**Space Rock**

*Graphical User Interface Design*

*Version 2.0*

Team #04

14 February 2017

***CS 460 Software Engineering***

[**SpaceRock GUI Documentation**](#_70czbg2jmkxe) **2**

[Introduction](#_6uqg3dsabiis) 2

[Screen Design](#_bk7rifo9n9w6) 2

[Functional Design](#_n9351idsgbae) 3

[Software Design](#_lcwwdqyxc68y) 5

[Networking](#_3pnsusxvqtpz) 5

[Processing](#_1j2l1iyjcu0v) 5

[GUI](#_xcl4mx5j0vd1) 6

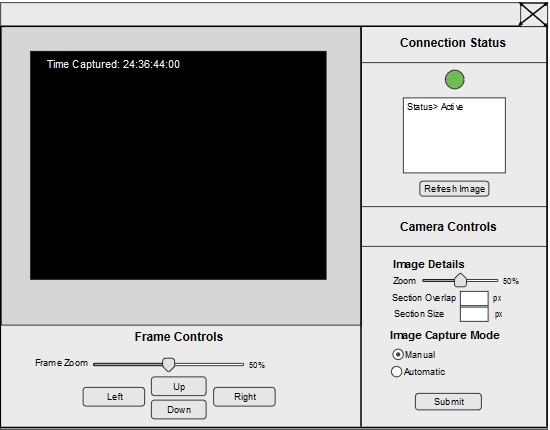
[SpaceCraft Emulator](#_dt4q0nkp6f7b) 6

# SpaceRock GUI Documentation

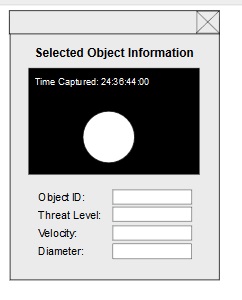
## Introduction

The SpaceRock system requires a Graphical User Interface (GUI) in order to allow ground-based users to control and evaluate the system’s performance. This document will describe the layout, the design, and the internal workings of the SpaceRock GUI simulator.

## Screen Design



*Figure 1: Mock-Up of SpaceRock GUI*



*Figure 2: Mock-Up of Selected Object Information Pop Up*

## Functional Design

|  |  |
| --- | --- |
| **UI Element** | **Function** |
| Object Display Canvas | Displays a grayscale image derived from the raw data sent by the satellite. This represents the last image captured by the camera and received by the operator interface. The grayscale image, its detected objects, and the zoom initially shown will be determined by the last camera settings at the time the submit button was pushed.  The image also contains a timestamp which details the time the image was taken.  Objects can be selected or hovered over and a pop up dialogue will appear to showcase their details. These details can be viewed from the Selected Object Information Panel. |
| Selected Object Information Panel | Displays the current object’s identification number, velocity, threat level, and size in text format while also providing an image of the object that has been selected and the timestamp detailing when the image was captured. |
| Satellite Camera Zoom | Adjusts the camera’s zoom toward the center of its field of view. This does not affect the frame that is currently displayed in the Object Information Panel. |
| Section Size | Adjusts the image processing section size. This does not update the current frame and changes will not be visible until the next image is received. |
| Section Overlap | Adjusts the amount of overlap each section is allowed to have with its neighbors. |
| Manual/Automatic Mode | Checking Manual Mode and clicking submit sends a signal to the satellite that instructs the camera to take a single image and return the data.  Checking Auto Mode and clicking submit sends a signal to the satellite which instructs it to continuously capture photographs and return the data. |
| Submit Button | The Submit button sends our currently selected settings to the satellite so that they will be utilized in the next frame capture. |
| Refresh Button | The Refresh Button allows us to manually capture a new image using the last settings in the event an image processing error is evident in the Object Display Canvas, such as a white screen or a noisy frame, and was not caught by the system’s automatic detection. |
| Status Indicator Light | The status indicator light resides near the top right of our GUI panel. It changes color depending on the connection status. If the operator interface has been in recent communication with the satellite or is currently receiving data, the indicator is green. If the Operator Interface has sent a signal and is waiting to receive something for longer than our set amount of time, the indicator light switches to yellow. If the connection has timed out entirely, the indicator light changes to red. The information provided by the status indicator light is supplemented by the text output that displays in the Status Output Console. |
| Status Output Console | The status console dynamically updates with messages to inform the user of the system’s connection status with the satellite. If the connection is lost(no message received after a given amount of time), a message indicating this will display. When the connection is active, this console displays the text “Connection is Active.” |
| Canvas Frame Zoom | The image object canvas frame slider allows you to zoom in on the last captured frame in the Object Display Canvas. This capability is also provided in the mouse controls |
| Up, Down, Left and Right Buttons | The Up, Down, Left, and Right buttons allow you to view different sections of the enlarged frame in the Object Display Canvas. This capability is also provided with a left mouse click and drag. |

|  |  |
| --- | --- |
| **Mouse Controls** | **Function** |
| Mouse Middle Button | Scrolling with the mouse middle button zooms in or out on the image in the Object Display Canvas. This zoom method only allows us to modify the last frame received by the ground control station. The camera zoom on the satellite is unaffected by this action. |
| Mouse Left Click | Mouse left click and drag lets you reposition the resized frame image in the Object Display Canvas. |
| Hovering Over Detected Objects | Hovering over detected objects will show a tooltip containing information about them such as their size and velocity |

## 

## 

## Software Design

### Networking

The SpaceRock GUI communicates with the SpaceCraft Emulator via a Transmission Control Protocol (TCP) stream containing serialized Java objects. TCP was chosen for ease-of-development, and experimentation may reveal that the extra overhead it introduces is unacceptable. In that case, switching to User Datagram Protocol (UDP) should not be difficult.

Several message types can be sent through the TCP stream. At present, this includes image segments requested by the operator (and the corresponding operator request), recognized object data (location and size) from a single image capture, and camera parameter updates.

The connection with the satellite is a two way stream. To keep the UI responsive while potentially receiving data from the satellite, any incoming data is handled in the background and the notifies the UI of updates.

### Processing

Once an incoming data stream has been received from the satellite, the list of newly identified objects (with screen positions and sizes as determined by the camera) are processed using a separate processing module.

This module will track object information from the three most recent images received by the satellite camera. The module will assess the similarities and differences between the objects of the most recent and previous images, and will assign labels and estimate the trajectory relative to the camera of those objects which are deemed identical across each image.

### GUI

The GUI is implemented using JavaFX. JavaFX was chosen as it is the modern Java UI framework, and presents very large amounts of flexibility in GUI layout and behavior. Additionally, JavaFX is hardware accelerated, ensuring that even large displays of space objects are not slowed down by the rendering system.

### SpaceCraft Emulator

The SpaceCraft Emulator acts as a model and simulation of how the real Spacecraft system would act and communicate with the Ground Station GUI and how it would transfer data between the Camera and the Ground Station. This emulator takes the place of the true SpaceCraft system that will be developed in the future.

The SpaceCraft Emulator sends a list of identified objects (position, size, etc.) to the Debris Processor, and optionally a list of images used for the raw image display mode. The Debris Processor can send to the SpaceCraft Emulator, camera parameters such as desired zoom level and also system parameters such as desired sector overlap. The SpaceCraft Emulator will then send necessary data to the Camera as well as receive data from the Camera and do some processing.