 Blender - Import_OBJ

4.2.2.1 Constructor
Creates local instances of filepaths and output file name

4.2.2.2 clear_blend_file
Deleted default objects from the blend file
Planetary Surface Flyover
Movie Generator

4.2.2.3 import_obj_file
Imports OBJ file from disk into a newly created blend file

4.2.2.4 select_objects
Selects all objects in scene and sets the selected attribute to false so no objects are
selected on the scene. This is important for later tasks that are dependant on selected
objects

4.2.2.5 set_textured_view
Sets view so that textures are enabled, which allows for rendering with textures

4.2.2.6 save_scene
Saves the blend file to disk with the specified file name.

4.2.3 Blender - BuildScene

4.2.3.1 Constructor
Calculates the number of frames and sets the end_frame property as well as linearly
creates all the points between each user selected point.

4.2.3.2 inter
Linearly create all points between two vertices specified by user.

4.2.3.3 set_end_frame
Sets the last frame property in blender

4.2.3.4 create_lamp
Creates a Sun Lamp object and sets its location and intensity

4.2.3.5 create_camera
Creates a camera object and sets position and camera angles specified by user

4.2.3.6 create_camera_path
Creates a camera path using the points created in 4.2.3.2

4.2.3.7 create_key_frame
Creates keyframes used in transitions of camera angles at vertices

4.2.3.8 bind_camera_path
Binds the camera to the camera path so that the camera will follow the path during
animation
4.2.3.9 new_reducer
Reduces the polygon count in the mesh

4.2.3.10 set_render_options
Using options set in 4.2.1.1, sets all of the configurable options in blender

4.2.4 Blender - RenderStills

4.2.4.1 Constructor
Creates local File Path object

4.2.4.2 get_frame_count
Gets total number of frames in the animation

4.2.4.3 render_stills
Walks the camera through the camera path rendering a still PNG file for each view of the camera at each position

4.2.5 Blender - AnimateScene

4.2.5.1 Constructor
Reads list of still PNG files from directory and sets local instances of location directories from FilePath object.

4.2.5.2 animate
Sets up the image sequencer and reads all the stills from the directory and sorts them in numerical order. Animates all the still images by appending them to each other in numerical order into an image sequence.

4.2.6 Job Management

4.2.6.1 timing
Profiling function for timing of functions

4.2.6.2 do_import
Performs import operations to create the blend file and import the OBJ file from 4.2.2

4.2.6.3 do_create_scene
Performs operations to create the scene from 4.2.3

4.2.6.4 do_render
Performs operations to render still images from 4.2.4
**Planetary Surface Flyover Movie Generator**

4.2.6.5 *do_animate*
Performs operations to animate still images from 4.2.5

4.2.6.6 *main*
Main function controls setup of environment, Parses file send by users selections, and the order of operations in this module 4.2.6.

4.2.7 *Job Management - CZMLParser*

4.2.7.1 *Constructor*
Creates local instance of File Paths object

4.2.7.2 *openFile*
opens and reads the CZML file from disk

4.2.7.3 *movie_Properties*
reads the movie properties from the CZML file into a python dictionary

4.2.7.4 *camera_Position*
reads the camera position from the CZML file into a python dictionary

4.2.7.5 *sun_Position*
reads the sun position from the CZML file into a python dictionary

4.2.7.6 *camera_Orientation*
reads the camera orientation from the CZML file into a python dictionary

4.2.7.7 *sun_Orientation*
reads the sun orientation from the CZML file into a python dictionary

4.2.7.8 *cesium_Version*
reads the cesium version number from the CZML file

4.2.7.9 *my_path*
reads the path information from the CZML file into a list of tuples

4.2.7.10 *blenderCamera*
reads camera path information into a simple format for easy importing into blender automation scripts in 4.2.6.6
4.2.8 Distributed Processing Scripts
Takes the finished rendered movie and uploads it through file transfer protocol to a server and provides a direct link to the user.

4.2.8.1 Mapper - Future Implementation

4.2.8.2 Reducer - Future Implementation

4.3 Sun Data Module

4.3.1 spkFiles
Loads the kernel that are needed for determining the calculations.

4.3.2 center_marsToSun
Finds the position of the sun from the center of mars.

4.3.3 spherToCart
Changes spherical(long,lat,alt) to cartesian(x,y,z) in km

4.3.4 cartToSpher
Changes cartesian(x,y,z) in km to spherical(long,lat,alt)

4.3.5 marsToSun
Finds the direction of the sun in respect to a point on mars
5. Other Nonfunctional Requirements

5.1 Performance Requirements
The software was designed to be able to run at an acceptable timeframe on any hardware however the better the graphics card and the processor on the machine running the software the quicker the render times will be. The software is also designed to run on parallel machines so having a server cluster will help render videos in acceptable timeframes.

5.2 Safety Requirements
There is no specific safety requirements since this software will not used in a capacity to control hardware.

5.3 Security Requirements
The safety requirements is to be handled in house by JPL. The software will be transferring data insecurely over unencrypted channels. Any and all additional security including protection of data and backups are outside the scope of the software being developed.

5.4 Software Quality Attributes
The software is designed to operate at a small scale so issues with scalability can arise with high volume. The render quality and times are designed to be at a high quality even when zoomed in as well as render frames at an acceptable time even on minimal hardware.

5.5 Business Rules
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
Planetary Surface Flyover Movie Generator

FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
6. Human Interface Design

6.1 Overview of User Interface
A simple user interface that is designed for the technical and non-technical user. It allows the user to create and modify a camera path that can be rendered into a movie via the back-end processing.

6.2 Screen Images

This is the page that will appear when the user first get to the page.
Planetary Surface Flyover
Movie Generator

User can load in a prepared JSON or CZML file.

User can edit a point.
Planetary Surface Flyover Movie Generator

The point and lines will change accordingly.

User can select a point for deletion.
Planetary Surface Flyover Movie Generator

User can preview what their chosen path looks like.

Preview has 2 modes. First person strictly follows the point data. Third person allows user to move camera around to see terrain to edit their path.
7. Appendices

Appendix A: Analysis Models

Level 0 DFD
Planetary Surface Flyover Movie Generator

Level 1 DFD
Appendix B: To Be Determined List
N/A