Software Requirements Specification
Version 1.0
Synthetic Data Pipeline for Pose Estimation

By:
William Stern
Nathan Pichette
Stephane Baruch
Hanibal Alazar

Table of Contents

1. Introduction
    1.1. Objective
    1.2. Purpose
    1.3. Scope
    1.4. Stakeholders
    1.5. Definitions

2. Product Description
    2.1. Product perspective
    2.2. Product functions
    2.3. User Characteristics
    2.4. Constraints
    2.5. Assumptions and dependencies
3. Sample Outputs

1. Introduction

1.1 Objective

The objective of our project is to create an api that will allow users to create synthetic data for pose estimation. The data must be a rendered clip that contains a satellite in motion. The clip will also contain a selected background and specific lighting conditions. An API will be used to adjust the video to the user's needs. The user will be able to set the location and strength of a light source, the model of the satellite, as well as the path of motion and rotation.

1.2 Purpose

The purpose of our project is to create data to train a machine learning algorithm. The algorithm is responsible for understanding the position and rotation of a satellite in order to land a small satellite on to a larger one. There are two current methods for creating the satellite data. The first method is using physical models and cameras to create videos. The problem with the first method is that it is hard to produce clips with very high and very low light conditions. The second method is to animate videos containing the satellite. The problem with this method is that in order to make multiple clips it is necessary to reanimate each video, this takes time. Our project will expedite the second method of data creation so that videos can be made to the users specifications quickly and without the need for animation experience.

1.3 Scope

The scope of this project is fairly small. Currently the only people who would use our project are Dr.white and his assistants. All users will be knowledgeable, be capable with technology and have an understanding of the use of our tool. This allows for our focus to be more centered around the creation of videos. We will still have an API however it can be fairly rudimentary because we don’t have to sell the product and the user will know what needs to be done.

1.4 Overview

This document will go through our project to describe all requirements that need to be fulfilled in order to have a complete product. The first section will describe the product as it is going to be given to the user. The second section will describe the features that will be included.

1.5 Definitions

API - Application Programming Interface

Blender - Open Source 3D Computer Graphics Tool
Python - Modern programming language built with C
mp4 - video file format

2. Product Description

2.1 User interface

The API for this product will be a bare bones interface that is built with python. The user will select from a variety of options to customize what they want to see in the video. Once completed they click the finished button which will begin rendering the click. Lastly the program will return to the user a mp4.

2.2 Product Features

2.2.1 Satellite Models

In the API the first option for the user will be a drop down menu where they are able to select which satellite they want to see in the video clip. Each satellite will have its own blender model to be incorporated. *May allow upload of personal satellite models if applicable.*

2.2.2 Satellite Path

The next option on the API will be x, y, and z fields that will be used to mark the start of the satellites path. This will be followed by another field where the user can input *an equation* that a center point of the satellite will follow. *May allow for choosing location of center movement point*

2.2.3 Rotation

The user will be able to input the amount of rotation on the x, y, and z axes. This rotation will accompany movement made along the user defined path. *May also be able to choose a point of rotation on the model.*

2.2.4 Lighting

The user will be able to set the lighting used for the rendered clip. *There may be options to add multiple light sources.* Each light source will be positioned based on x, y, and z coordinates that are defined by the user. The light sources will have adjustable brightness allowing for both low and high light scenarios.

2.2.5 Background

The user will have the ability to decide the background of the clip based dropdown list of background options. *May be able to upload a background*

2.3 User Characteristics

The intended users of this program are members of Dr. White's NETS lab. The user must be able to understand how the configuration file works and what types of inputs it takes. We are assuming that the user has an above average understanding of how to use a computer and a basic understanding of computer programming

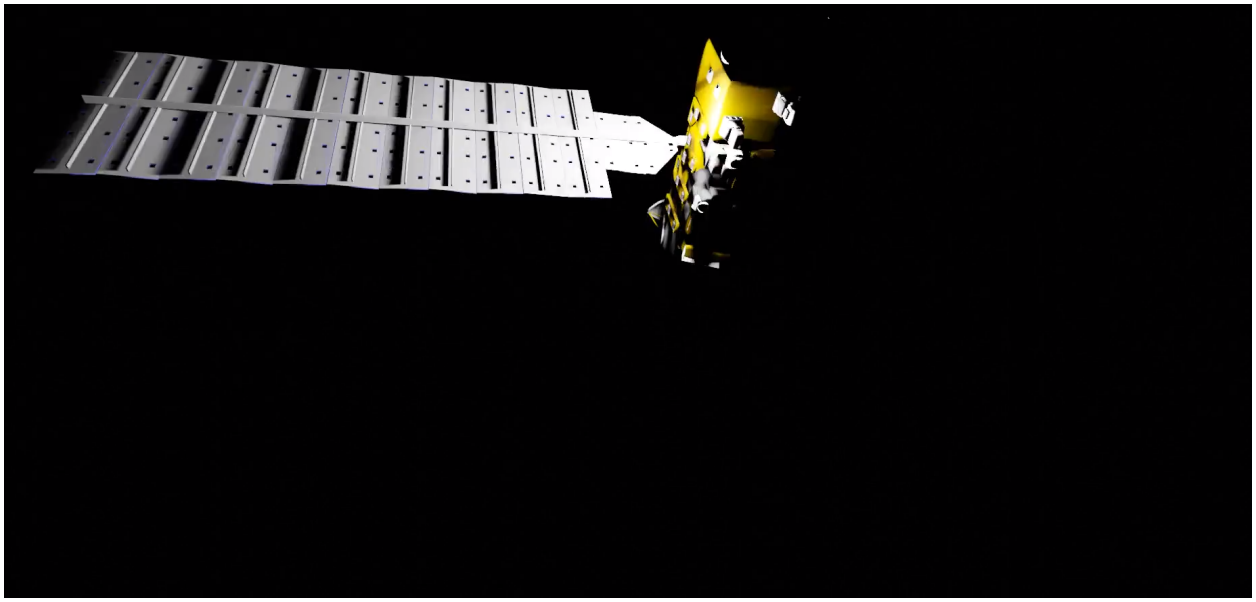
2.4 Constraints

The program must be able to run in a reasonable amount of time. The program must be able to run on a variety of computers with different operating systems and different graphic card configurations

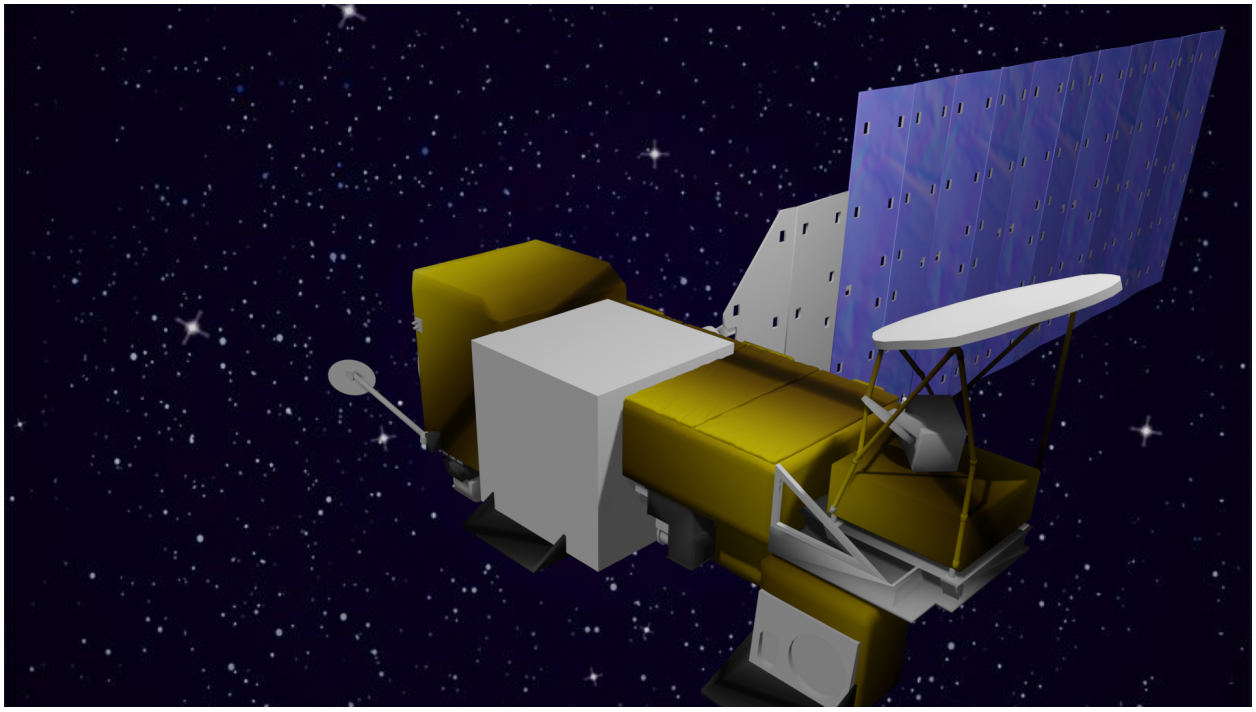
2.5 Assumptions and dependencies

The programs can run on Windows 10-11, and Mac OS, and Linux. Parts of the program that utilize Nvidia Omniverse will only run on Windows and Linux, and require a RTX 2080 graphics card or better.

3. Sample Outputs



Above is a screenshot of an automated blender render created using the blender API. This is a still taken from a short video that shows the satellite rotating and moving across the screen. Future iterations will have backgrounds and multiple sources of lighting.



Above is a sample blender render of a satellite. This is what the output will look like except it will be in the form of a short video. Everything in the photo will be customizable from the satellite model and background, to the lighting and camera angle.

```
[satellite]
satellite_file = "\nasa-aqua-satellite-obj\nasa-aqua-satellite.obj"

[background]
background_file = "space.jpg"

[lighting]
color = "yellow"
brightness = 90

[motion]
transform = [(0,0), (10,0)]
rotate = "full"
rotate_angle = 0

rotate_speed = 7
transform_speed = 3

[video]
length = 10
framerate = 30
dimensions = (100, 100)
```

Above is an example of what the configuration file might look like.