Design Document for Gigapixel Images

Revision 2.2

Authors:

Douglas Peterson

Thomas O'Brien

Daniel Garcia

Matthew Ostovarpour

Table of Contents

1 INTRODUCTION

1.1 INTRODUCTION AND BACKGROUND

This design document defines a technical solution for development of the stochastic image reading project. This project will result in two products.

1. A new stochastic sampling feature inside of the open source Geospatial Data Abstraction Library (GDAL).
2. A technical proof of concept image viewer that utilizes the GDAL sampling feature. The viewer shall be capable of performing expected image viewing tasks such as panning and zooming. As part of the proof of concept, we expect that the viewer will run on a typical

The products to be delivered will build upon earlier work performed in 2015. Upon completion, these products will be delivered to the United States Geological Survey (USGS) to positively address three key problem areas: slow read times; panning/zooming of non-processed images, and pyramids. It will be used by USGS analysts to view very large images without the need for any kind of image preprocessing. The products will reduce current sampling times by improving reading times and support additional image formats. A new image viewer will be more functional, be supported on additional platforms, and be easier to use.

Both products will run on all three major operating systems (Windows, Mac, Linux). The development languages used to build these products are C and C++.

1.2 PURPOSE OF THE DOCUMENT

The purpose of this document is to provide a detailed description of the stochastic image sampling algorithm as well as a detailed description of the architecture and technologies used in the image viewer.

1.3 SCOPE OF THE DOCUMENT

The scope of this project is to deliver two products:

1. A new sampling feature inside of the Geospatial Data Abstraction Library (GDAL).
2. A proof of concept image viewer that utilizes the new GDAL sampling feature.

We will be implementing a stochastic sampling algorithm into GDAL to provide another way of processing images. This algorithm will recognize which sections of the larger image will need to be sampled to become a single pixel in the output. Secondly, it will randomly sample a number of pixels for those sections based on either a user supplied value, or a default value. Finally, the result will be the desired image data returned by GDAL.

The “lightweight” image viewer will be a proof concept. It will provide the necessary features and be lower on cost and system resources. The image viewer will have the ability to zoom in on images, pan across the image, and open new images. It will utilize the stochastic algorithm that was implemented inside GDAL. It will also cache certain parts of the data so that it may run efficiently with as few calls to the GDAL library as possible.

Ultimately, this project should improve productivity of USGS analysts by speeding up the time it takes for analysts to interact with their large image data. Using this stochastic sampling approach, analysts will be able to view a quality image in much less time than it takes to run other preprocessing algorithms.

1.4 Glossary

Term	Definition
USGS	United States Geological Survey
GDAL	Geospatial Data Abstraction Library. An open source library used by many companies, including the United States Geological Survey, to read geospatial image data.
API	Application Program Interface. APIs are a set of routines, protocols, and tools for building software applications. The API specifies how software components should interact.
GL SL	Open GL Shading Language. GL SL is a high-level shading language based on the syntax of the C programming language. It is the principle shading language for OpenGL.
Downsampling	Downsampling is the use of algorithms and techniques to create images that are miniaturized duplicates of the master image.

Table 1: Glossary

1.5 Intended Audience

The intended readers of this document include:

- Developers and maintainers of the software
- Northern Arizona Faculty Mentor
- USGS Sponsor

2 System Overview

This section provides an overview of the components and architecture that make up the system, including the third-party libraries used by each component. The architecture overview contains a high level system diagram with a detailed description of each individual component, including a component diagram, the libraries that they use, and how they each fit into the whole application.

2.1 Third-Party Libraries

This application uses several third party libraries to perform more complex tasks such as cross platform window handling, OpenGL extension handling, etc. in order to make development move along quicker.

The libraries used are listed below:

Item	Syntax
GDAL (**G**eospatial **D**ata **A**bstraction **L**ibrary)	Handles efficiently reading multiple file types, gathering image metadata, image pixel data, and data type conversions so this application may remain agnostic of the underlying drivers used to load images into memory.
OpenGL	Cross Platform rendering API is used to create graphics applications that can run on multiple platforms. Requires other libraries to help it function.
GL SL	OpenGL Shading Language is used to write fragment shaders and pixel shaders for rendering polygons and textures to a window
GLFW3 (Open**GL** **F**rame**W**ork v3.x)	GLFW3 handles creating and destroying windows, responding to window events, and responding to user input across multiple platforms.
GLEW (Open**GL** **E**xtension **W**rangler)	GLEW dynamically loads OpenGL functions at runtime to support the latest version of OpenGL

Table 2: Third Party Libraries

2.2 Architectural Overview

This application is made up of three major components:

1. The window handler responds to window events, user input, and other types of interaction.
2. The reader performs image file reading and downsampling.
3. The renderer takes the downsampled image data and shows it on the screen.

See figure 1 for a visual representation.

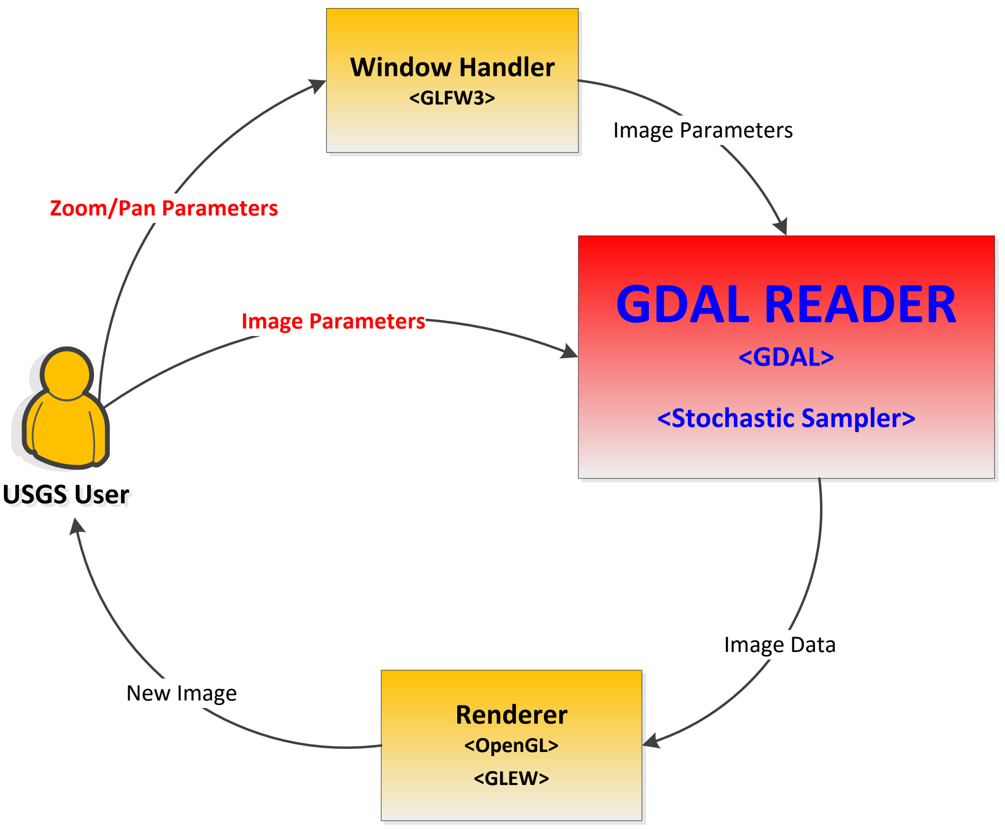

Figure 1: System Context Diagram

Each component in this diagram plays a vital role in the application.

- GDALReader utilizes GDAL to load pixel data via the supplied image parameters, then runs our stochastic sampling function to quickly output the image at a smaller size
- The Renderer then receives the image data from GDALReader and uses OpenGL to render the data to the screen.
- Window Handler will respond to user input and update parameters accordingly, then the image will be resampled by GDALReader and passed back to the renderer.

2.3 Component Descriptions

Each component has its duty for the software to run smoothly. The GDAL Reader, Renderer, and Window Handler communicate with each other by passing pointers to relevant data that must be updated. For example, a buffer with image data will be passed to GDALReader. GDALReader will store the image pixel data inside of it and then pass it to the renderer. The renderer will show the image on the user's screen. The buffer will be passed back to GDALReader when it needs to be updated.

2.3.1 GDALReader

GDALReader is a very key component of the system. This component utilizes the GDAL library to read image files that contain geospatial data.

When image parameters are passed to this component module, it will use GDAL to open and read the image data within the file. While reading the file, it will also be running the custom developed stochastic sampling algorithm to create an in-memory image that is the size of the window that is open. Once the sampling is complete, the in-memory image is passed to the renderer to show it to the user.

GDAL is used as it allows for the reading of images in many file types. This component also provides functionality to allow it to be used in any program. It provides data structures to store image metadata and to initialize an image, free the memory allocated for the image, and to sample the image to any size.

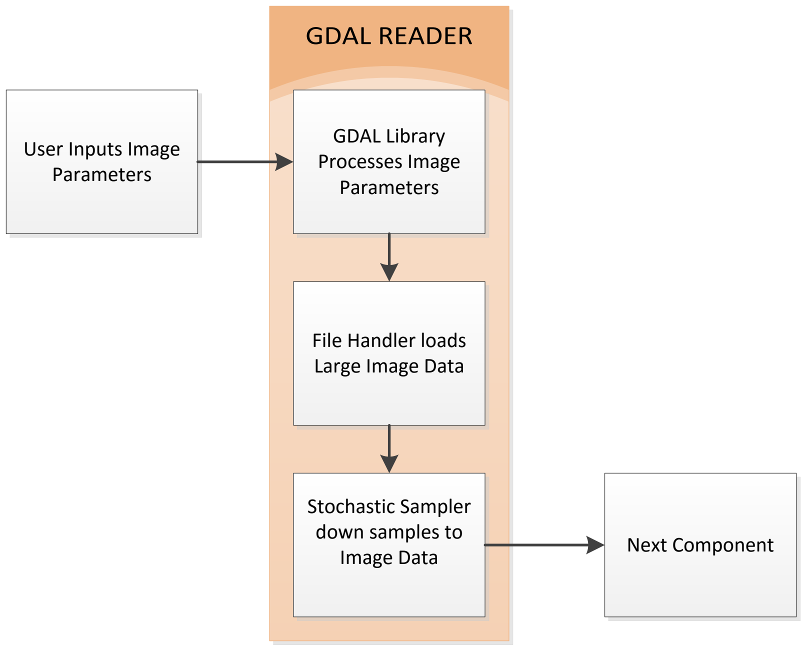
GDAL READER
User Inputs Image Parameters
GDAL Library Processes Image Parameters
File Handler loads Large Image Data
Stochastic Sampler down samples to Image Data
Next Component

Figure 2: GDALReader Component Diagram

The GDALReader component accepts ImageParameters including the filepath and the desired output size. It then uses GDAL in the subcomponent File Handler to load the LargeImageData. This data is then passed to the Stochastic Sampler component where it is downsampled to ImageData and passed on to the next component.

2.3.2 Window Handler

The Window Handler runs off of GLFW3. It catches all events sent to the window. All user input passes through this component. It may respond to clicks, drags, keypresses, window resizing, window closing, etc. Since windows are created with GLFW3, this component is built on top of GLFW3's API in order to respond to window events.

This component allows the user to fully interact with the image in the natural way that they should expect. There will be options for zooming in on a part of the image, panning across the zoomed image, and opening a new file.

"Lightweight" viewers will be investigated as part of the project proof of concept. The viewer is expected to simplify the user's experience when accessing the GDALReader component. It is not expected to be a full-featured image processor.

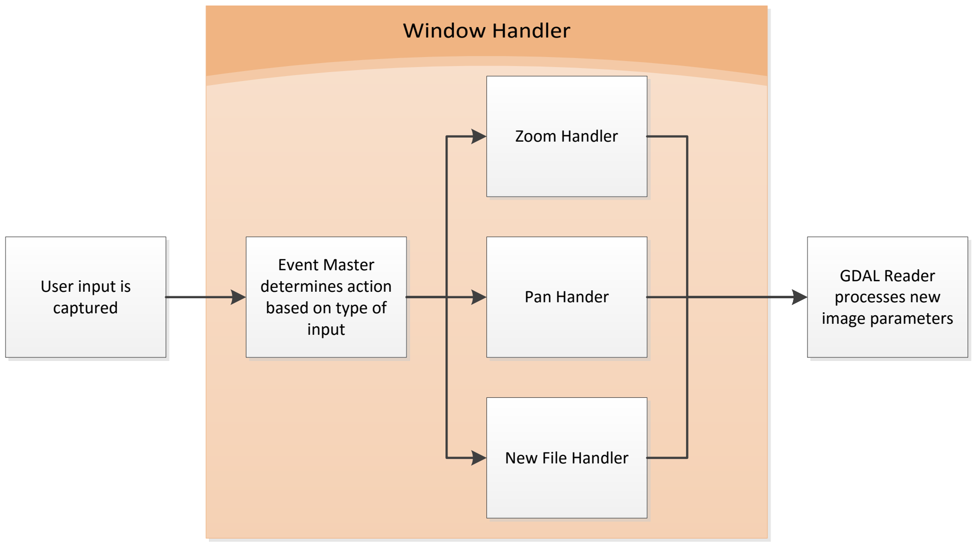
Window Handler
User input is captured
Event Master determines action based on type of input
Zoom Handler
Pan Hander
New File Handler
GDAL Reader processes new image parameters

Figure 3: Window Handler Component Diagram

The Window Handler component depends on GLFW3 to supply it with InputEvents. WindowHandler determines which action to take based on the InputEvent. When responding to the event, it will generate new ImageParameters to pass into GDALReader (see Figure 2).

2.3.3 Renderer

The image renderer component displays data to the user. It is built on top of OpenGL using the GLEW and GLFW3 third party libraries. GLEW allows the application to use the latest OpenGL API functions and GLFW3 handles windowing across Windows, Mac, and Linux. Note that the API functions need to be supported by the user's graphics card.

This component will receive image data that has been processed by the GDALReader and through calls to the OpenGL API and GLSL shaders. This will result in the image being rendered to the screen as a texture.

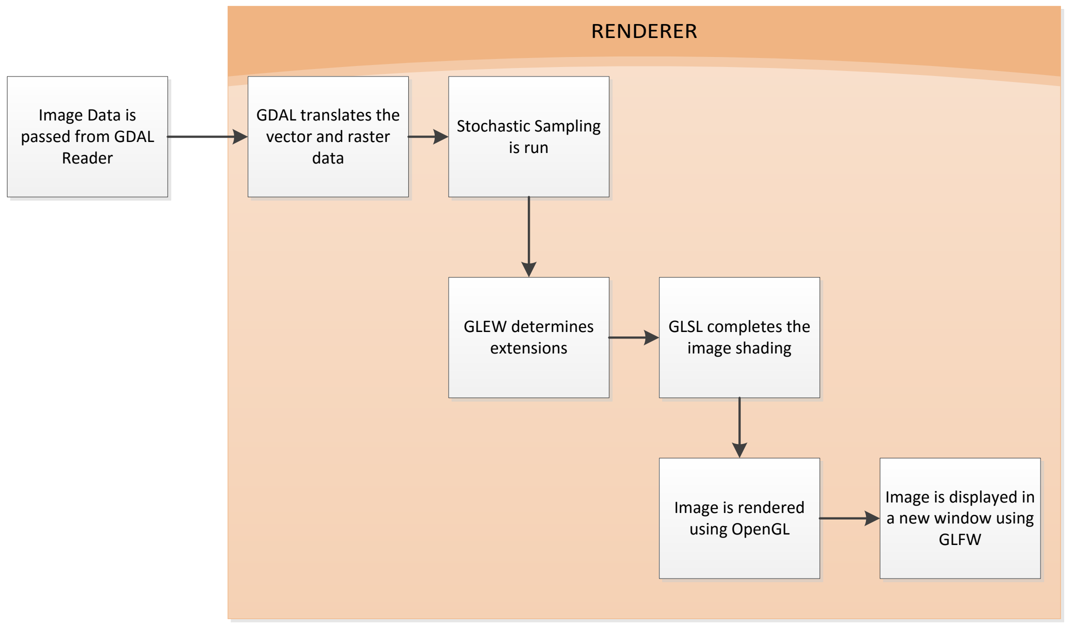
RENDERER
Image Data is passed from GDAL Reader
GDAL translates the vector and raster data
Stochastic Sampling is run
GLEW determines extensions
GLSL completes the image shading
Image is rendered using OpenGL
Image is displayed in a new window using GLFW

Figure 4: Renderer Component Diagram

3 IMPLEMENTATION PLAN

The delivery of this project is being done following a standard waterfall approach with the following phases of work: Analysis, Design, Development, Testing, and Implementation. The Requirements were a major output from the Analysis Phase of work. With this design document, the project is nearing the end of the Design Phase. The table below shows the estimated start and end dates for all of the phases of the project:

Task Name	Duration	Start	Finish	Predecessors
Gigapixel Images Project Schedule	**113 days**	**11/17/2015**	**4/21/2016**	
Development	**86 days**	**11/17/2015**	**3/15/2016**	
Ensure Understanding of Design and Technology	5 days	11/17/2015	11/23/2015	
Set up Environment	5 days	11/24/2015	11/30/2015	3
Optimize Reader Outputs	9 days	12/1/2015	12/11/2015	4
Implement an advanced "adaptive reading" feature	20 days	1/12/2016	2/8/2016	
Optimize and refactor the adaptive reader and rendering	53 days	12/14/2015	2/24/2016	5
Render GDAL Output	8 days	1/12/2016	1/21/2016	
Implement Image Viewer Features	30 days	1/22/2016	3/3/2016	8
Implement GUI Toolbar	8 days	3/4/2016	3/15/2016	9
Integrate GDALReader	8 days	3/4/2016	3/15/2016	9
Optimization	**25 days**	**3/16/2016**	**4/19/2016**	
Optimize and Add Additional Features	25 days	3/16/2016	4/19/2016	11
Testing	**25 days**	**3/16/2016**	**4/19/2016**	
End to End Testing	25 days	3/16/2016	4/19/2016	11
Implementation	**2 days**	**4/20/2016**	**4/21/2016**	
Demos and Training	2 days	4/20/2016	4/21/2016	15
Project Completed	0 days	4/21/2016	4/21/2016	17

Table 4: Project Schedule

The key steps and activities to perform to complete the project are described in the table below.

Phase	Activity Name	Description
Development	Ensure Understanding of Design and Technology	The development team will meet to review the Design Document (components and specific technology). Research will be performed as needed to fully understand technology and frameworks. Duration: 5 days
	Set up Environment	Developers will implement a primitive stochastic reader for the software. Duration: 5 days
	Optimize Reader Outputs	The developers will study the reader's outputs to find the most efficient outputs based on viewing area. Through trial and error, the team will find optimum settings to achieve a desired balance between quality and speed. Duration: 9 days
	Implement an advanced "adaptive reading" feature	The developers will build the advanced adaptive reading feature. This feature will analyze a section of the image and its color variation. Depending on the level of the variation the sampling rate will change. There will be a higher sampling rate for higher color variation and a lower rate for lower variation. Duration: 20 days
	Optimize and refactor the adaptive reader and rendering	The developers will test, fix bugs, and continually optimize the rendering process. Duration: 53 days
	Render GDAL Output	The developers will build and test the features that render the GDAL output in the OpenGL application. Duration: 8 days

Phase	Activity Name	Description
	Implement Image Viewer Features	The developers will build and test the following image viewer features: • An image panning feature for the software. Panning will take an input from the user's keyboard or other methods to allow the user to pan around the image while zoomed in. • An image zoom feature for the software. Image zoom will also take a user input, either from the keyboard or a mouse wheel to allow the user to view a section of the image with more clarity. • A "recently viewed" cache to allow for increased speeds. This will help speed up load times while viewing as the recently viewed images can be quickly added back when needed. Duration: 30 days
	Implement GUI Toolbar	The developers will build and test a toolbar for the GUI. A toolbar will allow for ease-of-use while interacting with our software. Duration: 8 days
	Integrate GDALReader	The developers will integrate the GDALReader component inside of GDAL. This integration will be tested. Duration: 8 days
	Optimize and Add Additional Features	This activity is optional and will be performed should time permit. The developers will continue to optimize, refactor, and implement optional features that are defined as part of this project. Duration: 27 days

Phase	Activity Name	Description
Testing	End-to-end Testing	The developers will build a plan and run tests checking all features for accuracy and performance. Duration: 20 days
Implementation	Demos and Training	The developers will work with the USGS analysts and Sponsor to install, demonstrate the project, and provide training. Duration: 2 days

Table 3: Project Activities

The diagram below shows the estimated project schedule for the remaining phases of the project in a Gantt chart format:

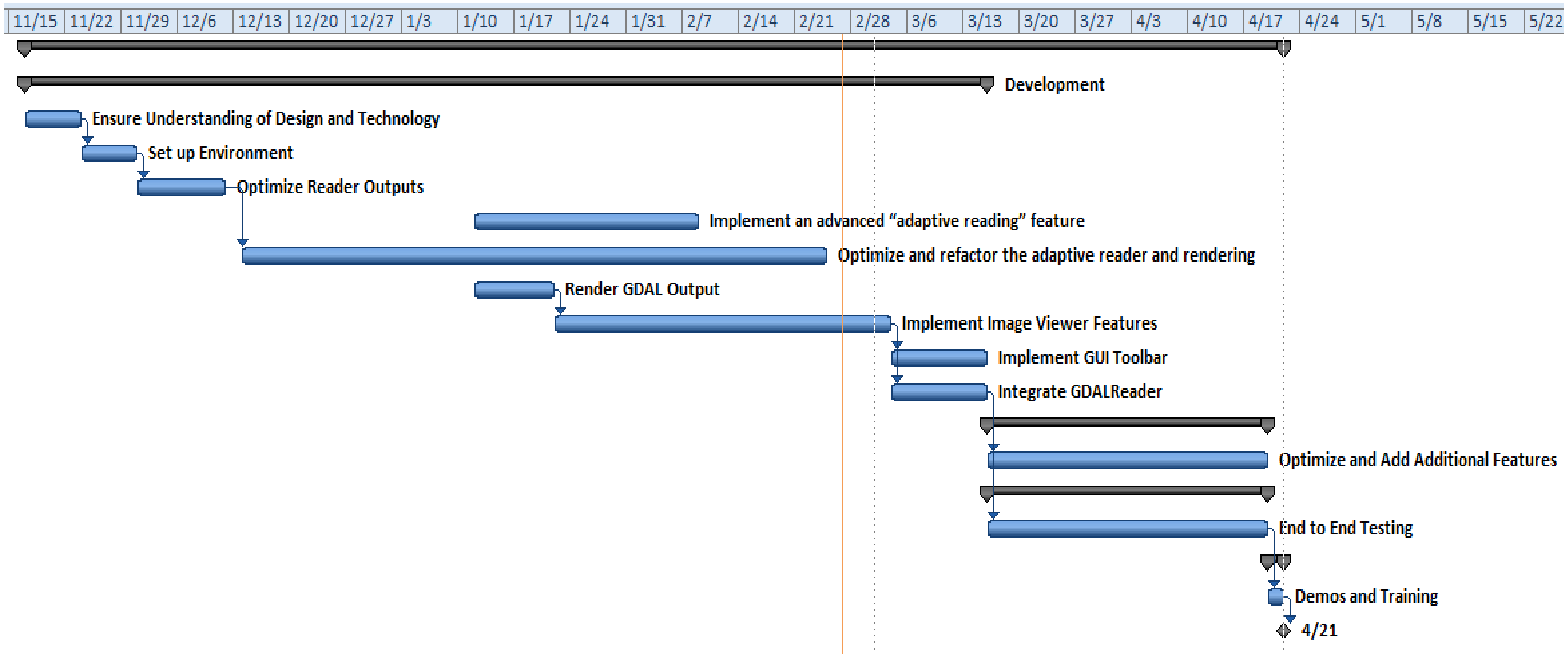

Figure 5: Project Schedule Diagram

4 Project Risks

Project risks are possible scenarios where the project is most likely to encounter trouble. These are potential hang-ups in the development process that could cause extra amounts of time or possibly failure. Fortunately for this project, there are only a few risks that have been anticipated and mitigations to these risks have been found.

Risk	Possible Impact	Mitigation
Incorrect sampling rate	This could affect image quality if too few samples are taken. Load time could be very long if too many samples are taken.	Test a variation of sampling rates to find a good balance between quality and speed.
High color variation in the image	Because our reader takes into account color variation to determine the sampling rate. A high color variation could lead to longer render times.	Adjust the amount of samples taken from an image with high color variation.

5 CONCLUSION

Overall, this is not a large project. However, we expect the implementation of the GDALReader component to take a majority of the development time that we have, therefore it will be well integrated. There is also the intention of having simple viewing enhancement functions such as scaling and basic zoom functions within the viewer. We also are using OpenGL along with GLEW and GLFW3 to use the most recent features to add on to the project. The final product for this project will be a lightweight image viewer paired with our fast image downsampling that uses a stochastic sampling approach. We also want to integrate the stochastic sampler (GDALReader component) inside the GDAL library so any entity using GDAL may benefit from our work. If all goes according to plan, we anticipate that this viewer will significantly improve load times for the images used at USGS and other agencies.