ASVGR (Advanced SVG Rendering)

Project Members: Clay Wagner
Advisor: Professor Annexstein
December 8, 2015

TABLE OF CONTENTS

PROJECT DESCRIPTION

Background

Display resolutions have increased exponentially over the past few years. Computer monitors with 3840 by 2160 resolution will soon become the standard for the average consumer. Additionally, phones are now exceeding 1920 by 1080 pixels, with the exact resolution depending on the height, width, and pixel densities of the phone.
The amount of different resolutions that applications need to support is greater now than ever before. If designers and programmers do not properly support different resolutions, applications may appear blurry, assets may not be properly positioned, or perhaps the application will not work correctly at all. In order to make sure their applications work correctly, the graphic designers of an application have to test their assets on many different devices in order to make sure their application works properly In the recent future, multiple different libraries, techniques, and APIs have assisted programmers in accounting for different resolutions and pixel densities.

Problem Statement

While programmers have some solutions to help minimize the amount of time spent on accounting for different resolutions, the same can't be said about graphic designers.
When designing assets for applications, graphic designers typically work with 2D vector image files (Adobe Illustrator Files or SVG files). 2D vector image files are described with mathematical equations instead of pixels, allowing them to scale properly to any resolution. This allows graphic designers to account for different displays.
The problem is that although graphic designers can use 2D vector images while creating assets for different resolutions, there is no proper way to render these 2D vector image files directly. Some libraries exist, such as Cairo and NVIDIA Paths Renderings, but these both have their share of lack of platform support, performance problems, and dependency problems. Because of this, artists have to export their 2D vector images to raster images. Raster image files are are a collection of pixels at a certain width and height. Scaling raster images to different resolutions and pixel densities will make them appear blurry. Depending on the amount of resolutions, pixel densities, orientations, and aspect ratios that an application has to support, the amount of times a 2D vector asset needs to be exported to a rasterized image could be enormous.

Team Members

Andrew Rose - roseaw@mail.uc.edu
Clay Wagner - wagnerc9@mail.uc.edu

Faculty Advisor

Professor Annexstein

Goal

The goal is to explore the possibility of exporting vector images into 3D image files. 3D image files are also described with mathematical vector equations, so although the actual file formats may be different, the concept of

a 2D vector image can easily be applied to 3D image files. Almost all modern platforms support some sort of 3D Graphics API, so applications should have no problem rendering these assets.
The final solution of this project would be to create an API for rendering vector images files. An application will also be designed for those who do not wish to use the API but want to export their 2D vector assets to common 3D image formats.

Subgoals

1. Use advanced GPU threading techniques
2. Use a cutting edge graphics APIs to help with performance
3. Create a parser for the most common 2D vector image format (SVG)

Helpful skills

- Graphics Programming,
- Software Engineering
- File Parsing
- Parallel Computing
- Vector Math
- Linear Algebra

DESIGN DIAGRAMS

Below are the design diagrams. Each diagram has its own legend to describe what each element of the respective diagram represents. The diagrams below go over the following subsystems:

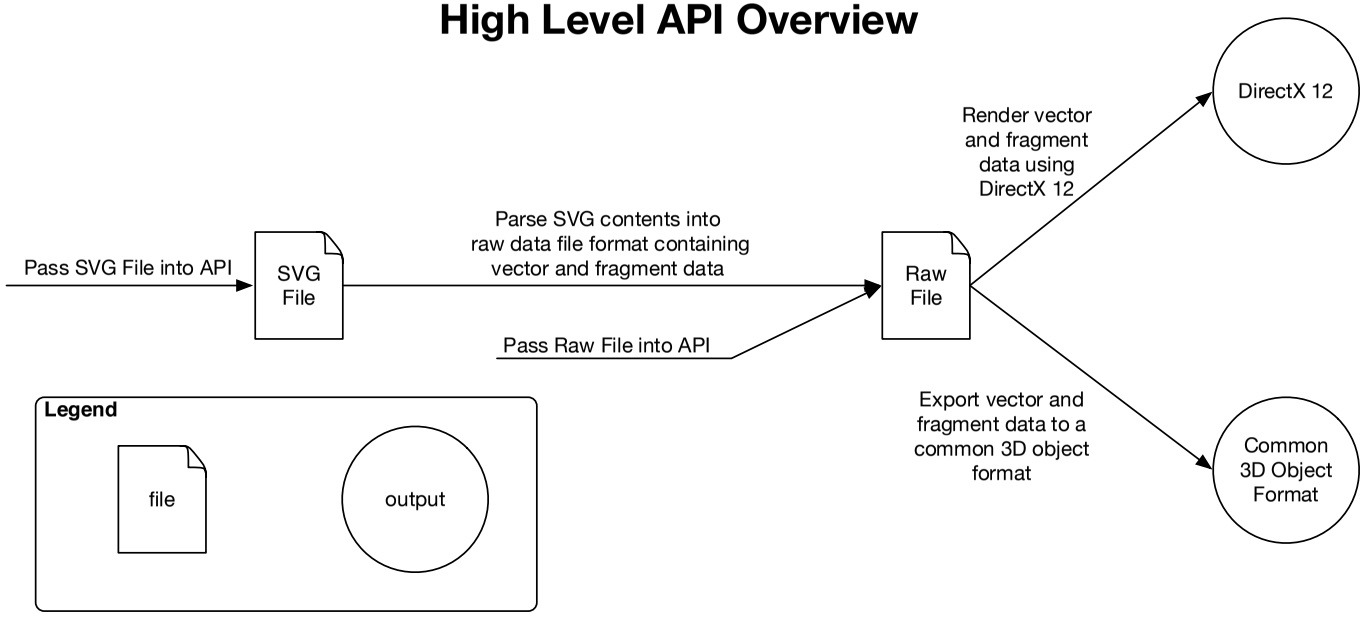
1. SVG Parser - The SVG Parser subsystem takes in an SVG file and exports a RAW file. The file structure for this format has yet to be decided, but it will contain allow necessary vertex and fragment (color, gradient) data from the SVG.
2. 3D Object Builder - The 3D Object Builder subsystem takes in a RAW file (described above) and generates a 3D object file from it. From there the 3D Object Builder can export to a common object file format, or be passed with additional information to the Renderer subsystem (described below).
3. Renderer - The Renderer subsystem takes in 3D object file and additional information from the 3D Object Builder subsystem. This additional data allows the renderer to do special thing when rendering the vector image (data that would otherwise be lost in a common object file format). This could include the rendering knowing how many and what vertices can be lost when rendering from a certain distance away from the camera, or how a gradient should be exactly rendered (as the common 3D object file format may lose some precision in image quality).

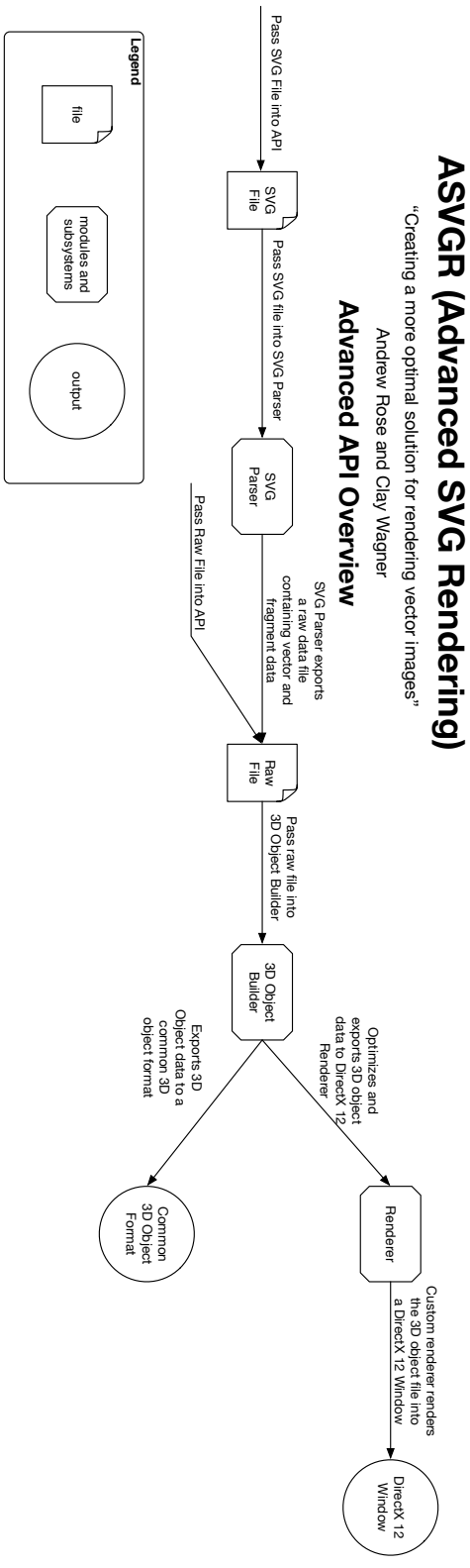
ASVGR (Advanced SVG Rendering)

"Creating a more optimal solution for rendering vector images"

Andrew Rose and Clay Wagner

High Level API Overview



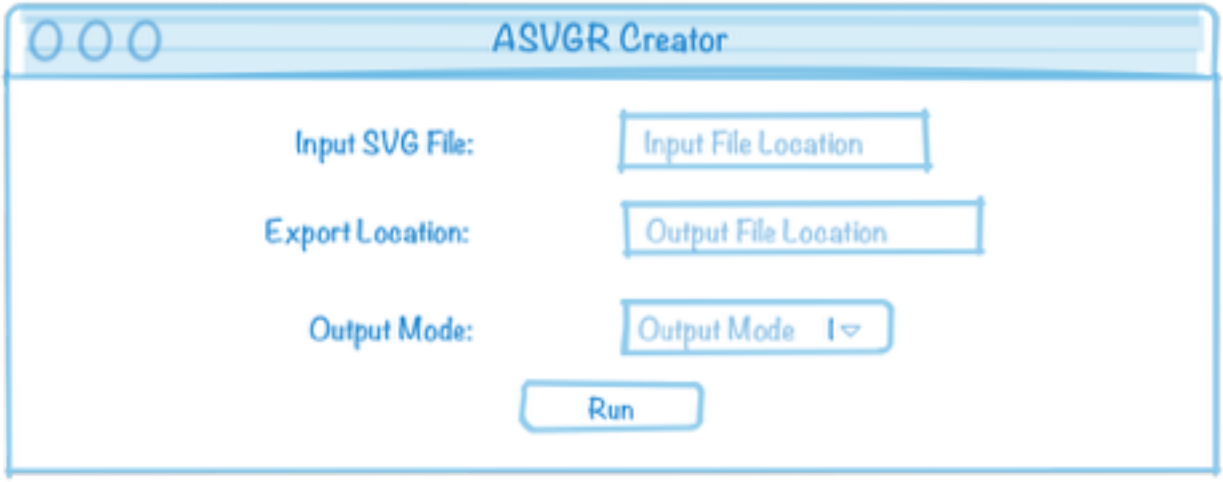

ASVGR (Advanced SVG Rendering)
"Creating a more optimal solution for rendering vector images"
Andrew Rose and Clay Wagner
Advanced API Overview
Pass SVG File into API
SVG File
Pass SVG file into SVG Parser
SVG Parser
SVG Parser exports a raw data file containing vector and fragment data
Pass Raw File into API
Raw File
Pass raw file into 3D Object Builder
3D Object Builder
Optimizes and exports 3D object data to DirectX 12 Renderer
Exports 3D Object data to a common 3D object format
Renderer
Common 3D Object Format
Custom renderer renders the 3D object file into a DirectX 12 Window
DirectX 12 Window
Legend
file
modules and subsystems
output

ASVGR (Advanced SVG Rendering)
"Creating a more optimal solution for rendering vector images"
Andrew Rose and Clay Wagner
Detailed API Overview
Pass SVG File into API
SVG File
Pass SVG file into SVG Parser
SVG Parser
SVG Parser exports a raw data file containing vector and fragment data
Pass Raw File into API
Raw File
Pass raw file into 3D Object Builder
3D Object Builder
Exports 3D Object data to a common 3D object format
Common 3D Object Format
Optimizes and exports 3D object data to DirectX 12 Renderer
Renderer
Custom renderer renders the 3D object file into a DirectX 12 Window
DirectX 12 Window
SVG Parser
Pass SVG File into Parser
SVG File
Read SVG file into a DOM-like structure
DOM structure
Extract SVG Data
Pass Raw File into SVG Data
SVG Data
Export to Raw File
Raw File
3D Object Builder
Pass Raw File into 3D Object Builder
Raw File
Convert data into vector and fragment data (lines, objects, curves, gradients, color)
Intermediary Graphic Data
Combine objects
3D Object Data
Optionally export to a common 3D object file format
Common 3D Object Format
Renderer
Pass 3D Object data into Renderer
3D Object Data
Pass data through graphics pipeline
Shader
Graphics Card renders 3D Object data
DirectX 12 Window
Legend
file
modules and subsystems
output
module or subsystem detail view
Intermediate Data

INTERFACE SPECIFICATIONS

The ASVGR API is the main focus of this project. A developer will use this API to create 3D objects they can render in their programs out of SVG images. The below programs (ASVRG Creator and ASVGR Viewer) display the types of things developers can do with the API. ASVGR Creator shows the ability of the API to output 3D objects to a file and ASVGR Viewer shows the ability of the API to directly render an SVG file using DX12.

ASVGR Creator

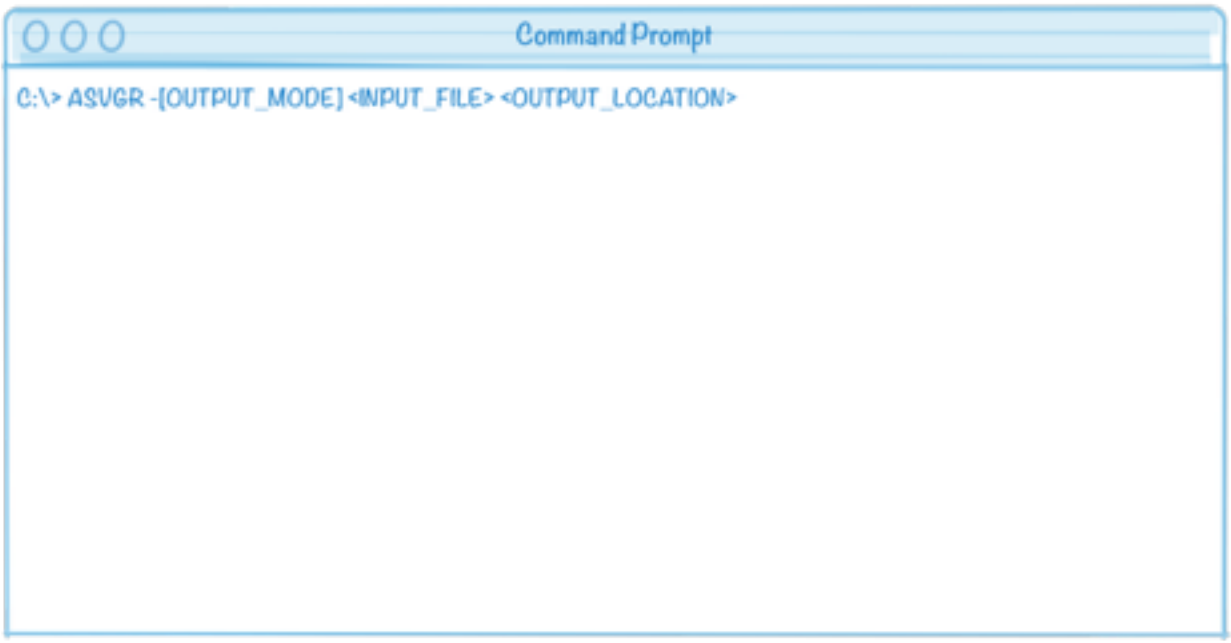


ASVGR Creator is a program that we will create to use the ASVGR API we are writing. It will show off the ability of the API to take an SVG image and output a 3D object that can be used to render the SVG image. For the output the user will be able to choose between standard 3D object formats and the specialized RAW format we are designing for the API.

Terminology

Input SVG File - The SVG file that will be taken by the API and turned into a 3D object.
Export Location - The location on the hard drive to output the 3D object file.
Output Mode - The format to output the 3D object created by ASVGR.

ASVGR Command Line Interface

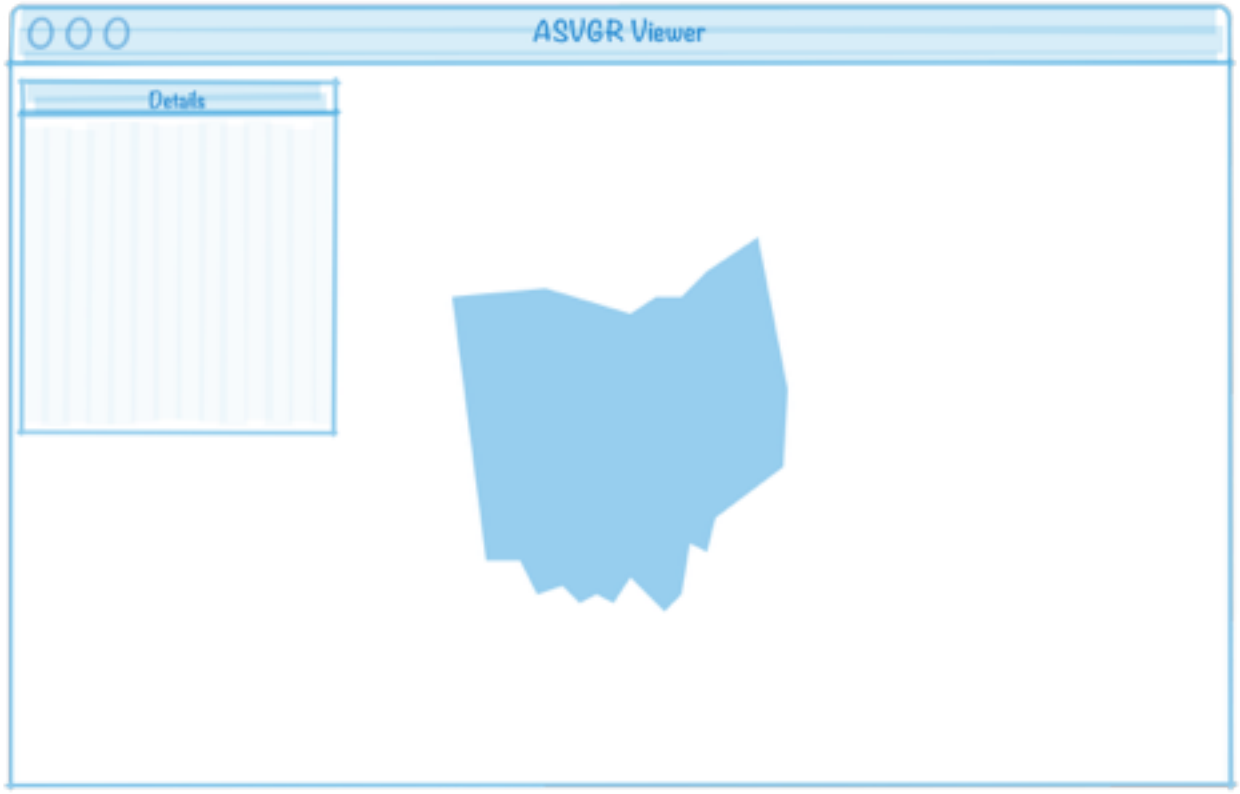


This is a command line interface for the ASVGR Creator program shown above. It will have all the same capabilities as the GUI program. An SVG file will be taken as input and a 3D object will be created at the output location in the specified format. Being on the command line this will enable developers to automate processes through scripts.

Terminology

OUTPUT_MODE - The format to output the 3D object created by ASVGR.
INPUT_FILE - The SVG file that will be taken by the API and turned into a 3D object.
OUTPUT_LOCATION - The location on the hard drive to output the 3D object file.

ASVGR Viewer



ASVGR Viewer will be a program that can be used to view a SVG file using the ASVGR API. It will display the image of the object the API created and details about that object. This application will use the DX12 portion of the API to directly render the 3D object. Details about the objects may include things like number of points/lines, size in memory, and size.

PROJECT TASK LIST

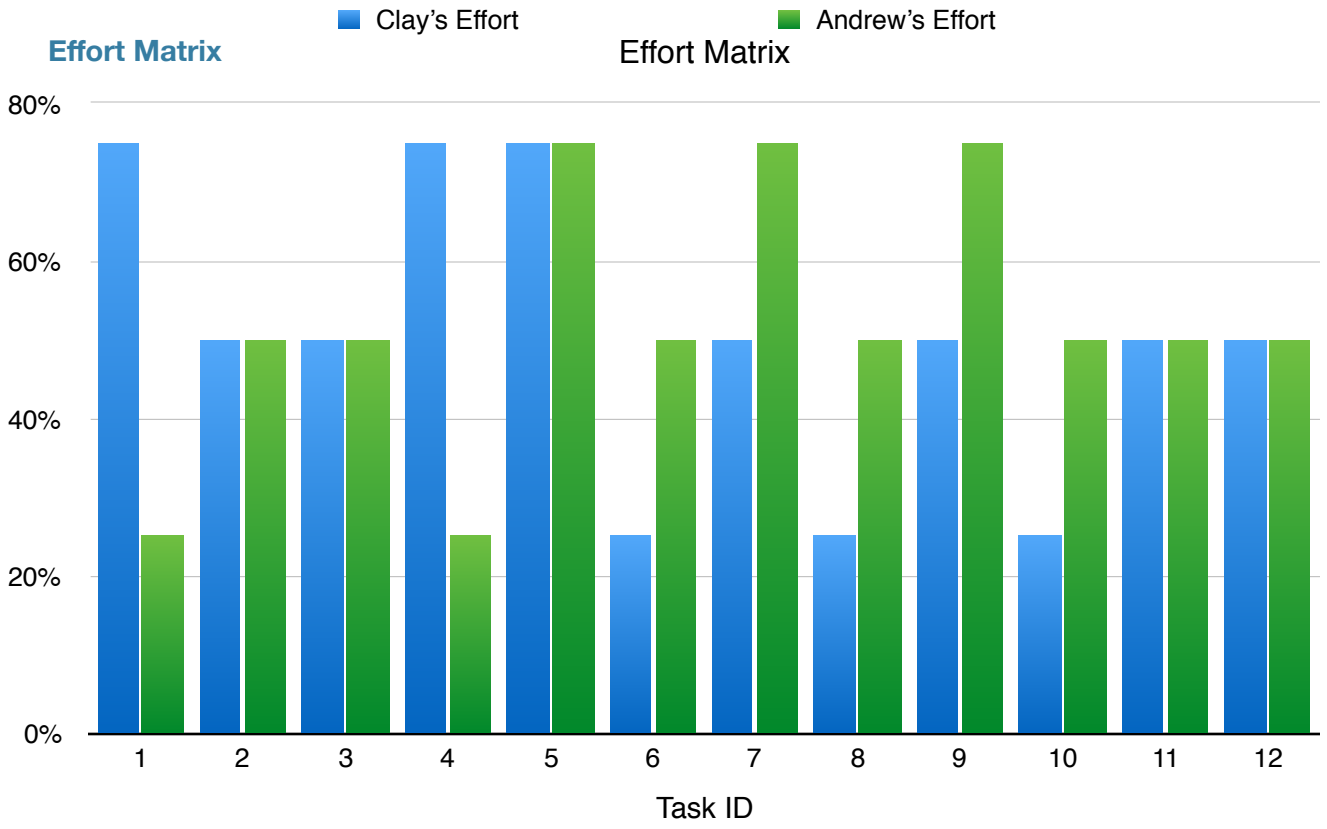
Task List

Task ID	Task Name
1	Setup organization system (kanban board, document storage, conversation programs)
2	Discuss and setup programming milestones
3	Research and reverse engineer SVG 1.1 Specification
4	Create SVG 1.1 specification data structure
5	Create SVG Parsing API
6	Research DirectX12 API
7	Research various 3D Object file formats
8	Create new Raw file format specification based on DirectX12 and 3D Object research
9	Implement 3D Object Builder API
10	Implement DirectX12 Rendering API
11	Implement 3D Object Format Export API
12	Create demos to show examples of our API

Timeline

Task ID	Duration	Start	Finish	Predecessors	Clay's Effort	Andrew's Effort	Completion
1	6	10/4/15	10/10/2015		75%	25%	100%
2	13	10/18/2015	10/31/2015		50%	50%	100%
3	55	10/4/2015	11/28/2015		50%	50%	80%
4	24	11/18/2015	12/12/2015	3	75%	25%	70%
5	27	12/13/2015	1/9/2016	3, 4	75%	75%	50%
6	14	11/1/2015	11/15/2015		25%	50%	10%
7	14	11/1/2015	11/15/2015		50%	75%	0%
8	13	11/15/2015	11/28/2015	6, 7	25%	50%	10%
9	41	11/29/2015	1/9/2016	8	50%	75%	0%
10	41	11/29/2015	1/9/2016	5,6	25%	50%	0%
11	20	1/10/2016	1/30/2016	8	50%	50%	0%
12	27	1/10/2016	2/6/2016	5,9,10,11	50%	50%	0%

Effort Matrix



ABET CONSTRAINT ESSAY

Economic

Even though we plan for our project to be open sourced, our team will develop ASVGR (Advanced SVG Rendering) for a target audience. Our team desires to create a clear and compact solution for software organizations, independent developers, and graphic designers alike. We have and will continue to tune our project to meet what we think are our target audience's desires. We also plan on making sure that ASVGR will have no paid dependencies required in order to fully utilize it. For this reason we are planning to use graphics APIs that are free to develop with, such as Microsoft's DirectX 12. Using DirectX 12 will constrain parts of the project to Microsoft only devices, but we are structuring the project in a way that will allow us to support additional graphic API's that target other platforms. This means that ASVGR will require no external programs for running, and its 3D Object Exporter will export to common 3D object format that can be opened by free 3D modeling software.

Professional

Our project will require that teammates have an advanced understanding in graphics programming as well as significant experience in parallel programming. We will have to account for the time it will take to fully learn and

utilize a brand new API such as DirectX 12. DirectX 12 is a brand new state of the art graphics API that significantly increases in speed compared to older versions of the library. Learning DirectX 12 is likely to help out team members in their professional lives, since they all have extreme interest in areas where it is used. If successful, the project would be very helpful to many graphics developers and it would give team members an example of their expertise in the areas of graphics programming and API design.

Legal

Our project may have some legal constraints as well. A few years ago NVIDIA released the NVIDIA Paths extension for OpenGL. It allows developers to render bezier paths (including SVG files) with GPU optimization. Their method "Stencil and Cover" has been heavily patented. However, their method involves actually taking in an SVG file or string, processing it on the GPU while embracing the benefits of extreme parallelization with CUDA, and then rendering it with OpenGL. Our method is very different from the method entailed by NVIDIA. Our method will involve converting the SVG file into an intermediary file of vector and fragment data arrays. This data will then be used to either create a 3D object file, or the data will be rendered through our optimized DirectX12 renderer. When converting this data, we will be using optimizations created by either our team with some possible influence from other open sourced projects (such as Cairo and OpenVG). However, we will need to pay careful attention to NVIDIA's patents during development to make sure that we do not violate them.

Social

As stated above, we plan for our project to be open sourced. Therefore, when writing source code, we need to keep in mind that we have an audience who also needs to be able to understand and utilize the ASVGR API. This will entail clean code, very clear and concise documentation, and a few demos to help developers get started. We will also need to carefully consider under what type of license we wish to release the project under. This will affect potential users of the API in what exactly they are allowed to do with it. Examples of licenses we will probably consider are the BSD or MIT License due to the ability of the end user to create derivative works for commercial applications.

GROUP STATUS REPORT SLIDES

ASVGR

CREATING A MORE OPTIMAL SOLUTION FOR RENDERING VECTOR IMAGES

PROJECT INFORMATION

- Team Members
  - Andrew Rose
  - Clay Wagner
- Advisor
  - Dr. Annexstein

RASTER IMAGE VS VECTOR IMAGE

- Raster Image
  - A collection of pixel and color information
- Vector Image
  - A collection of paths, gradients, and shapes
  - Described using mathematical formulas

ABOUT THE PROJECT

- Resolution and Pixels per Inch (PPI) of displays continue to increase
- The amount of different resolutions, aspect ratios, and PPI that developers need to support is greater than ever
- This has led software developers and designers to turn towards vector images to prevent redoing assets

ABOUT THE PROJECT

- Designers rasterize vector images for appropriate resolutions, aspect ratios, and PPIs
  - May need to do this for each supported device
  - Support for future devices may require designers to constantly go back and rasterize new images
- Certain applications exist that may need to rasterize vector images dynamically

WHY USE 3D RENDERING APIS

- Current 2D Rendering APIs are slow
  - Most APIs do not utilize graphics cards effectively
  - Most APIs are not multithreaded
- 3D APIs utilize graphics cards effectively
- DirectX 12 and Vulkan both support multithreading
- Current graphics cards are optimized for 3D Rendering

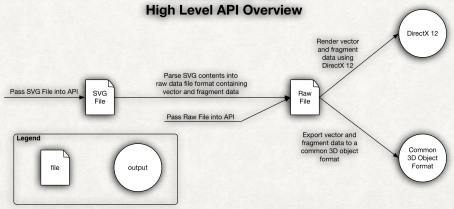
INTRODUCING ASVGR

ASVGR (Advanced SVG Rendering)

"Creating a more optimal solution to rendering vector images"

Andrew Rose and Clay Wagner

High Level API Overview



DESIGN DIAGRAMS

USER INTERFACE

CURRENT 3D SOLUTIONS

- OpenGL Rendering Raster Image to Texture
  - Take an image (or rasterized image) and render it on a rectangle.
  - If rasterized vector image, all data describing image is lost.
  - Only looks good if render environment is perfect
- Autodesk Scaleform
  - Renders using Flash API
  - Expensive
  - Not open source
- Nvidia "Paths" OpenGL Extension
  - Only supported on NVIDIA graphics cards
  - Not open source
  - Not updated regularly

CURRENT PROGRESS

- Project
  - Flushed out project goals
  - Documentation
- Research
  - SVG File Structure
  - Direct X 12 Rendering
  - 3D Object File Formats
- Code
  - Parser nearly complete
  - Preliminary rendering and object exporting work has begun
- Issues
  - Time Constraints

WHAT'S NEXT?

- Both of us will finalize RAW data format
- Andrew will focus on DirectX 12 Rendering portion of project
- Clay will focus on 3D Object Exporting portion of project


QUESTIONS?

SELF ASSESSMENT ESSAYS

Andrew Rose

I am in a group with Clay Wagner and we are working on designing and implementing a library that allows developers to work with SVG (Scalable Vector Graphics) files directly rather than have to rasterize them before hand. We are hoping that this will solve the problem of having to have different sized images for different screen resolutions and device sizes. Currently a large bulk of time and space is spent creating and storing the various graphics required for applications and games to look good on all devices. This project will look at solving this problem with regards to texturing in games. So that instead of using many differently sized textures with differing level of details developers could just use one SVG image with our library to display textures at any size and resolution. If this is successful this could eventually be expanded to general UI use cases.

We will need to use information we gained while taking classes like Computer Graphics, UI, Algorithms, Parallel Computing, and Linear Algebra to complete this project. In Computer Graphics we learned the basics of the graphics pipeline and how to apply it to create an application. I do some Graphic related things as hobbies and have a background in using OpenGL and DirectX API's. This project will require us to use these low level API's to efficiently render data on output devices. We will have to develop a few algorithms to quickly/correctly create 3D objects out of SVG data. Parallel Computing will come in handy because some of the things we might need to do will be computationally heavy. I have taken all of the classes necessary, including Linear Algebra, to get a Math minor from UC. This will come in handy for this project since we will be doing a lot of work with vectors and math equations to correctly build objects from SVG data.

I worked for a single company (Siemens PLM) for all 5 semester of my co-op experience. I mainly dealt with complex legacy code and adding functions/classes to various libraries they had created over the years. Most of the code was in C/C++. I have seen first hand what bad API's look like and have a better understanding of how to properly structure one so that it works well. Also having done a lot of work in C/C++ it will help because those languages are what DirectX/OpenGL use. The team I worked with at Siemens was a large one that spanned three continents and contained a very diverse group of people. This was probably one of the greatest benefits of my experience on co-op. It helped me to learn to communicate effectively via most forms of communication with people who don't always have the same background as you. Being able to communicate effectively is a cornerstone of any team project.

This project is extremely interesting to me because it is primarily based in areas that I enjoy to work. It is an interesting idea to solve a rather complex problem. We feel that there has yet to be a usable solution to this problem. Our first approach will be to first write a parser for the SVG language that parses the data into a format more easily used to create 3D objects. Then we will attempt to create those 3D objects. Once we have figured out how to do the basics we will start to actually implement an API gives developers the ability to render SVG graphics using our library.

The result of this project should be a library that a developer can use to render SVG files. For this project we are only supporting the DirectX graphics library to directly render the SVG files, but our API will also enable

developers to get a 3D object from a SVG file. This will enable developers that use OpenGL to take advantage of our library as well. In order for this library we are creating to work it will need to be able to render the graphics at a reasonable framerate on modern hardware. We will know we are done and have done a good job when we have implemented a library that correctly renders SVG files to an output device as fast as current hardware supports. The library should have a developer friendly API that other developers can easily and quickly use to render SVG files. We will be doing some very process intensive work to correctly create and render the SVG files, so we will have to be mindful to squeeze as much performance as possible out of everything.

Clay Wagner

The senior design project that I will be working on with Andrew Rose is to develop an API to render vector graphic files with modern rendering APIs such as DirectX12. Vector images are different than the majority of images you see every day. Vector images are described with mathematical functions, allowing them to be rendered at different resolutions and pixel densities while retaining the quality of the image. A rasterized image at different resolutions will appear tiny or blurry, depending on the size of it. In order to get around this, graphic designers will render their design out at multiple different resolution, so that it will look correct on the respective resolution. The problem is that in the modern world, there hundreds of different resolutions, screen sizes, pixel densities that can make up a display device. Rendering vector images with modern rendering APIs will cut down on the amount of hours artists and programmers spend on repeatedly rendering their vector images to rasterized images for different resolutions. Although attempts have been made at this before, the current solutions perform very poorly due to using graphics APIs with a significant amount of overhead. If we use a modern rendering API such as DirectX12, we will be able to avoid this overhead and have better performance than previous attempts have had.

My experience both in school work and co-op have prepared me for the creation of my senior design project. My course in Graphics Programming with Dr. Han will help immensely in the design of this senior project, as this project is a graphics programming project. My course in data structures and algorithms will help with design of a parser for the SVG file. Additionally, these courses will help me create efficient algorithms for rendering. The main way to increase performance with graphics rendering is to reduce the amount of draw calls. Using the information I learned from data structures and algorithms well help me limit these calls.

My co-op experience will also help my immensely in the design of my senior project. During my co-op experience I had opportunities to create parsing tools, create efficient and complicated algorithms, work with graphics programming. Most importantly I learned how to truly design real world software and how to manage real world software And API projects. Over the past 6 months I have acted as an assistant manager for a project. It has been my job to help develop features, delegate these features, find and track issues, and review people's code. While I don't think I will need a super organized structure for a two person team, I do think that my ability to manage large software projects will greatly help with my senior design project.

I am extremely excited to start this project with Andrew Rose. This project has been something I have wanted to do for years. The main issue I had before is that graphics libraries (until recently) have had a large amount of overhead and had many threading issues. Now, the technology has caught up for this project to be possible with the introduction of DirectX12, Metal and Vulkan (Vulkan is coming out later this year). I truly believe vector images are the future of both art and user interfaces. If graphics card manufacturers are focusing on optimizing 3D API development, than the best way to draw these vector images are by utilizing the 3D APIs. This will most likely be one the first attempts of rendering SVGs using the newest graphics APIs.

My preliminary approach to designing a solution will be researching the new APIs and also researching the specification file for the SVG file format. If we can understand these very well, we should be able to create an initial API. After that, all that would be left would be bug fixing, extra features (such as exporting to other formats), and performance optimizing. Andrew and I need to create an overview of the major items that need to be done. I think that the majority of the time Andrew and I will be working together but if need be, we should make sure we delegate tasks fairly. I also think that if Andrew and I create a schedule of flexible deadlines and due dates we will reach our expected solution. I expect our final result to be an API that renders SVG files almost perfectly in DX12, with some thought put towards optimization. Along with this API, I hope to include a few demos that utilize our API and show off what it can do. I think I will have done a good job with the project if Andrew and I are spending the last few weeks before the deadline implementing demos, meaning we successfully followed our own deadlines, we will know that we have done a good job.

PROFESSIONAL BIOGRAPHIES

Andrew Rose

Co-op or Other Experience and Responsibilities

- TCIN Developer, Siemens PLM, Milford Ohio (5 Semesters)
  - Reconfigured entire testing library to run more efficiently
  - Redesigned save-as functionality for older feature so it was compatible with newer ones
  - Worked with people in Pune daily on various projects

Skills/Expertise Areas

- Programming: C++, Java, Python
- Operating System: Unix, Linux, Windows, Mac
- Web Development: HTML, CSS, JavaScript
- Office Applications: Microsoft Office 2010
- Libraries: OpenGL, WebGL, DirectX 11
- Other: UE4

Areas of Interest

- Artificial Intelligence
- Neural Networks
- Game Design
- Cloud Computing

Type of Project Sought

- Procedural Generation of Game Environment
- Neural Networks for use in games
- Processing of vector images

Clay Wagner

Co-op Experiences

- Software Development Co-op, Kinetic Vision, Cincinnati, Ohio (2 Semesters):
  - Designed and developed numerous engineering focused applications, user interfaces, and technologies for various clientele.
  - Helped manage and delegate development tasks for other employees at the company. o Directly collaborated with and presented to clientele about various projects.
- Engineering Intern, Signal X Technologies, Northville, Michigan (2 Semesters):
  - Developed a report generation tool for a client where the reports visually represented data processed in the main application.
  - Assisted with development on an application to operate a machine which ran a durability test on car seats.
  - Designed and tailored the user interface elements for various testing applications.
  - Developed a web application for an oil drilling client, where the application displayed drilling data graphs in a high quality and professional user interface. Users had the ability to scroll through data, change which channels they are viewing, and also toggle live updating.
  - Developed elements of both the Spectrum Graph and Waterfall Graph for MajX DSA 2.9, the new version of the company's sound and vibration measurement and analysis tool. o Designed and developed a touch screen application that operated a tire balancing test.

Skills/Expertise Areas

- Programming Languages: C#, C++, C, Python, Objective-C, HTML, JavaScript, CSS
- Operating Systems: Mac OS X, Windows, Linux, Unix
- Libraries and Technologies: OpenGL, .NET, DirectX, Unreal Engine 4
- Office Applications: Microsoft Office, iWork suite, Google Apps Suite

Areas of Interest

- Graphics Programming
- Interactive Media
- Front-End Design
- Parallel Computing

Type of Project Sought

- Developing an application that converts vector image files (SVG files) into vector and fragment (color) data, rendering them directly with a graphics API (OpenGL, DirectX, or DirectX 12). This application could also have the possibility to export to well-known 3D object file formats.
- A cross platform material design (Google's design specification) user interface API rendered using a graphics API (OpenGL, DirectX).

BUDGET

ASVGR (Advanced SVG Rendering)

Senior Design

Invoice # 10001

Invoice Date: 12/07/15
Services Thru: 12/07/15

Date	By	Type	Service Summary	Hours/Qty	Rate	Amount
In Reference To:		**Non Project Related (Professional Services)**				
08/31/15	AR	Documentation	Professional Biography.	1.00	75.00	$75.00
08/31/15	AR	Meeting	Discussed initial idea.	1.00	75.00	$75.00
08/31/15	CW	Meeting	Initial Meeting	1.00	75.00	$75.00
08/31/15	CW	Documentation	Professional Biography	1.50	75.00	$112.50
09/14/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
09/14/15	AR	Documentation	Project Description.	2.00	75.00	$150.00
09/14/15	CW	Meeting	Initial Meeting	1.00	75.00	$75.00
09/14/15	CW	Documentation	Project Description	2.00	75.00	$150.00
09/21/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
09/21/15	AR	Documentation	Self Assessment Essay.	1.50	75.00	$112.50
09/21/15	CW	Documentation	Self Assessment Essay	1.00	75.00	$75.00
09/21/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
09/23/15	CW	Direct X Research	General research	1.00	75.00	$75.00
09/23/15	CW	Exporter Researcher	Researched the FBX file format	1.00	75.00	$75.00
09/25/15	AR	Direct X Research	Research DX12 basics for presentation due monday.	2.00	75.00	$150.00
09/26/15	AR	Direct X Research	Research DX12 basics for presentation due monday.	0.50	75.00	$37.50
09/26/15	AR	SVG_Research	Researched the SVG Specification.	1.00	75.00	$75.00
09/26/15	CW	SVG_Research	Researched the 1.2 Specification Document	3.00	75.00	$225.00
09/27/15	AR	SVG_Research	Research the SVG Specification.	1.00	75.00	$75.00
09/27/15	AR	Direct X Research	Research DX12 basics for presentation due monday.	0.50	75.00	$37.50
09/27/15	CW	SVG_Research	More specification research	0.50	75.00	$37.50
09/27/15	CW	Exporter Researcher	OBJ File formaat	1.00	75.00	$75.00
09/28/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
09/28/15	AR	Documentation	DX12 Class Presentation.	1.00	75.00	$75.00
09/28/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
09/30/15	AR	SVG_Research	Research the SVG Specification.	0.50	75.00	$37.50
09/30/15	CW	Documentation	Class Presentation	2.00	75.00	$150.00
10/01/15	AR	Documentation	Design Diagrams	1.00	75.00	$75.00
10/02/15	AR	Documentation	Design Diagrams	1.00	75.00	$75.00
10/02/15	CW	Documentation	Class Diagrams	2.00	75.00	$150.00
10/05/15	AR	Documentation	Design Diagrams.	2.00	75.00	$150.00
10/05/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00

Date	By	Type	Service Summary	Hours/Qty	Rate	Amount
10/05/15	CW	Documentation	Class Diagrams (Rework due to better program)	3.00	75.00	$225.00
10/05/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
10/12/15	AR	Documentation	Did the Task List.	1.00	75.00	$75.00
10/12/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
10/12/15	CW	Documentation	Task List	0.50	75.00	$37.50
10/12/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
10/19/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
10/19/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
10/19/15	CW	Direct X Research	Researched Compute Shaders and Threading	2.00	75.00	$150.00
10/26/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
10/26/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
10/27/15	AR	Documentation	Began Work on the Effort Matrix.	2.00	75.00	$150.00
10/31/15	CW	Parser Programming	Began parser programming	3.00	75.00	$225.00
11/01/15	CW	Parser Programming	Minor parser programming	1.00	75.00	$75.00
11/02/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
11/02/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
11/06/15	AR	Documentation	Finalized and Submitted the Effort Matrix.	0.50	75.00	$37.50
11/09/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
11/09/15	CW	Documentation	ABET Constraint Essay	2.00	75.00	$150.00
11/09/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
11/09/15	CW	Parser Research	Need to research parser more to continue programming.	1.00	75.00	$75.00
11/10/15	AR	Documentation	Worked on ABET Constraint Essay.	2.00	75.00	$150.00
11/10/15	CW	Documentation	ABET Constraint Essay	1.50	75.00	$112.50
11/12/15	AR	Direct X Research	Researched DX12.	2.00	75.00	$150.00
11/16/15	AR	Meeting	Weekly Meeting	1.00	75.00	$75.00
11/16/15	CW	Meeting		1.00	75.00	$75.00
11/17/15	AR	Direct X Research	Researched DX12.	1.00	75.00	$75.00
11/21/15	CW	Documentation	UI Specification	3.00	75.00	$225.00
11/23/15	AR	Documentation	Finish up the Interface Specification.	2.00	75.00	$150.00
11/23/15	AR	Meeting	Weekly Meeting.	1.00	75.00	$75.00
11/23/15	CW	Documentation	Finished Up UI Specification	0.50	75.00	$37.50
11/23/15	CW	Meeting	Weekly Meeting	1.00	75.00	$75.00
11/29/15	CW	Parser Programming	Minor Parser Programming	2.00	75.00	$150.00
11/29/15	CW	Other	Setup Ebility	1.50	75.00	$112.50
11/29/15	CW	Documentation	Rerecorded hours in ebility	1.00	75.00	$75.00
11/30/15	AR	Documentation	Work on time sheet, meeting notes, and presentation for class.	3.00	75.00	$225.00
11/30/15	AR	Meeting	Weekly Meeting.	2.00	75.00	$150.00
11/30/15	CW	Meeting	Weekly Meeting	2.00	75.00	$150.00
12/01/15	AR	Documentation	Finish up final meeting notes.	1.00	75.00	$75.00
12/02/15	AR	Documentation	Finish up final meeting notes.	2.00	75.00	$150.00

Date	By	Type	Service Summary	Hours/Qty	Rate	Amount
12/02/15	CW	Documentation	Worked On Final Documentation	1.00	75.00	$75.00
12/03/15	AR	Documentation	Final Fall Design Report.	2.00	75.00	$150.00
12/03/15	AR	Documentation	Worked on Raw Format.	0.25	75.00	$18.75
12/03/15	CW	Documentation	Worked On Final Documentation	2.00	75.00	$150.00
12/05/15	AR	Documentation	Worked on Raw Format.	0.25	75.00	$18.75
12/06/15	AR	Documentation	Worked on Raw format.	0.50	75.00	$37.50

Total Hours:	102.50
Total Labor:	$7,687.50
Total Invoice Amount:	**$7,687.50**
Total Amount Due:	**$7,687.50**