University of Dayton

Department of Computer Science

CPS 491 - Spring 2019

Dr. Phu Phung

Capstone II Project

HTML5-based Property Sketch Web Application

Team 8 members

1. Caroline Gallo, galloc3@udayton.edu
2. Daniel Illg, illgd1@udayton.edu
3. Michael Carlotti, carlottim1@udayton.edu

Company Mentors

Dwayne Nickels, Architect

Chris Drake, Project Manager

Michael Lange, VP of Development

Tyler Technologies

1 Tyler Way

Moraine OH 45439

Project Management Information

Management board (private access) https://trello.com/b/L9bTqRTW/cps491-scrumteam3

Source code repository (private access) https://bitbucket.org/cps491s19-team8/cps491s19-team8/src/master/

Project homepage (public) https://cps491s19-team8.bitbucket.io

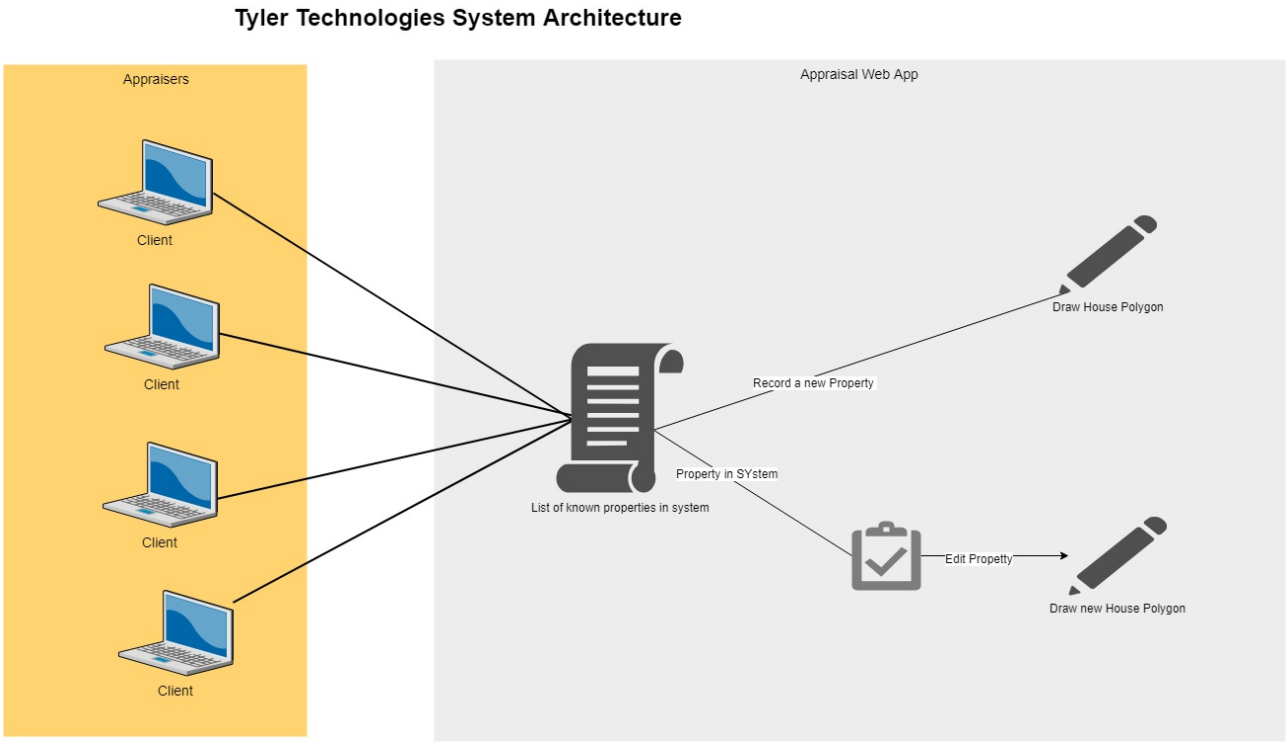
Revision History

Date	Version	Description
1/28/19	0.0	Initial draft
2/5/19	0.1	Added hours and revision history
2/12/19	1.0	Added Sprint 1
3/5/19	2.0	Added Sprint 2
4/2/19	3.0	Added Sprint 3
4/26/19	4.0	Added Sprint 4

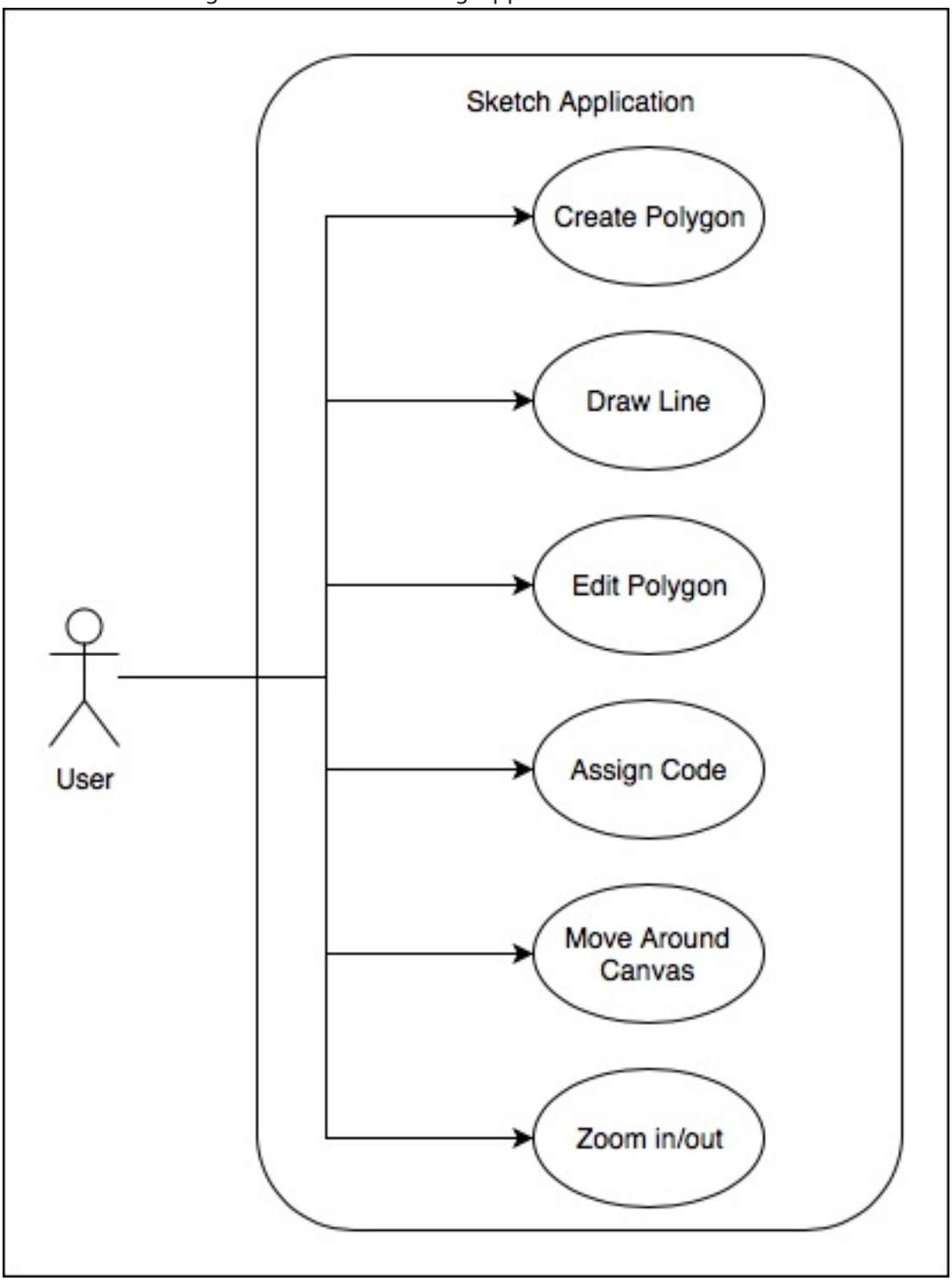
Implementation

Sprint0 - Design Stage

The overall architecture of our project:



The Use Case Diagram for the Sketching Application:



Use Case Descriptions:

Use case name:	Create Polygon	
Scenario:	User creates a polygon	
Triggering event:	User chooses a destination and then creates a new polygon there.	
Brief description:	When a user wants to create a new polygon, they will select a location and then create a new polygon there, probably through a pulldown menu or a pushed key.	
Actors:	User	
Related use cases:	Draw Line, Assign Code	
Stakeholders:	User	
Preconditions:	The user is in the sketching application.	
Postconditions:	A new polygon has been created that the user can draw sides on. The polygon will accept codes users might assign to it.	
Flow of activities:	Actor	System
	1. User clicks on a location and selects 'create polygon'	1.1 The system creates a new polygon and stores its starting location.
	2. User draws the first line of the polygon	2.1 The system stores the end point of the drawn line.
Exception conditions:	2.1 The endpoint of the line is invalid(off the screen)	

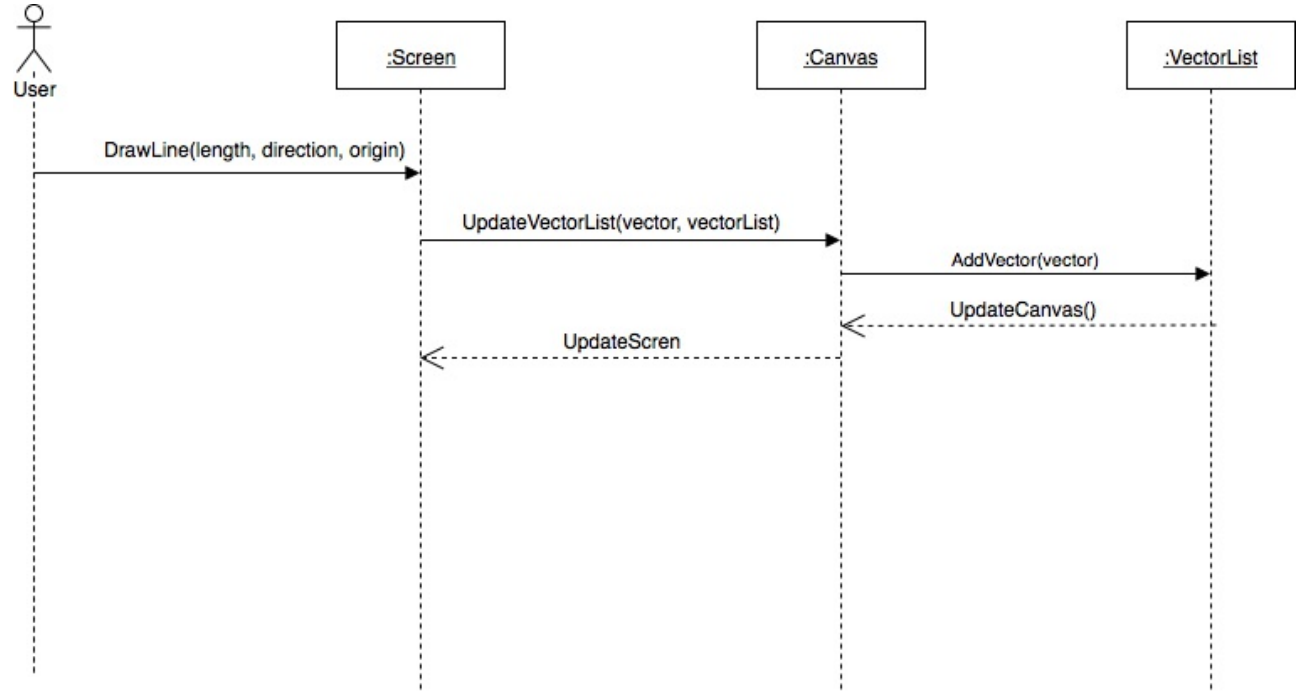
Use case name:	Edit an existing polygon	
Scenario:	User wants to edit an existing polygon	
Triggering event:	User wants to change the dimensions and placement of an existing polygon	
Brief description:	After the creation of the polygon, the user can change where the polygon is on the page and the size of the polygon by dragging it or changing the values.	
Actors:	User	
Related use cases:	User creates new polygons	
Stakeholders:	Architects	
Preconditions:	User must create a valid polygon	
Postconditions:	Polygon's dimensions are updated to new values Polygon's placement is updated to new values	
Flow of activities:	Actor	System
	1. User creates polygon	1.1 System creates and displays a new polygon
	2. User changes dimension values or placement values by dragging the polygon or changing the values	2.1 System validates and updates dimension values 2.2 System validates and updates placement values 2.3 System displays updated polygon
Exception conditions:	2.1 User enters invalid dimension values 2.2 User enters invalid placement value	

Use case name:	Move Around Canvas	
Scenario:	Move around the canvas	
Triggering event:	User wants to change their view by moving around the canvas	
Brief description:	User will click and drag to pan around the canvas. This will allow the user to focus on different areas.	
Actors:	User	
Related use cases:	Zoom in/out	
Stakeholders:	User	
Preconditions:	All the polygons are not already on the screen. Likely, the user will be "zoomed in."	
Postconditions:	The area of the canvas that is visible on the screen has changed. All polygons have been redrawn in the right place with the right proportions.	
Flow of activities:	Actor	System
	1. User clicks and drags the mouse in a desired direction	1.1 The variables holding the display area is modified within the system. 1.2 Polygons are redrawn on the screen to match their new locations.
	2. User releases the mouse	2.1 The system stops updating the variables holding the display area. 2.2 Polygons are redrawn
Exception conditions:	1.1 The edge of the canvas has been reached	

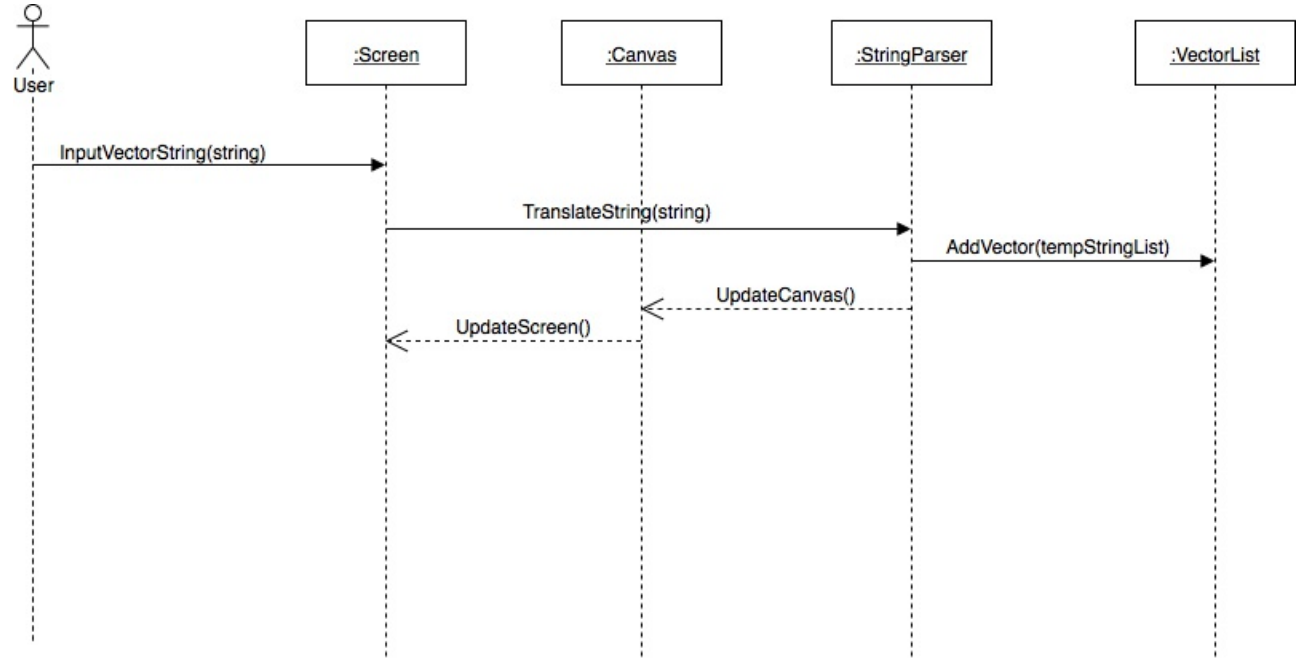
Use Case Name	Zoom In and Out	
Scenario	User wants too zoom in to look at a specific part of the property and zoom out once done	
Trigger Event	User accesses a property on file	
Brief Description	The user has started to draw or update a property and needs to zoom in in order to properly draw the picture	
Actors	User	
Related Use Cases	Pan around canvas	
StakeHolders	User	
Precondtions	The property must exist, the user must have access to the property file	
Post-conditions	The user's screen should display only a portion of the drawing that user has decided to zoom in or out on	
Flow of Activity	Actor	System
	1.1 User opens sketch application 2.1 User chooses a property to update or draws a new property 3.1 User zooms in or out on canvas	1.2 System loads list of available properties 2.2 System either create new property file or display existing file contents 3.2 System updates the screen to that specific portion of sketch is displayed on the whole screen
Exceptions	3.1 User tries to zoom in or out past max distance	

Use case name:	Add a code	
Scenario:	User adds a code for a polygon	
Triggering event:	User wants to use a code to specify a polygon	
Brief description:	User may add only one code to a polygon, but it is not required to add a code for all polygons.	
Actors:	User	
Related use cases:	Create a polygon	
Stakeholders:	Architects	
Preconditions:	User must have created a polygon	
Postconditions:	Code is assigned to polygon	
Flow of activities:	Actor	System
	1. User creates a polygon 2. User assigns a code to one polygon	1.1 System displays polygon 2.1 System assigns the code to the specified polygon
Exception conditions:	1.1 User creates invalid polygon 2.1 Polygon already has a code	

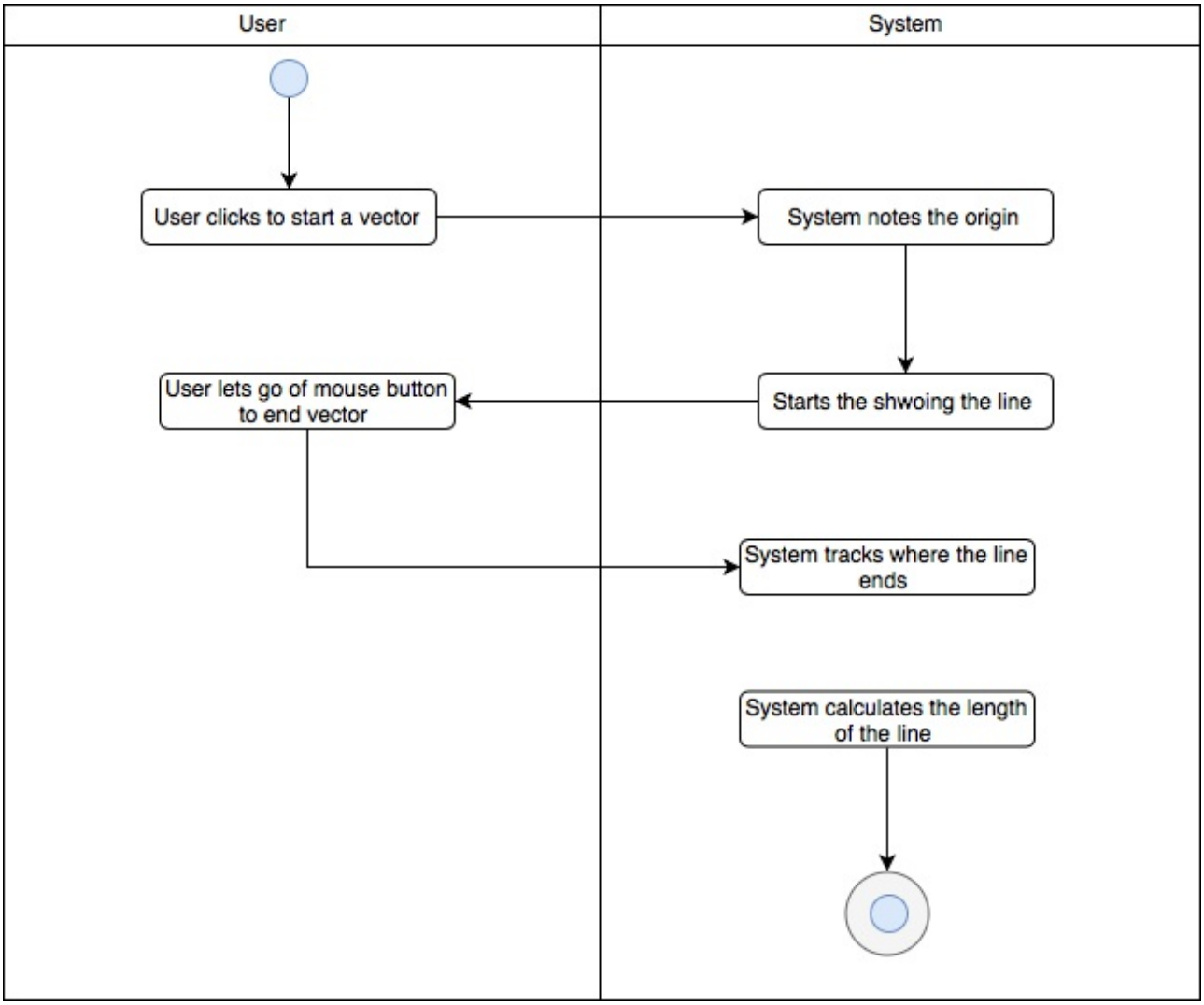
Sprint 1:
Sequence Diagrams:
Draw on Canvas



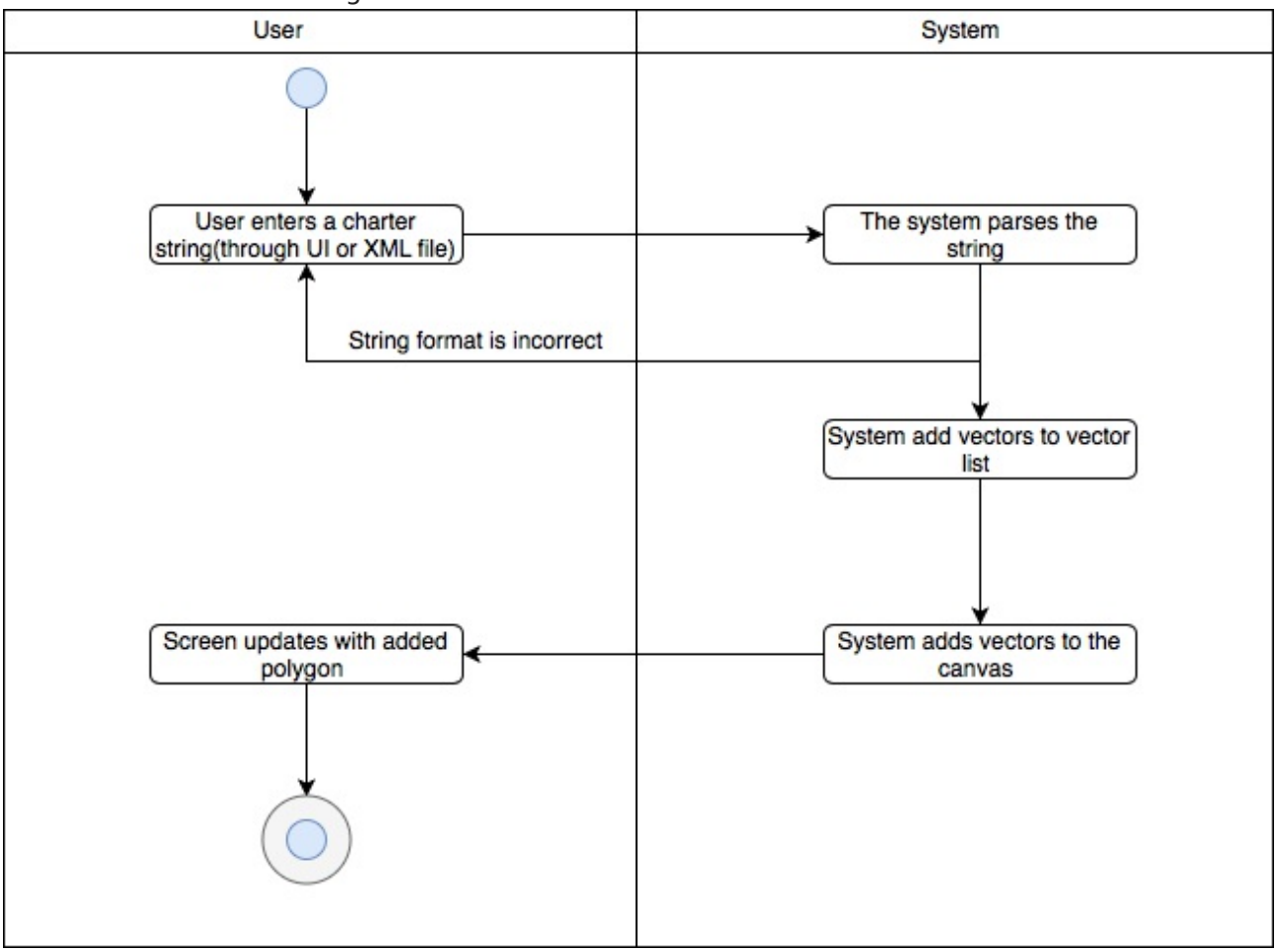
Draw with Character String



Activity Diagrams:
Draw on Canvas



Draw with Character String



Important Algorithms and functions:

drawCanvas:
- This function is used to draw everything on the canvas. It does this by calling several other functions that handle the individual components of what needs to be drawn.

drawGrid:
- This function is called by drawCanvas. It draws horizontal and vertical lines on the canvas in a gray color help the user draw their own lines. The drawn grid lines are scaled to the zoom amount.

drawPolygons:
- This function is called by drawCanvas. It iterates through the array of polygons and draws the lines within them. These lines are stored as a set of points. A polygon will always have at least two points as the user is required to draw a line when the polygon is created.

drawCurrentLine:
- This function is called by drawCanvas if the user is currently drawing a line. The polygon the user is interacting with is stored in a variable. This function takes the last point of that polygon and the users mouse position and calculates where the end of the line should be. Currently, all lines are 90 degree angles so the function calculates the revised mouse coordinates. If the end of the line is close enough to the beginning of the current polygon, it will connect. This function is also used for drawing curved lines.

displayLineLengths:
- This function is called by drawCanvas. It goes through a seperate array of line lengths and displays them next to the corresponding lines. In future iterations of this product, these lengths will be editable and more readable.

clickWithinPolygon:
- This function asks the question if the coordinates of a mouse click are within a polygon.
The function iterates through every polygon in the array polygons.
If the polygon isn't finished, the function will auto-finish it temporarily.
The function then iterates through the points and determines if the mouse coordinates have lines on all four sides.
If in all four directions there is a line in the polygon, the function determines that this is the polygon that needs to be moved.
It then returns 1 to the calling function and sets currentpolygon to the id of that polygon.

movePolygon:
- For a given polygon, movePolygon will update the points in each polygon by change in the x and y directions of the mouse.
The function will stop the polygon from being moved off screen.

translateVectorToDrawing:
- This function takes a character vector(string) as input and draws it on the canvas.
The vector must be in proper format in order to be accepted.
The vector must start with 'm', which is the command for move.
This command signals that the next commands given will be for moving the starting point of the polygon to a desired location.
Later, we might also add a feature to just specifically give coordinates.
Now, read the next few commands. The commands are: 'l', 'r', 'u', and 'd' followed by a number indicate moving the distance indicated by the number left, right, up, or down.
The command 'a' is used to signal that the next two move commands to be given are a pair and should be drawn as such so that the line is not at a 90 degree angle.
The command 's' signals to the system that the starting point has been reached and every point from now on should be drawn.
If the function reaches the end of the vector without any errors, it will add this polygon to the collection of polygons and draw it.

translateDrawingToVector:
- This function iterates through all polygons being displayed on screen and turns them into character vectors.
To start, it creates a new string with the value 'm' at the front for move.
It then adds 'r' with the distance from 0 to the x value of the polygon's starting place signaling how far to move right before drawing.
It then adds 'l' with the distance from 0 to the y value of the polygon's starting place signaling how far down to move before drawing.
The starting point will now be placed at this location. Add the letter 's' to signal that the next points are to be drawn.
Now the function iterates through all the points and checks the direction of the next point.
If the next point moves left or right without moving up or down, then add 'l' or 'r' with the corresponding change in the x axis.
If the next point moves up or down without moving left or right, then add 'u' or 'p' with the corresponding change in the y axis.
If the next point changes in both the x and y axis, then put the letter 'a' in for angled line.

Now compute the change in the x and y axis seperately and add both of them after the 'a' with their corresponding letters.
The polygon has now been translated into a character vector.

undo:
- The undo feature uses a global stack in which ever new line that is drawn is pushed onto it. The stack holds the polygon for which the event happened and the event itself.
When a line is drawn, the event is set to 0 and the id of the polygon for which the line was drawn onto is recorded.
When a user presses the 'u' key, the undo action checks the event.
If the event is a drawing action, the function will pop the event off the stack and pop a value off the recorded polygon.
If the polygon only had two points left before the action, the polygon is completely deleted.

auto-complete:
- When a user presses the 'a' key, this code checks the undo stack to see what the last recorded polygon was.
The code then checks to see if the ending point of this polygon is also the starting point.
If it isn't, the code copies the starting point onto the end of the stack and the polygon is now completed.

loadFile:
- This function takes the file the user has selected, reads in the text using a FileReader.
Then it sets a timeout function to translate the results after half a second since the readAsText function runs asynchronously.

editLine:
- If a user clicks on a line in edit mode, then two boxes appear on the right side of the screen with the x and y lengths of the line.
The user can update these lengths and press the "edit" button to then change the line lengths manually.

findClick:
- This function iterates through all the polygons to see if the user clicked on any point or line. First, the function checks if the mosue coordinates are near any vertices. If they are, the vertex turns blue and the
user can move it around the canvas. If not, it checks if the mouse is on a line. The way it does this is by
checking to see if the mouse is between the x coordinates of the points on either ends of line. If the x coordinates of
these two points are close enough together, it checks to see if the y coordinate of the mouse is between the y coordinates of the two points
and also if the x coordinate of the mouse is the same as the x coordinates of the two points on the line.
If the x coordinates are not equal, then the function essential calculates the slope of the line and determines
if the mouse coordinates are on this line. An extra variable called buffer is introduced to help make sure a click near/on the line
is counted. The buffer is calculated by dividing 1000 by the total distance of the line, as shorter lines will have less precise calculations for
the approximate location the mouse y coordinate should be to correspond with the line slope. If

the click is on either a line or a point,
the function will return a struct containing the polygon number, the starting point, and whether it is a line or point to be edited.

calculatePerimeter:
- This function is a function used to calculate the perimeters of the polygons drawn on the canvas.
The function works in paralell with the displayLineLengths function. The function displayLineLengths calculates the length of each line of all the polygons
and as it does this these values are stored in an array called side_lengths. After the function displayLineLength finishes displaying a line, the function
calculatePerimeter is called with the array side_length as it's arguement. In the function itself the line lengths are added up and then returned the function amount.
This amount is stored in an array called perimeter. Then we clear the values in side_length so that it can be called to store the values of the next polygon.
All polygons who are drawn on the canvas have their perimeters stored in this array. This function doesn't have undo functionality built into it, but in the undo
we have call to remove the top value in the stack for the side_length array and when the side_length hits zero then the top value on the perimeter array is removed from the stack.

zoomIn:
- This function iterates through all the points adjusts them according to the new scalar value. Scalar is a global variable that all point coordinates are divided by.
When zooming in, the scalar value is increased. When zooming out, the scalar value is decreased.

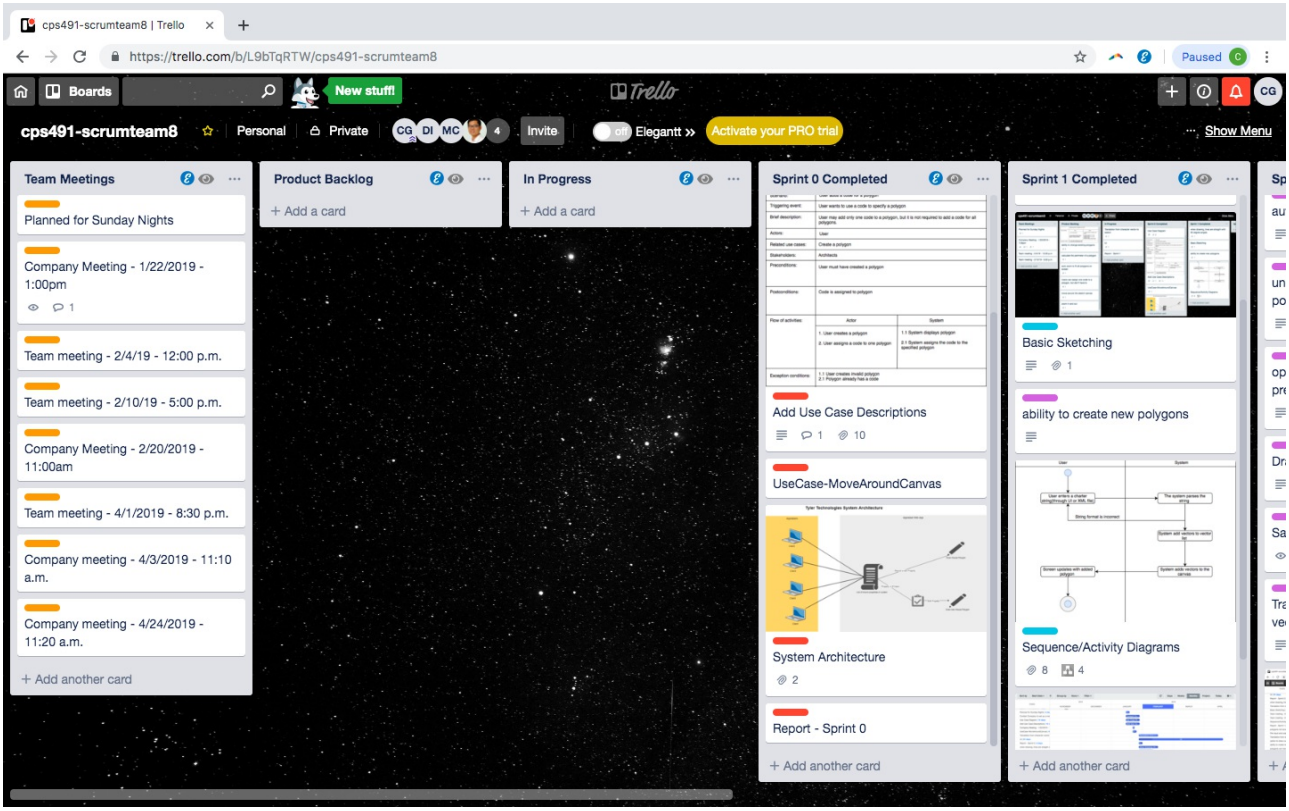
Pan around canvas:
- When a user clicks and drags on any part of the canvas or a part that is not within a polygon when in edit mode, the values of a struct called offset will be modified.
When a point is place on the canvas, its coordinates will be its position on the canvas minus the offset values.
When a point is drawn, it will be drawn at its value plus the current offset.
By adding the offset anytime the system needs to interact with a point, the user can move around the canvas and still be able to interact with polygons.
CalculatePerimterNew:
- This is new version of the calculate perimeter function that removes some the bug and problems that the old calculate perimeter had, and makes the operation of calculating perimeter a
much more efficient process. The calculatePerimeterNew function takes two inputs, the 2d array called polygons and the perimetter array. It is called in the building of polygon datatable.
The function runs through a 2d array that has all the points that make up the polygon. In the function there is a double for loop that first runs through the polygons in the 2d array.
The next for loop then runs through the actual points on the loop. It takes the first point and the last, and calculates the distnace between those two points using our scalar value
to make sure the points show in units not pixels. Then it increments doing the distance between the first and second point, and does the same thing again until it has run through all the points stored in the second part of polygons 2d array. This value is then returned and stored in a local variable called temp_perimeter. Temp perimeter is set equal the corresponding polygon number. After all the polygon's have had their perimeters calculated the function returns the filled our perimeter array.
CalculateArea:
- This function takes two inputs the polygons 2d array and the second is the array for storing

areas. This function is called when adding the values to the datatable so that information is up to date when the user wants to see the polygon data assigned so far. The function uses a double for loop to calculate the area of each polygon. The first for loop runs through the different polygons stored in the polygons 2d array. The next for loop runs through the points stored that make up each polygon. There is an algortihm used for calculating the area of irregular polygons that we used in order to calculate the values of what are sometimes very complicated polygons. The value is calculated by going either clockwise or counter-clockwise through the points of each polygon and scaling their values so that the number used are in units rather than pixels. After each point another value is added to a local variable to store the temp area number that is being calculated. After all the points have been put through the equation then the number we have is halfed and stored into the area array. The temp area variable is cleared and the function goes onto the next polygon stored in the 2d polygons array until we have run through all the current polygons. The newly filled out array of areas is returned and set equal to our areas array.

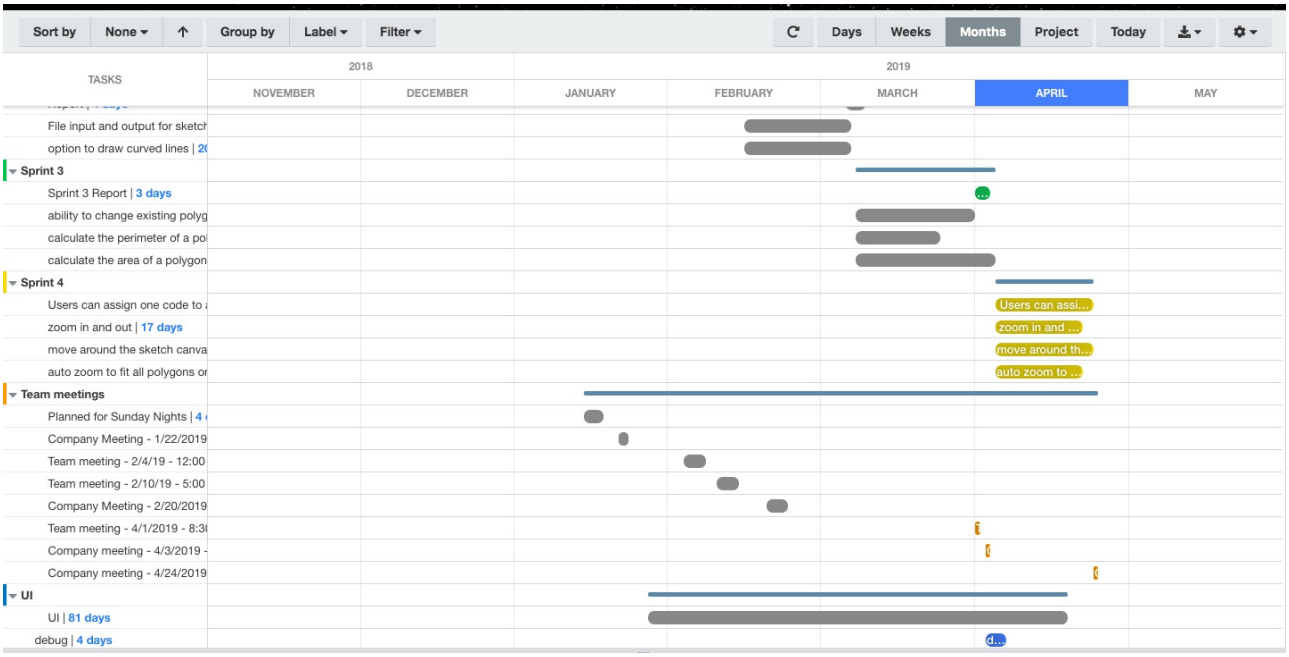
Deployment

Software Process Management

Trello Board:



Gantt Chart:



Scrum Process

Sprint 0

Duration: 1/15/2019-1/28/2019

Completed Tasks

- Contacted Tyler Technologies to get necessary information
- Updated our Trello board in preperation of Sprint 1
- Completed multiple Use Case Descriptions
- Complete a Use Case Diagram
- Set up a project website through bitbucket

Contributions

Daniel Illg, 5 hours, contributed:
- Use Case Descriptions (Create Polygon, Move Around Canvas)
- Use Case Diagram
- Edited report online
- Contacted Tyler Technologies

Caroline Gallo, 5 hours, contributed:
- Use Case Descriptions (Edit Polygon, Add Code)
- Updated gantt chart
- Changed due dates using Elegantt extension

Michael Carlotti, 4 hours, contributed:
- Use Case Descriptions (Draw a Line, Zoom In/Out

Sprint Retrospection

This sprint was mainly used as a planning sprint in order to figure out how we want to approach the other sprints. We assigned our due dates to our sprints, and what we plan on accomplishing in each sprint. We created use case descriptions to have a clear understanding of the processes. We accomplished what we wanted to do for this sprint and did not run into any issues.

Sprint 1

Duration: 1/29/2019-2/13/2019

Completed Tasks

- Basic sketching
- Ability to create new polygons
- Lines are straight with 90 degree angles
- Basic UI
- Sequence/Activity Diagrams

Contributions

Daniel Illg, 10 hours, contributed:
- Basic sketching
- Create polygon
- Drawing features for line lengths, polygons, and a basic grid
- Started working on manually changing line lengths

Caroline Gallo, 8 hours, contributed:
- Basic UI including toolbar and buttons
- Created record table (still needs more testing)
- Researched ways to do file input/output (ended up pushing this to later sprint)
- Edited report
- Updated gantt chart

Michael Carlotti, 7 hours, contributed:
- Sequence Diagrams (Draw on Canvas, Draw with strings)
- Actvity Diagrams (Draw on Canvas, Draw with strings)
- Work on functions to turn a string into a vector(still incomplete)
- Worked on ways to convert strings into vector(still incomplete should be finished early sprint 2)
- Researched ways to turn strings into vectors, JSOn file parser for file I/O, possible table implementation of vectors on screen

Sprint Retrospection

Sprint 1
- Could have finished more of the implementation if we had spent more time working during the first half of the sprint. This is because we would have had more time to test different functionalities.
- Communication issues in the first half of the sprint.
- Could be better about pushing our code regularly

Sprint 2

Duration: 2/14/2019-3/6/2019

Completed Tasks

- Character vector to drawing
- Drawing to character vector
- Simple undo feature to undo lines drawn
- Ability to draw angled lines
- Ability to click and drag polygons around the screen
- Auto-complete feature

Contributions

Daniel Illg, 11 hours, contributed:
- Implemented character vector to drawing
- Implemented drawing to character vector
- Implemented a simple undo feature to undo lines drawn
- Implemented the ability to draw angled lines
- Implemented the ability to click and drag polygons around the screen
- Implemented the auto-complete feature

Caroline Gallo, 9 hours, contributed:
- Research for writing to a JSON file
- Research on using XML files and JSON files
- Implemented the ability to save the canvas as an image to the user's computer
- Implemented the ability to upload a file from the user's computer
- Implemented the ability to draw a curved line/bow

Michael Carlotti, 12 hours, contributed:
- Research for JSON parsing
- Implementation for JSONParsing
- Research for XML parsing/File storage/File Input
- Creating JSON and Test Data
- File Input Using JSON

Sprint Retrospection

Sprint 2
-In terms of reading a file using JSON it appears that it isn't possible without using an external api. Also in order to read from a JSON file without using a server the only way to access the file is
through a js file with the json data hardcoded into it. This isn't quite what we're looking for in this project need to be more dynamic wth the file, i.e input and out. We should be able to update the vector
after accessing the file. Next sprint we're gonna need to to look into options of file using xml or the local storage in a web browser using ztringify() to store vector information, access said information,
and then update/save when necesary.
- Update code more frequently.
- Research ahead of time so when implementing concepts there isn't as many errors.
- Be more clear ahead of time on which task each person is working on so multiple people aren't working on the same task.

Sprint 3

Duration: 3/7/2019-4/3/2019

Completed Tasks

- Ability to change existing polygons
- File input and output for sketch application
- Users can assign one code to a polygon, but don't have to
- Calculate the perimeter of a polygon
- Option to draw curved lines

Contributions

Daniel Illg, 10 hours, contributed:
- Edit vertices by dragging them around the screen
- Edit lines by dragging them around the screen
- Edit lines by clicking on them and then updating their x and y dimensions
- Edited lines/vertices turn blue
- Click on polygon to assign a code to it
- Upload and read files from a computer
- Redid the character vector to drawing function to work with specified commands
- Reworked the drawing to vector function to use the new format
- Added in the F command to vectors to auto-finish drawings
- Added in the Pn commands, where n is an integer between 0 and 9
- Added in the An commands, where n is an integer
- Added in the X command, which makes a rectangle by given dimensions

Caroline Gallo, 8 hours, contributed:
- Implemented curves being saved in an array so they display correctly
- Implemented option to do horizontal or vertical curves
- Implemented equation for perimeter of curves
- Report
- Updated homepage to include project information and link to file with code
- PowerPoint slides

Michael Carlotti, 10 hours, contributed:
- Implemented calculatePerimeter function
- Started to work on the calculateArea function
- Researched into calculating the area of irregular function
- Made skeleton code for calculating area of an irregular polygon

Sprint Retrospection

Sprint 3
- Found out what we already had implemented was what the company wanted for file input/output with the alert and input text box for the vector strings. Next sprint, we will work on implementing it into the record table.
- Need to continue updating UI and could spruce up homepage if time allows

Sprint 4

Duration: 4/3/2019-4/26/2019

Completed Tasks

- Calculate area of a polygon
- Pan around canvas
- Zoom in/out
- Drawing bows with character vectors
- Auto-zoom/pan to fit all points on the canvas
- UI updates
- Record table to keep track of polygon codes, areas, etc.
- Bug fixes

Contributions

Daniel Illg, 10 hours, contributed:
- Pan around canvas
- Zoom in/out
- Auto-zoom/pan to fit all points on the canvas
- Put description and demo on website
- Bug fixes

Caroline Gallo, 10 hours, contributed:
- Implemented table functionality
- Implemented bootstrap to homepage
- Implemented bootstrap to demo page
- Updated UI
- Organized code so that's uniform throughout

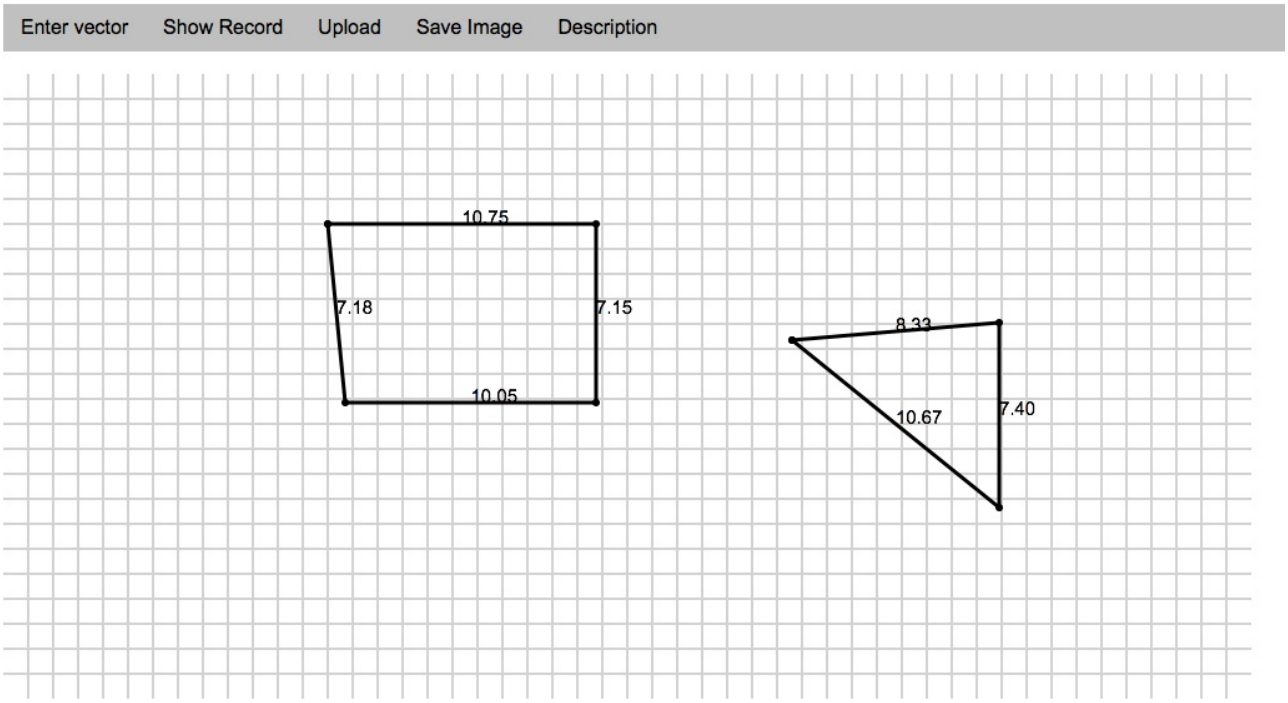
Michael Carlotti, 8 hours, contributed:
- Rsearched area functions
- Made a area function
- Redid Perimeter function
- Bug Fixes

Sprint Retrospection

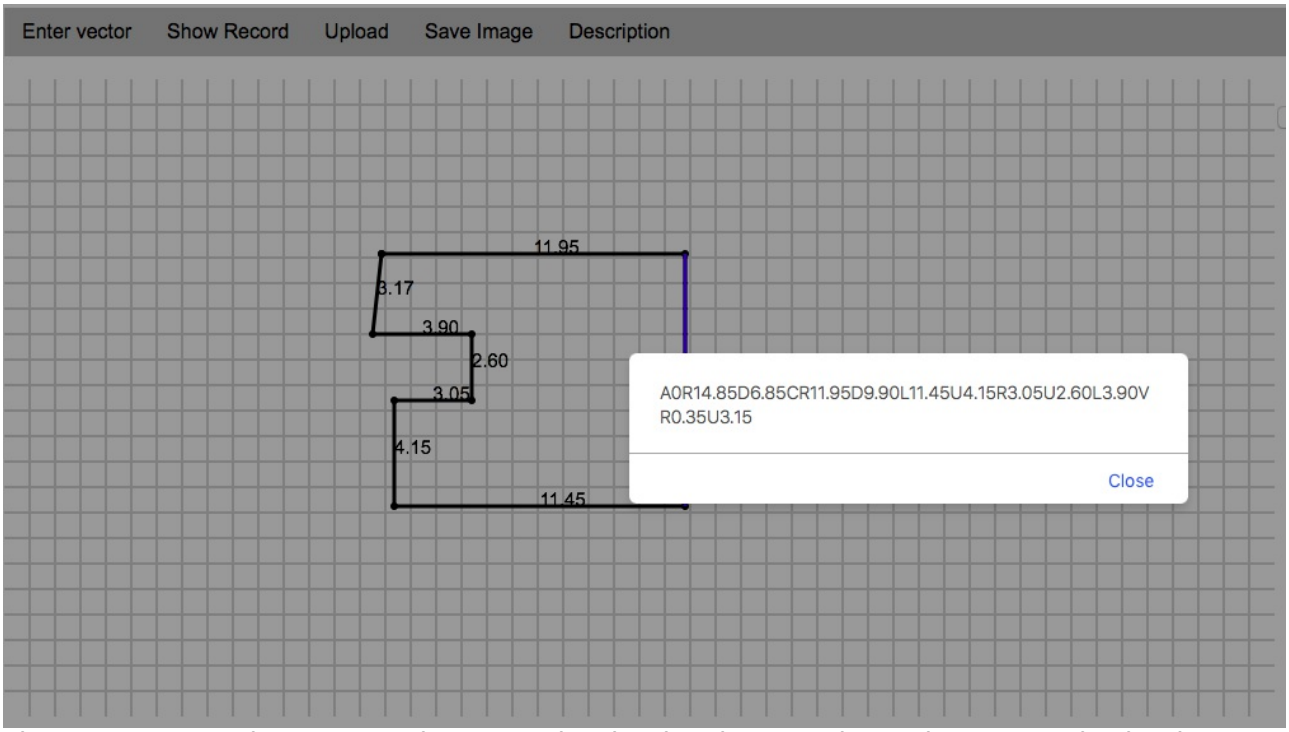
Sprint 4
- We didn't have much left for this sprint so we were able to accomplish everything we originally set out to do.
- We were also able to implement a few of the optional features.

User guide/Demo

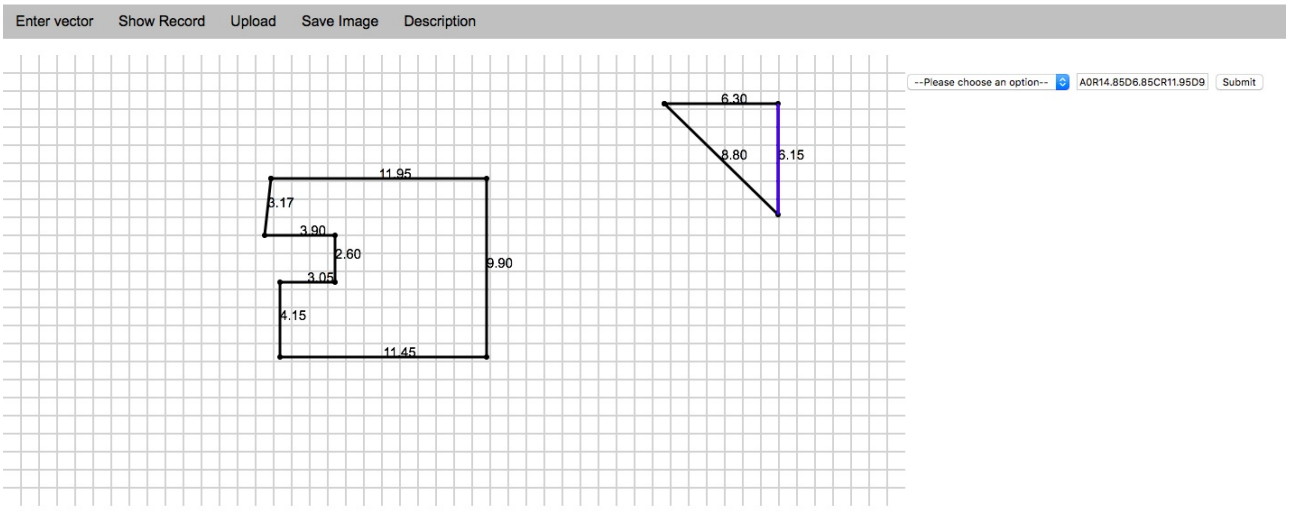
Multiple shapes can be drawn and displayed on the canvas. The user can begin to draw a polygon by pressing any non-special key, and the line lengths are displayed.



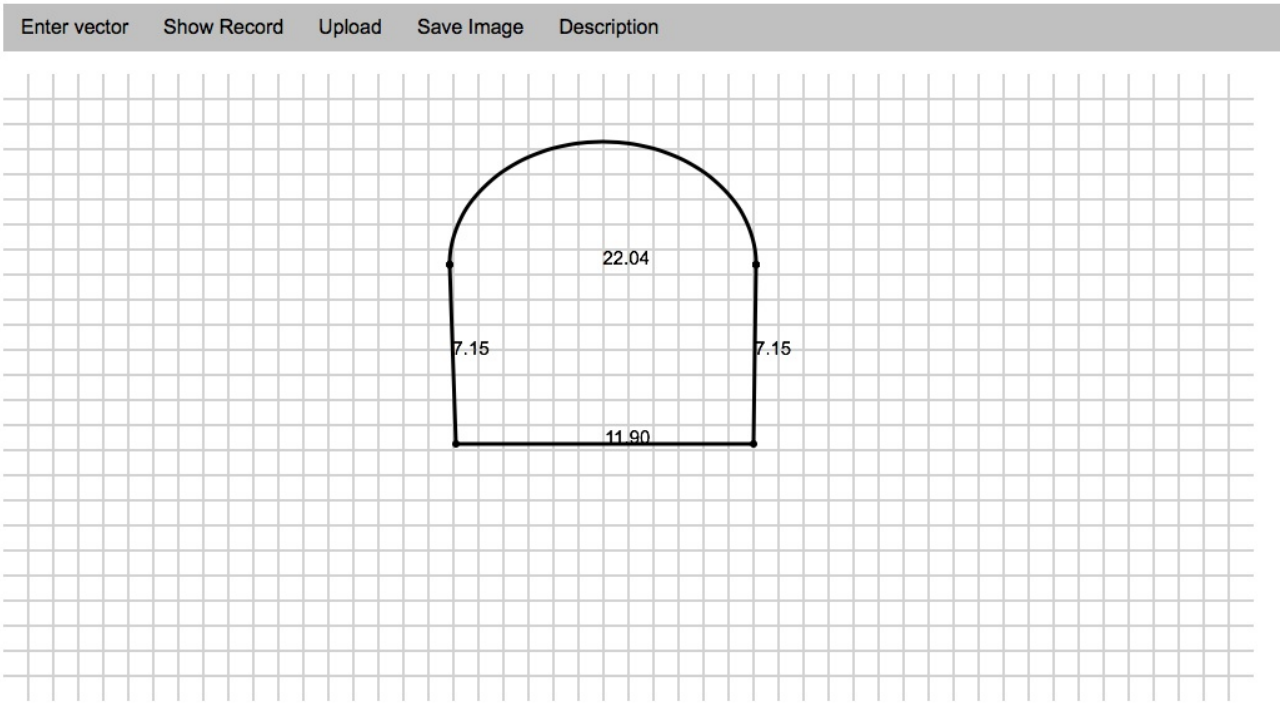
The user can receive the file output by pressing 's' and the vector string will be alerted.



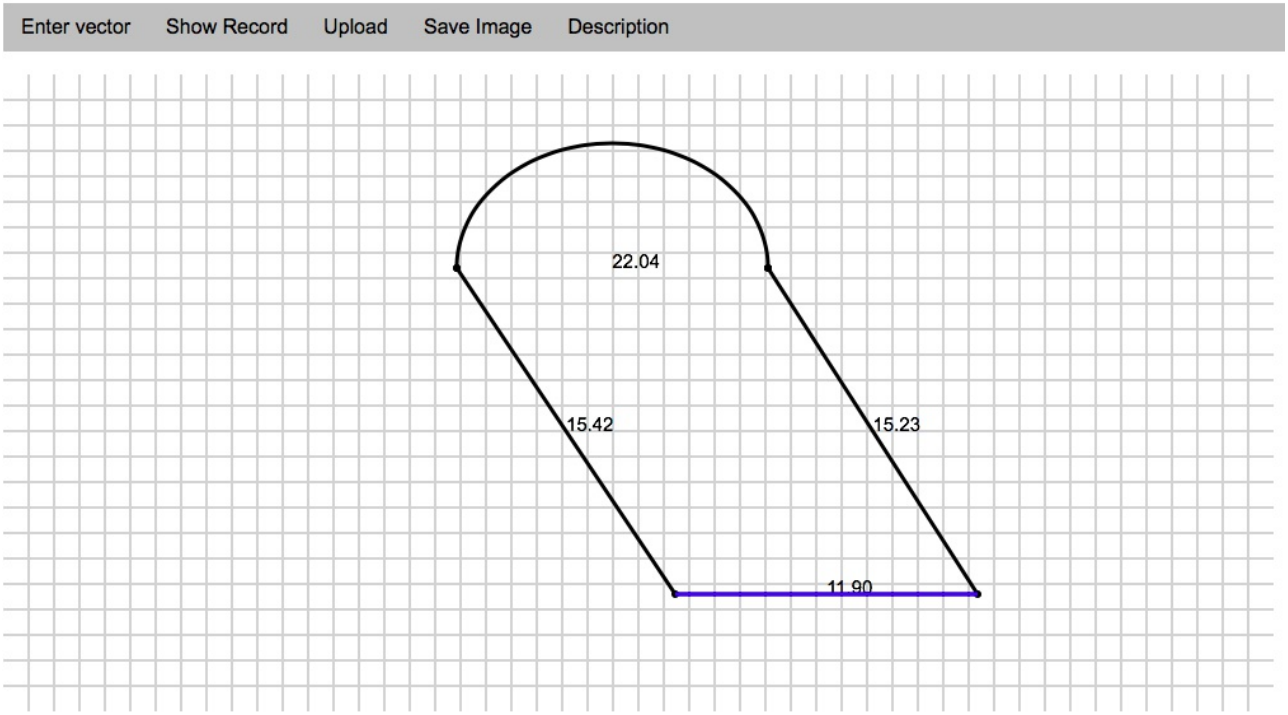
The user can use the vector string to render the drawing. Another polygon can also be drawn on the canvas.



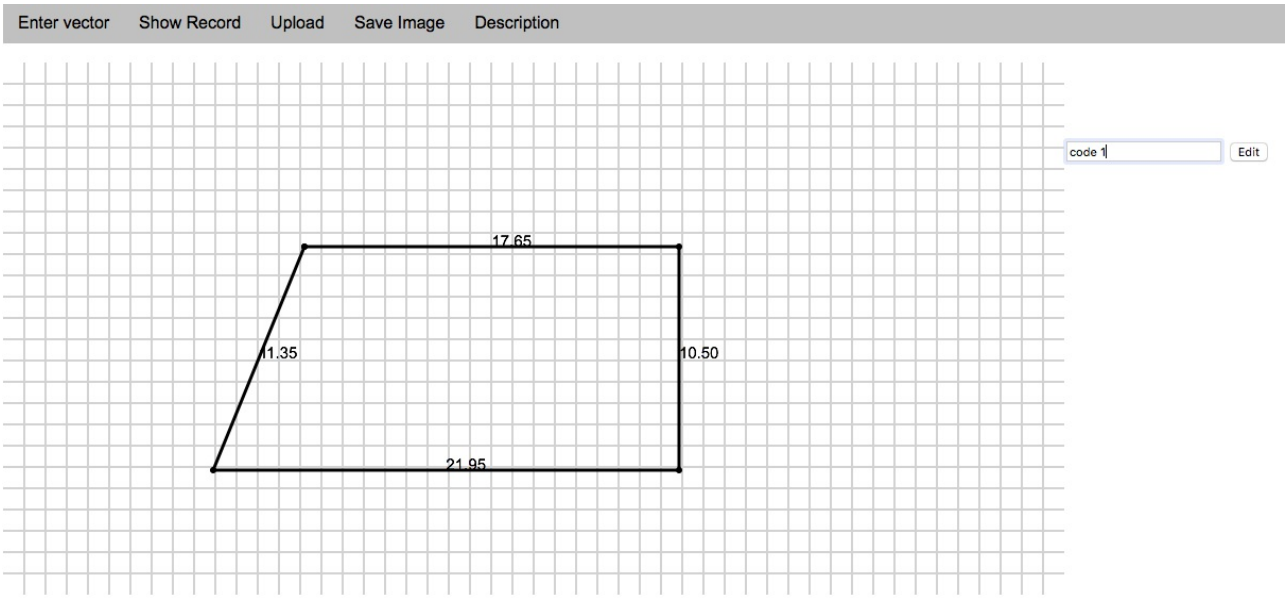
The user can draw curve lines by pressing 'b' to display the options of horizontal and vertical curves. The user must also turn on freedraw by pressing 'f' before the curve can be drawn.



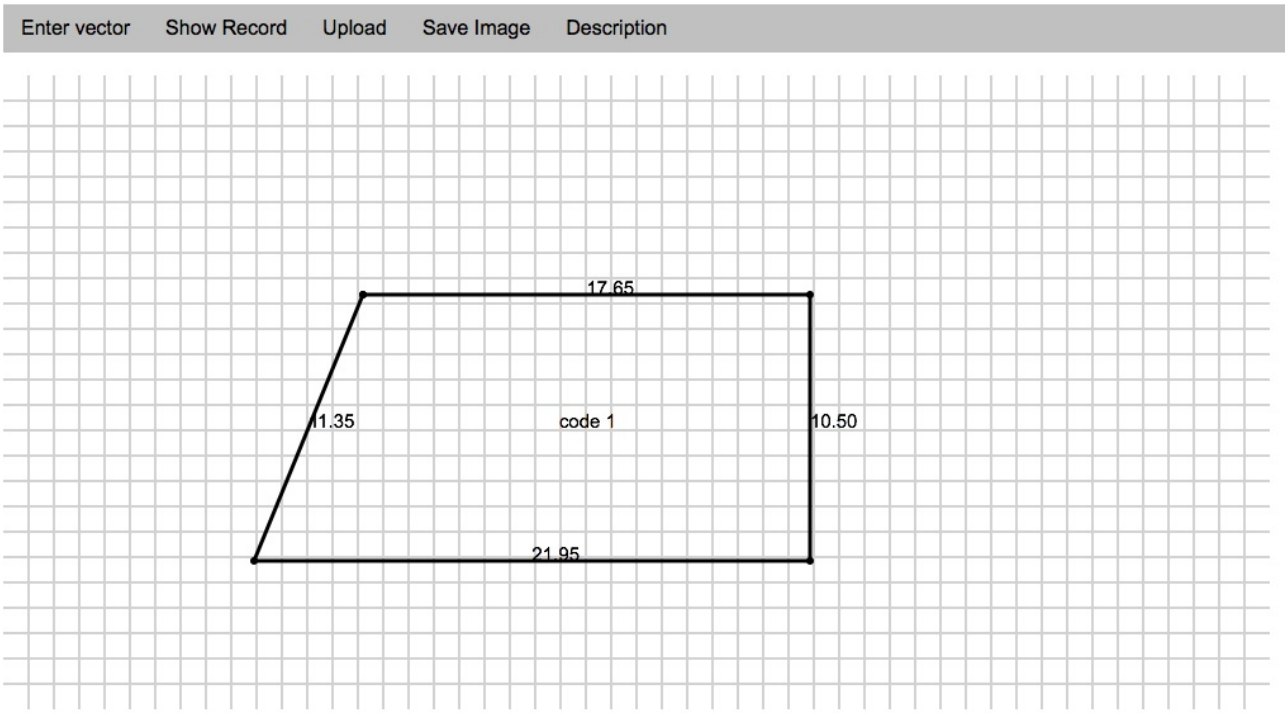
The user can edit a polygon by pressing 'e'. They may edit the polygon by either dragging the line or the dot at the end of the line.



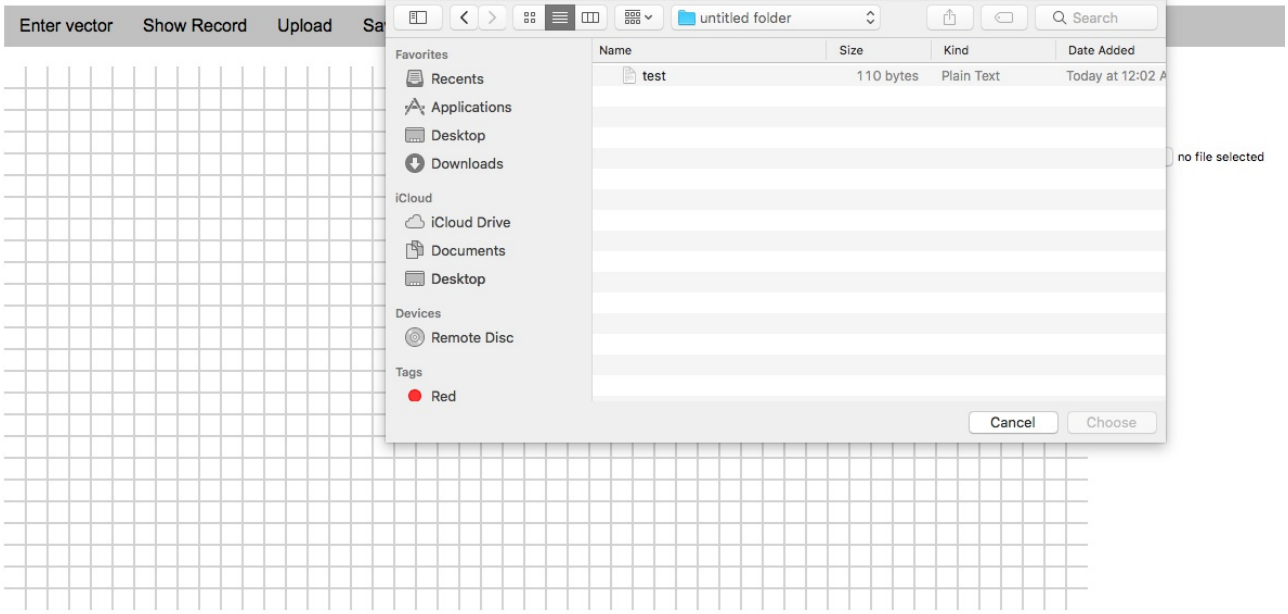
A code can be added to a polygon while edit mode is on.



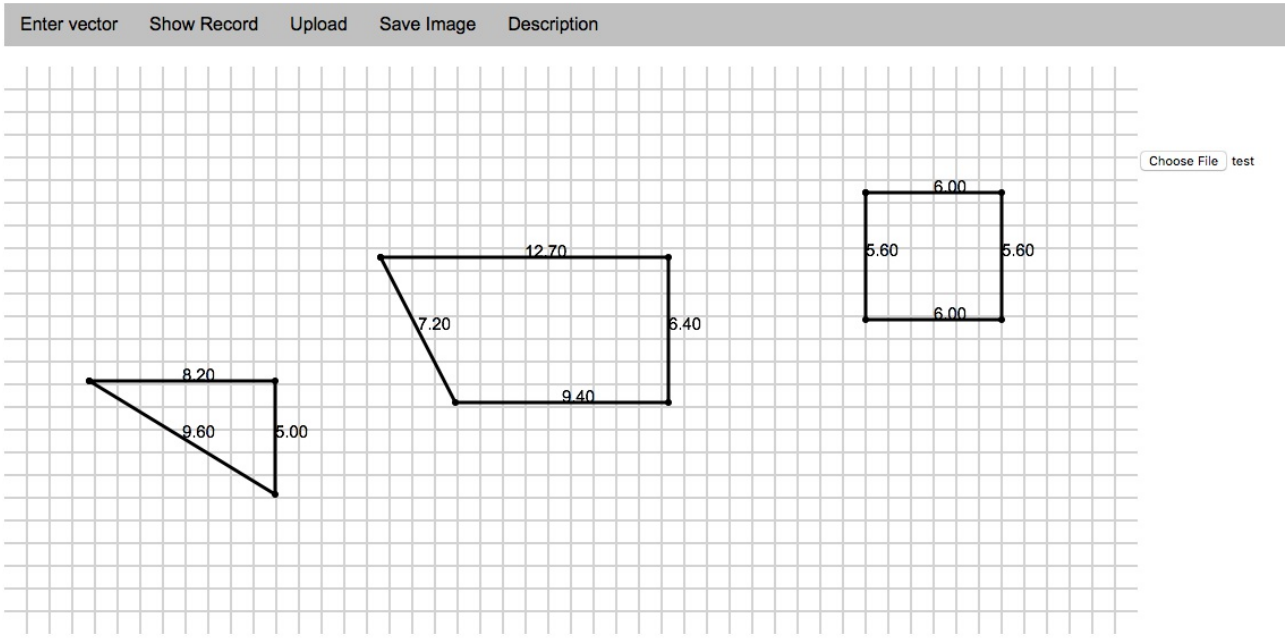
The code is displayed within the polygon. Only one code can be assigned to each polygon, but the polygon does not need a code to be assigned to it.



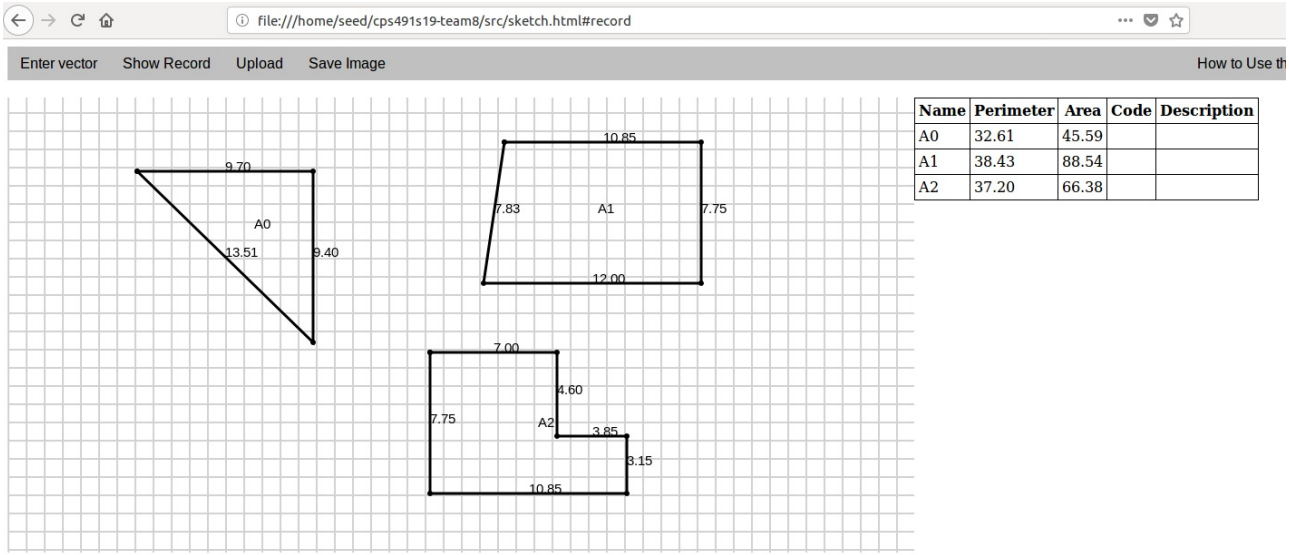
A file can be uploaded from the user's computer files.



The file contains the vector strings which are then displayed on the canvas as shown.



After Polygons are drawn, enter as stringed, or uploaded if you click on the "show record" button the polygon's perimeter and area will be displayed in a table



Name	Perimeter	Area	Code	Description
A0	32.61	45.59		
A1	38.43	88.54		
A2	37.20	66.38		

Acknowledgments

We would like to thank Tyler Technologies, Dwayne Nickels, Chris Drake, and Michael Lange for giving us this project and working with us throughout. We would also like to thank Dr. Phung for his guidance throughout the semester.