

Open Source Library for Visual Data

Project Team 19

Matthew Guerrero (001331823)

George Tackie (001295771)

Ben Twomey (001454410)

Justin George (001409006)

.....

College of Engineering and Applied Sciences
University at Albany, SUNY

Project Sponsor

Michael Phipps

University at Albany, SUNY

1400 Washington Ave, Albany, NY, 12222

15-05-2021

Acknowledgements

For our acknowledgements, we would like to thank our sponsor Professor Michael Phipps for taking the time out of his day to answer all of our questions. Professor Phipps was able to provide detailed insight on how to properly implement the design for our open-source library. Professor Phipps was also shown each milestone we made to our design and provided positive feedback.

Abstract

We are looking to create an open source library that allows us to change both the software program and the user experience of the application. We provided a framework utilizing C# and SkiaSharp that developers can use to create their own software program. Those who choose to use the framework in the future will be able to create their own software or make an update to the current application to fit their requirements. This report will detail the proposed system application which includes the system requirements, system implementation, technical design, and testing results of the experiments performed to verify functionality of the framework.

Contents

1	Introduction	2
2	Background and Related Work	4
3	Proposed System/Application	6
3.1	An Overview	6
3.2	System Requirements	7
3.3	Technical Design	8
3.4	System Implementation	9
3.5	Use of Computer Science Theory and Software Development Fundamentals . .	9
4	Experimental Design and Testing	11
4.1	Experimental Setup	11
4.2	Results and Analysis	13
4.3	Limitations	14
5	Ethics and Legal Practices	15
6	Effort Sharing	16
7	Conclusion and Future Work	18
	Bibliography	19

1. Introduction

There is a need for an open source library that will allow developers to create applications based around structured drawing. This is an important problem as current applications such as Microsoft Visio and Creately use proprietary software making it impossible for a developer to modify or enhance the application based on end user requirements. This problem is challenging because there are no open source library systems we could use as a baseline framework. We created a framework to provide our users the ability to create a system or application.

Existing solutions such as Microsoft Visio and Creately provide an application already for a user, however there are no developer tools within these applications that allow for software modifications. Both these tools are insufficient because they do not allow a user to update the software based on their needs. Their limitations include being one dimensional and expensive for a user that might not need the full suite of the software program.

Many problems can be solved visually. Our core idea was to provide a framework for users to have the freedom and accessibility to use the open source library for solving problems visually. The framework provides users the ability to apply modifications to the source code allowing them to create visual applications.

The key characteristics of our solution is to provide an open source library that can be tailored to any developer's needs. This means the system must be able to handle modifications to the source code. A proof of concept was completed to prove that the framework works as intended and is easy to understand.

The structure of the project followed a Presentation-Abstraction-Controller (PAC) model, which is a hierarchy of class agents all connected to the main controller. This model arranged our code in three specific areas which were the view (presentation), the data (abstraction) and the mediator (controller). The forms for each window for the graphical user interface (GUI) was contained in the presentation of this model. Each script for the shapes, connections and file import and export classes were abstractions of the model. The controllers in our model were the shape controller and the main application scripts.

As for the structure of our project, we mentioned earlier that we implemented our open source library system via the **PAC model**. This model arranged our code in 3 specific areas which were the view (presentation), the data(abstraction). and the mediator(controller). The forms for each window for the GUI was contained in the presentation of this model. Each script for

the shapes, connections and file import and exports were the abstraction of the model. And our controllers were the shape controller script and the main application script. With this format of our code, everything was kept in a neat and concise order to manage and maintain without much of a problem.

2. Background and Related Work

Prior solutions such as Microsoft Visio and Creately are great programs for developers to implement their designs but also come with constraints which can make it virtually impossible to enhance or modify their designs. Microsoft Visio is an innovative solution that helps you visualize data-connected business process flows with a host of integrated features. Microsoft Visio is great for creating and co-authoring professional looking diagrams such as flow charts, organizational charts, floor plans and network diagrams[2]. These are effective for decision making, data visualization, and process execution to help increase productivity across the business[2]. Although Microsoft Visio is a very efficient and widely used program it comes with a good amount of limitations. The limitations that are commonly known can have a negative effect on the use of Visio for future developers. One of the most noticeable limitations would be the exclusive software that Microsoft runs on all their programs. Microsoft software always uses a high level of exclusivity which restricts developers for making updates and improvements to their designs. With these restrictions you cannot implement an innovative idea that you or your team develops internally. Another key limitation in Microsoft Visio is the cost of the program. The cost will affect businesses or companies that will eventually find the license fees to be too expensive. These fees can be very tedious and can round up to \$500 per person[2]. This can be very difficult for a developer especially because Microsoft Visio software is proprietary and there also is a learning curve involved which incur additional costs for training. Our framework can provide the same functionality as Microsoft Visio for free, with the option to perform endless customization and creator freedom.

Creately is also a proprietary visual work space. This software focuses on how to plan, brainstorm, analyze and design everything from flowcharts to Unified Modeling Language (UML) diagrams[3]. Creately is much like Microsoft Visio in its exclusiveness. The software restricts the creativity of developers limiting the personal effect of designing software to their specific demands.

Our open source library satisfies the limitations of both Microsoft Visio and Creately. Our framework is free. It provides a software developer the tools needed to complete any visual project they wish to accomplish and is not proprietary software that requires licensing fees. Our framework can create complex diagrams while Creately only allows for simple diagrams. Table 1 provides a breakdown of the limitations of past works compared to the framework we have developed.

Table 2.1: Existing Limitations of Past Works

Software	Costs	Proprietary	Types of Diagrams
Microsoft Visio	\$500	Yes	Complex but 2D Diagrams
Creately	\$15 per month	Yes	Simple Diagrams
Open Source Library	Free	No	Complex Diagrams

Yes = Third party developer cant update software

No = Third party developer can update software

3. Proposed System/Application

3.1 An Overview

The goal of this project is to build an open-source library that will enable developers to create applications that have an interface based around structure drawing. The library will enable developers to quickly and easily create a “canvas” where users can drag, drop, and connect shapes together.

There are major key differences between what has been done. Majorly, the open source aspect of the application, where the library will be built to support open source changes made by outside contributions while other applications have their applications specifically focus on their source code and workings.

The choices made for the full implementation of the application were done with the framework in mind. The idea is to build a framework that performs similarly to those applications on the market in the user level, yet the functionality of the framework also works at the developer level to give developers the ability to use the current built framework to update or create a new framework using the one currently build in our application as a baseline.

We used the SKiaSharp point method to create our shapes. Each point was then connected with a line to simulate the shape we needed to create. We used certain formulas to make sure our shapes looked correct to our standards.

Perimeter of a Rectangle

l - length
 w - width
 P - perimeter

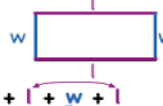

$$P = w + l + w + l$$
$$P = 2w + 2l$$
$$P = 2(w + l)$$

Figure 3.1: Perimeter formula of a rectangle

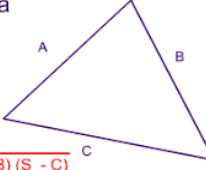
$$(x - h)^2 + (y - k)^2 = r^2$$

Figure 3.2: Circle standard formula

Heron's Formula

$$S = \frac{A + B + C}{2}$$

© mathwarehouse.com

$$\text{Area} = \sqrt{S(S - A)(S - B)(S - C)}$$


The diagram shows a triangle with vertices at the top, bottom-left, and bottom-right. The top side is labeled 'A', the bottom-left side is labeled 'B', and the bottom-right side is labeled 'C'.

Figure 3.3: triangle formula

3.2 System Requirements

There are multiple requirements that had to be met in order for the application to work properly. We have two different types of users who will use this framework and application. There is an end user of the application who will interact with the graphical user interface that is created based on the framework. There is also a software developer who will utilize the open-source framework to design and build an application.

The application that has a graphical user interface to interact with, which holds necessary requirements the application has met. The graphical user interface provides a canvas for drawing, a Toolbar, the ability to drag a shape from the toolbar and around the canvas, a zoom functionality as well as an import and export functionality that converts to a Portable Graphics Format. When the application detects connections, there a “rubber band” effect that occurs which rubber bands the line connected to two shapes. Connection functionality also works with clicks in the application. Left click of the mouse drags connection line while right click attempts to establish connections based off the connection type chosen. If a connection to shape with the correct set connection type is done, the shape will connect and indicate a visual marker that connection is established.

There is also a set of Non-functional requirements that have been met. The application meets Availability and Stability requirements as the framework is open source and the latest version of the C# .NET framework was used. The application meets Efficiency requirements by only processing draw cycles when required. The application was designed to limit the amount of required draw cycles per frame rate. The application meets Reliability, Cost and Platform Capability requirements as the application has been unit tested. The program cost is intended to limit the required amount of processing without needing to meet strict hardware limitation due to the use of C#, which also allows for multiple platforms to have access to the application.

Thus, because the application meets all functional and non-functional requirements, along with being written in C#, it works with all types of operating systems.

3.3 Technical Design

The technical design involved the use of multiple different users having accessibility to the application as a whole. According to Figure 3.4, the User Case diagram displays the front end of the technical design, focusing on the surface usage of the application. The user will have multiple key functions when utilizing the application. The first function is the use of the toolbar. The toolbar is the main component of the User Case. It involves accessibility to multiple features such as shapes, import, export, pointer-connector tools and the connection type drop down menu. The second function is the creation and use of shapes. Using the toolbar, shapes would be created upon a click of the mouse and dragging it onto the canvas. The third function is using the mouse pointer to move shapes to a different location within the canvas. The fourth function is the creation and validation of connections. A visual marker is used to indicate valid or invalid connections with a default visual marker set in place to show that connections can be made. This is further influenced by the fifth function which is the pointer-connector tool. The Pointer tool is used to drag shapes around the canvas while the connector tool sets connections up and creates the visual marker on a connection point.

The internal system flow, as seen in Figure 3.5, is the back end of the application and focuses on ensuring certain events occur that allows the front end to work properly. The first function would be the button events. The buttons play an integral back end role as they are the indicators to either accessing shape classes, import and export functions as well accessing the pointer-connector tool. The second function would be mouse events. Mouse events play a key role on how the user in the front end manipulates the shape by accessing necessary classes that perform the actions. The last function focuses on SkiaSharp control events. SkiaSharp controls allow for the shapes to be created, moved and scaled with assistance of the first functions in the backend as well as input from the front end of the system.

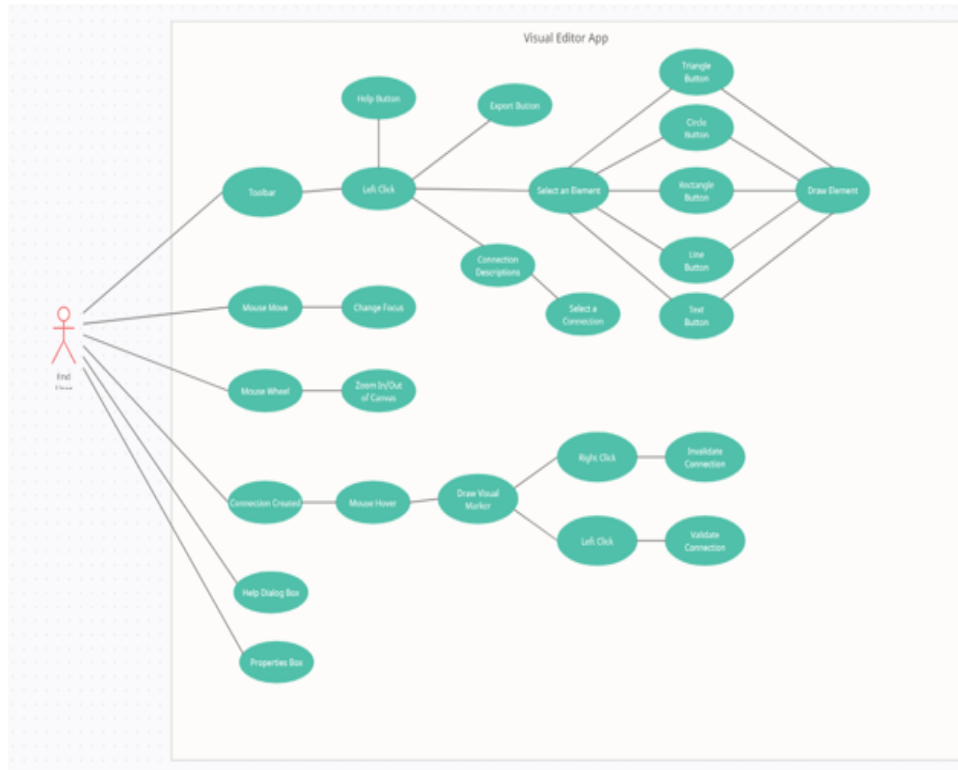


Figure 3.4: Use Case Diagram

3.4 System Implementation

The system implementation focused on using the Visual Studio 2019 integrated development environment to work with the .NET framework and the use of C#. SkiaSharp was used as the graphics tools for creating the application. This was integrated within the window's form tools which helped render the drawing surface for the canvas. The .NET framework was used as it provides the necessary tools and libraries for running on Windows.

3.5 Use of Computer Science Theory and Software Development Fundamentals

When creating this open source library system, we used three main computer science theories, which are inheritance, polymorphism, and encapsulation. Inheritance is used for object oriented programming, and it's purpose is to enable the ability for code to be reused. The code is allowed to have objects interacting with one another to serve many tasks. Properties from a class could be inherited by another class or function when an object is referenced in another class. In our software program, we have shown this by inheriting from the parent classes of shapes and connection classes. The controller class used an arraylist of the base shape class to iterate over each shape type using polymorphism. Polymorphism was also used

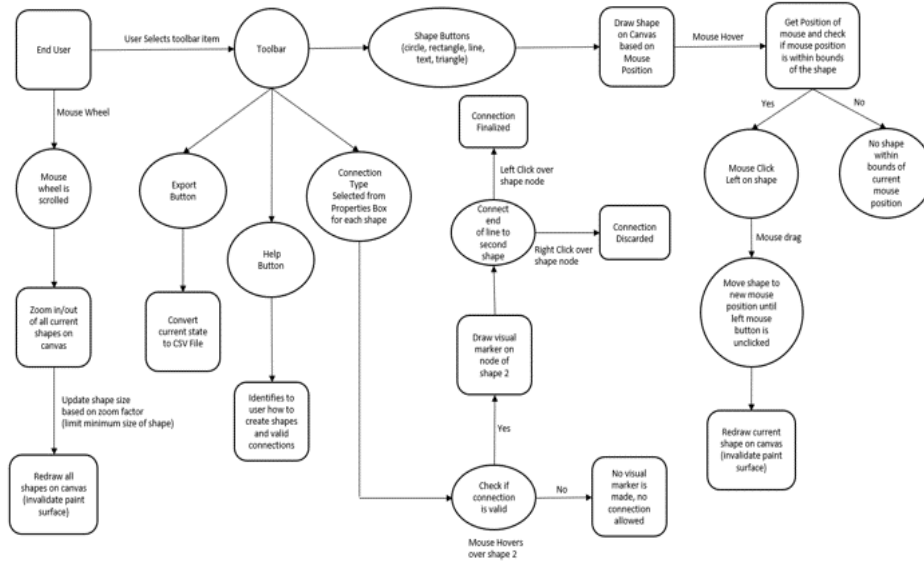


Figure 3.5: System And Data Architecture

within the connection class to create connections of various types of shape classes. We used polymorphism to easily make our code much cleaner and understandable for those wanting to modify our code for their own purposes. This was seen mostly within our shapes classes and controller classes. Encapsulation is a programming style where implementation details are hidden. Our code used this in conjunction with inheritance because of the calling of other classes and methods which were called into separate class as Public, Private, Override, and Virtual.

As for our performance of our software development fundamentals we put most of our focus on having reusable code, a well developed programming design pattern, and continuous unit testing. Since the reuse of our code is one of our integral part of this assignment, we made sure to make each class as module as possible. The programming design pattern we utilized was the PAC model we described in the introduction section of this report. The Unit testing of our application was done every time we introduced a new concept or feature to our application's code. This was to ensure that everything we implemented performed as expected. Our program did suffer some errors such as movement latency due to the rendering cycle of the draw sequence within the design of the background grid. We had to fix the connections made between other shapes and their connection points, while also simplifying the GUI toolbar. We also had to troubleshoot issues with the drag drop feature of shapes from the toolbar to the canvas.

4. Experimental Design and Testing

The Application needed a plan for experiments, results, analysis, and validation for the main system functionalities. There were multiple experiments that were conducted throughout the building of the application to test that the application and see if the requirements for the application were met. All experiments were conducted using the Visual Editor Application.

The objective of each experiment was to demonstrate that the application meets the specified application requirements.

4.1 Experimental Setup

- Experiment #1:

The objective was to test that a user can create the predefined shapes onto the drawing canvas of the application. The user will then move the shape around the canvas using the left mouse button. The user will then zoom in and out of the canvas using the mouse wheel demonstrating the scaling of the specific shapes.

The size of the data will be three to four users depending on accessibility. The data will be collected via the user's input.

- Experiment #2:

The objective was test that a user can make a connection between two shapes and a line shape based on certain criteria. If the criteria are met, the user can make a connection. If the criteria are not met, the connection is invalidated.

Thus, the experiment was done on a user's computer. The requirements to test the experiment was mouse and a way to run source code from their laptop.

It was required that the developers interact with the user to setup valid connections. The developer will setup the connection types within each shape's property box prior to testing.

A table will be used to collect this data from the user. It is expected that one to two users will test all connection types.

- Experiment #3:

This experiment was to test how well the help instructions work within the user interface. The user will use the help dialog box for navigating the application.

The size of the data will be three to four users depending on accessibility. The was data

collected via the user's input on what instructions need to be improved.

- Experiment #4:

This experiment will test the export function. The user will export a screenshot within the user interface to a PNG file. Once the screenshot has been taken, the user will have to check their file folder that was set to export the screenshot to determine if the screenshot was placed in the folder. They will be open the PNG file and validate it against what is currently displayed on the application. This will consist of comparing the applications screenshot capabilities to see if there was a match between what was screen captured and the main application.

They will also need to be able to define the file path for the PNG file. This was a required setup by the software developer.

The size of the data will be three to four users depending on accessibility. The data was collected via the PNG file input.

- Experiment #5:

This experiment will test the import function. The user will import an image file into the canvas of the main application. Once the image file was imported, the user must check to see if this is the right set of data that was imported. This could be done simply by viewing the canvas.

They will also need to be able to use the file path that the image file was stored in. The size of the data will be three to four users depending on accessibility.

Table 4.1: Experiment #1

Test	Input Method	Shape	Valid?	Expected Result	Actual Result
1	Mouse	Circle	Yes	True	True
2	Mouse	Rectangle	Yes	True	True
3	Mouse	Triangle	Yes	True	True
4	Mouse	Diamond	Yes	True	True
5	Mouse	Circle	Yes	True	True

Table 4.2: Experiment #2

Test	Input Method	Shape	Valid?	Expected Result	Actual Result
1	Mouse	Circle	Yes	True	True
2	Mouse	Rectangle	Yes	True	True
3	Mouse	Triangle	Yes	True	True
4	Mouse	Diamond	Yes	True	True
5	Mouse	Circle	Yes	True	True

Table 4.3: Experiment #3

Test	Input Method	Shape	Valid?	Expected Result	Actual Result
1	Mouse	Circle	Yes	True	True
2	Mouse	Rectangle	Yes	True	True
3	Mouse	Triangle	Yes	True	True
4	Mouse	Diamond	Yes	True	True
5	Mouse	Circle	Yes	True	True

Table 4.4: Experiment #4

Test	Input Method	Shape	Valid?	Export	Expected Result	Actual Result
1	Mouse	Circle	Yes	True	True	True
2	Mouse	Rectangle	Yes	True	True	True
3	Mouse	Triangle	Yes	True	True	True
4	Mouse	Diamond	Yes	True	True	True
5	Mouse	Circle	Yes	True	True	True

Table 4.5: Experiment #5

Test	Input Method	Shape	Valid?	Import	Expected Result	Actual Result
1	Mouse	Circle	Yes	True	True	True
2	Mouse	Rectangle	Yes	True	True	True
3	Mouse	Triangle	Yes	True	True	True
4	Mouse	Diamond	Yes	True	True	True
5	Mouse	Circle	Yes	True	True	True

Overall, the experiments were done on a user's computer. The requirements to test the experiment were always a mouse and a way to run source code from a laptop.

4.2 Results and Analysis

There were many expected results and some unexpected result that stuck by the end of the application's completion. Tables were used record each experiment to ensure each experiment when through case thoroughly and properly. While each case is simplistic in nature, not reviewing each small case could lead to major bugs that otherwise could have been avoided.

Thus, the best possible way for a user to be successful using this interface is to have a mouse equipped and learn how to use the properties that go along with the open-source library. It is safe to say this open-source library has a lot of functions that most sources

don't have which makes this application far more unique than others. For example, the export button along with the help button makes things very much easier on the user. Especially the help button because this feature can give its user direction.

Expected Results:

- With the use of a mouse a user can drag and drop selected items then make a connection between two shapes and a line shape based on certain criteria. If the criteria are met, the user can make a connection. If the criteria are not met, the connection is invalidated.
- All shapes can be expanded and minimized.
- The shapes are able to be dragged and dropped into the canvas.
- Export function is able to produce a screen shot of canvas.
- Export function converted into PNG file and placed into a folder.
- Import function takes image file and places it on canvas.

Unexpected Results:

- Import function does not connect shapes dragged into canvas with the image placed into canvas.

4.3 Limitations

There were some present limitations that affected the result of the main application. A limitation that is anticipated is that there is not a multitude of features that can be implemented, which can lead to some lacking confidence from other developers to use our library. This is due to the limited framework that is being built. Another limitation that is anticipated from our library is the smaller framework. The framework is the base needed to operate the main application; however it is small and simplistic.

The application was based off of a requirements for a framework. There could be use cases that we did not initially think of when working on the framework. There is an expectation that there could be gaps between what we used the framework as versus what a different developer would use it as. As such, the developer might require different classes or new methods for implementation. However, the fact that this is an open source library, expands on the fact that this is an evolving project with limitless possibilities.

5. Ethics and Legal Practices

There were legal and ethical practices that were taken into account during the creation of the application. We made sure to not include personal or private data from users when conducting validation tests. SkiaSharp was the drawing library that we implemented which is open source. Our framework follows the MIT license which grants permission to any user free of charge to copy our software.

6. Effort Sharing

INDIVIDUAL CONTRIBUTIONS:

George Tackie:

- Created the main application front end.
- Created the export front end.
- Created the import front end.
- Created the background grid class for the canvas to work with on the GUI.

Ben Twomey:

- Created interface for shape class.
- Created shapes classes.
- Created the shape controller.
- Created the connection/connectionPoint classes.

Justin George:

- Created the Help box front end.
- Created the help box back end.

Matthew Guerrero:

- Worked on properties box
- Helped implement shape controller class
- Worked on the connection class

JOINT CONTRIBUTIONS:

There were also joint contributions between all of the group members. George and Matt worked together on implementing the properties box with the main application. George and Justin worked together on implementing the Help box with the main application to ensure proper functionality. Ben and George worked on drag effect for manipulating shapes from the toolbar to the canvas. Everyone in the team also ran test validations for every piece of the code. (as shown in Table 6.1).

Table 6.1: Effort sharing

Team size	Joint efforts	Ben	George	Justin	Matthew
4	J (30%)	I (17.5%)	I (17.5%)	I (17.5%)	I (17.5%)

J = description of tasks jointly performed

I = description of tasks individually performed

7. Conclusion and Future Work

The project has many internal and external workings that allow for the best possible use from all different types of parties that wish to operate using the application. Visual structured data in the form of graphs can allow those who wish to use the application for business or personal purposes. Developers can choose to work with the internal code, modify and, update the application to build on the strengths and patch up the weaknesses our current application holds.

Future work can involve creating a redo and undo function for when implementing shapes on the canvas. Also, additional shapes should be added to the toolbar, along with different types of lines. Lines could be made dotted, or the ability to add color to the lines. Other features could involve importing images from a database that were saved previously by a user. The user could then pull other user's saved project from the database.

Bibliography

1. Wagner, Bill. (n.d.) *C# Docs - Get Started, Tutorials, Reference*. C# Docs - Get Started, Tutorials, Reference. | Microsoft Docs. Retrieved February 20, 2021 from docs.microsoft.com/en-us/dotnet/csharp/
2. Creately. (n.d.). *Creately*. Retrieved February 20, 2021, from <https://creately.com>
3. Microsoft. (n.d.). *Microsoft Visio*. Retrieved February 20, 2021, from <https://www.microsoft.com/en-us/microsoft-365/visio/>
4. Microsoft. (n.d.). *SkiaSharp Namespace*. Retrieved February 20, 2021, from <https://docs.microsoft.com/en-us/dotnet/api/skiasharp?view=skiasharp-2.80.2>