Scrum Log: Sprint 1

03/31

- Completed user stories
- Assigned priorities to all user stories
- Assigned developers to each user story
- Team Name: work in progress

04/01

- Finalized priorities
- Assigned initial work to each developer
    - Begin work on vector container: Aram, Aspen, and Luke
    - Reading from shapes.txt and parser / docker: Gautam, Tim, Paul
    - Creating abstract base class shape and derived classes: Eric, and Kevin

Remember to:

- Create your own branch named after the feature and work on that.
- Once the feature is complete, submit a pull request or message the chat so scrum master can merge into main branch

04/05:

- Working on templated vector
- Working on base shape class and derived classes

04/07: ↓↓↓

Basic Application Outline (the example for what shapes.json and render_area.json look like are at the very end of this document"

Web Service:

- Has 2 basic roles:
    - Storing all the project data into files so it is "saved" when the project is closed
    - Returning data as a string in a predefined formatted so it is easy to parse
- All of the project data that is stored by the webservice exists in 2 files
    - shapes.json: this is the file that stores all possible shapes that can be rendered. Its structure is below (note: there is exactly one entry for every shape).
    - render_area.json: this is the file that stores all of the shapes currently in the render window. Its structure is below (note: there can be 0-999 entries of any shape type for all shapes).
- Has 6 API endpoints

- **Get /shapes**: returns the shapes.json file as a string
- **Post /shapes**: updates the shapes.json file with new a new list of shapes
- Put /shape: updates the properties of a single existing shape in the shapes.json file
- Delete /shape: deletes one shape from the shapes.json file
- **Get /render_area**: returns the render_area.json file as a string
- **Post /render_area**: updates the render_area.json file with new data

Qt App Backend:

- Has functions that interact with the 6 api endpoints above to get data formatted as a string and to update the necessary files with new data
- App logic:
  - On app startup, the qt app reaches out to the two get endpoints and gets the data back formatted as a string
  - The data is then transformed from a string into a vector of shapes and stored in memory
    - This is done by the parser (built by us). The parser is a function that takes in the data formatted as a string and returns a vector of shapes
    - So:
      - Data from shapes.json -> parser -> vector<Shape> shapes
      - Data from render_area.json -> parser -> vector<Shape> renderedShapes
    - Now the data is in memory as vectors, making it easy to work with
  - With the data in memory, our front end code just "looks" at these two vectors in main and renders them in the UI.

Qt App Frontend:

- This is where all of the code to render the UI resides.
- The code for the render area is "looking" at the renderedShapes vector.
  - When a user clicks on a shape and changes it from lets say red to green, the following happens
    - The shape is identified by its trackingId
    - The corresponding element in the renderedShapes vector that matches the trackingId is found
    - Its data member is changed from red to green (or whatever the property is)
    - Since our code for the render is "looking" at this vector, after something is changed, it is immediately reflected in the render window
- The code for the shapes box is looking at the shapes vector
  - When an administrator adds or deletes a possible shape that can be rendered, the following happens

        - Same as above but now with the shapes vector using the shapeId to track down the element we need in the shapes vector
- However, we have a problem now, these changes are made in RAM so they are not permanent, meaning, when the app closes, all these changes are lost
- To make the changes permanent, the data must now flow from the frontend all the way back to the webservice, since that is what deals with persistent data

Qt App Backend continued:

- This part is all about "saving" our changes
- There are three main mechanisms that I can think off that will trigger the "save"
    - Save button: self explanatory
    - Interval timer: a save is automatically triggered every 5 seconds or similar
    - Inactivity timer: a save is triggered after 5 seconds or similar of inactivity
        - Inactivity = no shape has been updated or changed in 5 seconds
- Doesn't matter what the mechanism is, all that matters is what happens next
- When a save is triggered:
    - A copy of the renderdShapes vector is made
    - The copy is fed to a "reverse parser"
        - The reverse parser takes in a vector of shapes and spits out a string formatted as json (just like it came in from the Get endpoints)
    - The data from the reverse parser is then passed to the Post /render_area endpoint

Web Service continued:

- The data is received by the webserver through the post endpoint
- All of that data is then taken and overwritten into the render_area.json file
- The data is now "saved"
- What about the data for the shapes vector?
    - This data will be sent to be saved as soon as the shapes vector is updated instead of waiting for one of the three mechanisms from above
    - This is because the shapes vector will generally be updated and changed a lot less then the render area because once the shapes are set for a project, most people aren't going to be messing with them again
    - The route that the shapes vector goes through to be saved is the same as above but it obviously has its own API endpoint for it: Post /shapes
- So in summary
    - On app startup: data flows from the web service to the qt backend and frontend to build the selected project and hydrate the render windows
    - After the render area is hydrated and set up: data flows from the qt frontend and backend all the way back to the web service throughout the user session to save it in long term storage and make the data persist between user sessions

Other things

- The logic for creating a shape report (user stories 7, 8, and 9 i think) should be stored in the Qt backend.
    - When a shape report is triggered, it gets its data by making a copy of the renderedShapes vector then doing what it needs to do
- TrackingId:
    - In addition to a shapeId that separates a shape from all of the other shapes, we need a "trackingId" that separates a single circle from all of the other shapes
    - Why?
        - If a user has two circles in the render area and then goes to change lets say the color of one of them, how does the code know which one of the two circles to change? They both have the same shapeId cuz they're both the same shape, so that can't work
        - This is where the trackingId comes in, it separates one of those circles from the other so you know exactly which one to track down in memory and change
    - TrackingId structure: this can change but just what I came up with
        - A 4 digit integer where the first digit is the same as the shapeId
        - For example:
            - The valid trackingIds that can be assigned to rectangles ranges from 4000 - 4999
            - For Circles: 7000 - 7999
            - And so on…

Next steps:

- By April 15, we should be 95% done with the webserver code, 95% done with the abstract and derived classes code, and 95% done with the vector class code
- By April 15, we should be working on the Qt backend and everything that entails while another group could get started on the frontend code

This is what the string looks like when data is received from the endpoints and how it should look like when sending data back to the webservice
(Note: the objects in shapes.json will also have trackingId's since the trackingId is a data member of the shape abstract class and all of its derived classes. However, when the data is stored in shapes.json, the trackingId should be set to null since it is not necessary in shapes.json. Only reason they don't have an entry for trackingId here is because I'm lazy, but in the project they will)

shapes.json:


```
[
  {
    "ShapeId": 1,
    "ShapeType": "Line",
    "ShapeDimensions": [20, 90, 100, 20],
    "PenColor": "blue",
    "PenWidth": 2,
    "PenStyle": "DashDotLine",
    "PenCapStyle": "FlatCap",
    "PenJoinStyle": "MiterJoin"
  },
  {
    "ShapeId": 2,
    "ShapeType": "Polyline",
    "ShapeDimensions": [460, 90, 470, 20, 530, 40, 540, 80],
    "PenColor": "green",
    "PenWidth": 6,
    "PenStyle": "SolidLine",
    "PenCapStyle": "FlatCap",
    "PenJoinStyle": "MiterJoin"
  },
  {
    "ShapeId": 3,
    "ShapeType": "Polygon",
    "ShapeDimensions": [900, 90, 910, 20, 970, 40, 980, 80],
    "PenColor": "cyan",
    "PenWidth": 6,
    "PenStyle": "DashDotDotLine",
    "PenCapStyle": "FlatCap",
    "PenJoinStyle": "MiterJoin",
    "BrushColor": "yellow",
    "BrushStyle": "SolidPattern"
  },
  {
    "ShapeId": 4,
    "ShapeType": "Rectangle",
    "ShapeDimensions": [20, 200, 170, 100],
    "PenColor": "blue",
```

```
  "PenWidth": 0,
  "PenStyle": "DashLine",
  "PenCapStyle": "RoundCap",
  "PenJoinStyle": "RoundJoin",
  "BrushColor": "red",
  "BrushStyle": "VerPattern"
},
{
  "ShapeId": 5,
  "ShapeType": "Square",
  "ShapeDimensions": [250, 150, 200],
  "PenColor": "red",
  "PenWidth": 0,
  "PenStyle": "SolidLine",
  "PenCapStyle": "RoundCap",
  "PenJoinStyle": "RoundJoin",
  "BrushColor": "blue",
  "BrushStyle": "HorPattern"
},
{
  "ShapeId": 6,
  "ShapeType": "Ellipse",
  "ShapeDimensions": [520, 200, 170, 100],
  "PenColor": "black",
  "PenWidth": 12,
  "PenStyle": "SolidLine",
  "PenCapStyle": "FlatCap",
  "PenJoinStyle": "MiterJoin",
  "BrushColor": "white",
  "BrushStyle": "NoBrush"
},
{
  "ShapeId": 7,
  "ShapeType": "Circle",
  "ShapeDimensions": [750, 150, 200],
  "PenColor": "black",
  "PenWidth": 12,
  "PenStyle": "SolidLine",
  "PenCapStyle": "FlatCap",
  "PenJoinStyle": "MiterJoin",
```

```
    "BrushColor": "magenta",
    "BrushStyle": "SolidPattern"
   },
   {
    "ShapeId": 8,
    "ShapeType": "Text",
    "ShapeDimensions": [250, 425, 500, 50],
    "TextString": "Class Project 2 - 2D Graphics Modeler",
    "TextColor": "blue",
    "TextAlignment": "AlignCenter",
    "TextPointSize": 10,
    "TextFontFamily": "Comic Sans MS",
    "TextFontStyle": "StyleItalic",
    "TextFontWeight": "Normal"
   }
 ]
```


render_area.json:


```
[
   {
    "ShapeId": 1,
    "ShapeType": "Line",
    "ShapeDimensions": [34, 76, 123, 89],
    "PenColor": "red",
    "PenWidth": 3,
    "PenStyle": "DashLine",
    "PenCapStyle": "RoundCap",
    "PenJoinStyle": "BevelJoin",
    "TrackingId": 1320
   },
   {
    "ShapeId": 2,
    "ShapeType": "Polyline",
    "ShapeDimensions": [50, 60, 80, 20, 100, 100],
    "PenColor": "green",
    "PenWidth": 4,
    "PenStyle": "SolidLine",
    "PenCapStyle": "FlatCap",
    "PenJoinStyle": "MiterJoin",
```

```
  "TrackingId": 2471
 },
 {
  "ShapeId": 3,
  "ShapeType": "Polygon",
  "ShapeDimensions": [300, 300, 350, 250, 400, 300],
  "PenColor": "blue",
  "PenWidth": 5,
  "PenStyle": "DotLine",
  "PenCapStyle": "FlatCap",
  "PenJoinStyle": "MiterJoin",
  "BrushColor": "yellow",
  "BrushStyle": "SolidPattern",
  "TrackingId": 3619
 },
 {
  "ShapeId": 4,
  "ShapeType": "Rectangle",
  "ShapeDimensions": [400, 400, 150, 100],
  "PenColor": "blue",
  "PenWidth": 2,
  "PenStyle": "DashDotLine",
  "PenCapStyle": "RoundCap",
  "PenJoinStyle": "RoundJoin",
  "BrushColor": "cyan",
  "BrushStyle": "HorPattern",
  "TrackingId": 4383
 },
 {
  "ShapeId": 5,
  "ShapeType": "Square",
  "ShapeDimensions": [500, 500, 80],
  "PenColor": "orange",
  "PenWidth": 1,
  "PenStyle": "SolidLine",
  "PenCapStyle": "RoundCap",
  "PenJoinStyle": "RoundJoin",
  "BrushColor": "magenta",
  "BrushStyle": "VerPattern",
  "TrackingId": 5524
```

```
},
{
 "ShapeId": 6,
 "ShapeType": "Ellipse",
 "ShapeDimensions": [200, 200, 100, 60],
 "PenColor": "black",
 "PenWidth": 6,
 "PenStyle": "DashDotDotLine",
 "PenCapStyle": "FlatCap",
 "PenJoinStyle": "MiterJoin",
 "BrushColor": "green",
 "BrushStyle": "SolidPattern",
 "TrackingId": 6852
},
{
 "ShapeId": 7,
 "ShapeType": "Circle",
 "ShapeDimensions": [300, 300, 90],
 "PenColor": "black",
 "PenWidth": 8,
 "PenStyle": "SolidLine",
 "PenCapStyle": "FlatCap",
 "PenJoinStyle": "MiterJoin",
 "BrushColor": "red",
 "BrushStyle": "Dense1Pattern",
 "TrackingId": 7901
},
{
 "ShapeId": 8,
 "ShapeType": "Text",
 "ShapeDimensions": [100, 600, 300, 40],
 "TextString": "Sample Text A",
 "TextColor": "purple",
 "TextAlignment": "AlignCenter",
 "TextPointSize": 14,
 "TextFontFamily": "Arial",
 "TextFontStyle": "StyleNormal",
 "TextFontWeight": "Bold",
 "TrackingId": 8675
},
```

```
{
 "ShapeId": 3,
 "ShapeType": "Polygon",
 "ShapeDimensions": [100, 100, 150, 50, 200, 100],
 "PenColor": "teal",
 "PenWidth": 2,
 "PenStyle": "SolidLine",
 "PenCapStyle": "FlatCap",
 "PenJoinStyle": "MiterJoin",
 "BrushColor": "pink",
 "BrushStyle": "Dense4Pattern",
 "TrackingId": 3214
},
{
 "ShapeId": 4,
 "ShapeType": "Rectangle",
 "ShapeDimensions": [10, 10, 200, 100],
 "PenColor": "gray",
 "PenWidth": 5,
 "PenStyle": "DashLine",
 "PenCapStyle": "RoundCap",
 "PenJoinStyle": "RoundJoin",
 "BrushColor": "orange",
 "BrushStyle": "NoBrush",
 "TrackingId": 4098
},
{
 "ShapeId": 5,
 "ShapeType": "Square",
 "ShapeDimensions": [50, 50, 50],
 "PenColor": "red",
 "PenWidth": 2,
 "PenStyle": "DotLine",
 "PenCapStyle": "FlatCap",
 "PenJoinStyle": "RoundJoin",
 "BrushColor": "blue",
 "BrushStyle": "DiagCrossPattern",
 "TrackingId": 5890
},
{
```

```
  "ShapeId": 6,
  "ShapeType": "Ellipse",
  "ShapeDimensions": [600, 200, 120, 80],
  "PenColor": "green",
  "PenWidth": 3,
  "PenStyle": "DashLine",
  "PenCapStyle": "FlatCap",
  "PenJoinStyle": "BevelJoin",
  "BrushColor": "white",
  "BrushStyle": "Dense7Pattern",
  "TrackingId": 6799
},
{
  "ShapeId": 1,
  "ShapeType": "Line",
  "ShapeDimensions": [0, 0, 500, 500],
  "PenColor": "blue",
  "PenWidth": 4,
  "PenStyle": "DashDotDotLine",
  "PenCapStyle": "SquareCap",
  "PenJoinStyle": "MiterJoin",
  "TrackingId": 1534
},
{
  "ShapeId": 2,
  "ShapeType": "Polyline",
  "ShapeDimensions": [20, 30, 40, 50, 60, 70],
  "PenColor": "navy",
  "PenWidth": 2,
  "PenStyle": "SolidLine",
  "PenCapStyle": "RoundCap",
  "PenJoinStyle": "MiterJoin",
  "TrackingId": 2670
},
{
  "ShapeId": 7,
  "ShapeType": "Circle",
  "ShapeDimensions": [100, 100, 60],
  "PenColor": "maroon",
  "PenWidth": 1,
```

```
  "PenStyle": "SolidLine",
  "PenCapStyle": "FlatCap",
  "PenJoinStyle": "BevelJoin",
  "BrushColor": "yellow",
  "BrushStyle": "HorPattern",
  "TrackingId": 7104
},
{
  "ShapeId": 8,
  "ShapeType": "Text",
  "ShapeDimensions": [200, 700, 400, 30],
  "TextString": "Test Shape Text",
  "TextColor": "black",
  "TextAlignment": "AlignLeft",
  "TextPointSize": 12,
  "TextFontFamily": "Courier New",
  "TextFontStyle": "StyleItalic",
  "TextFontWeight": "Normal",
  "TrackingId": 8999
},
{
  "ShapeId": 6,
  "ShapeType": "Ellipse",
  "ShapeDimensions": [150, 150, 90, 45],
  "PenColor": "cyan",
  "PenWidth": 3,
  "PenStyle": "DotLine",
  "PenCapStyle": "FlatCap",
  "PenJoinStyle": "MiterJoin",
  "BrushColor": "blue",
  "BrushStyle": "Dense3Pattern",
  "TrackingId": 6044
},
{
  "ShapeId": 3,
  "ShapeType": "Polygon",
  "ShapeDimensions": [700, 100, 750, 50, 800, 100],
  "PenColor": "black",
  "PenWidth": 2,
  "PenStyle": "DashDotLine",
```

```
    "PenCapStyle": "FlatCap",
    "PenJoinStyle": "MiterJoin",
    "BrushColor": "gray",
    "BrushStyle": "CrossPattern",
    "TrackingId": 3933
   },
   {
    "ShapeId": 4,
    "ShapeType": "Rectangle",
    "ShapeDimensions": [300, 300, 100, 200],
    "PenColor": "lime",
    "PenWidth": 3,
    "PenStyle": "SolidLine",
    "PenCapStyle": "RoundCap",
    "PenJoinStyle": "RoundJoin",
    "BrushColor": "purple",
    "BrushStyle": "Dense6Pattern",
    "TrackingId": 4632
   },
   {
    "ShapeId": 5,
    "ShapeType": "Square",
    "ShapeDimensions": [100, 100, 120],
    "PenColor": "black",
    "PenWidth": 4,
    "PenStyle": "DashLine",
    "PenCapStyle": "RoundCap",
    "PenJoinStyle": "RoundJoin",
    "BrushColor": "red",
    "BrushStyle": "SolidPattern",
    "TrackingId": 5012
   }
 ]
```


04/08:

- Mutator and accessor functions created for shape class
- Created ApiClient class to interact with APIs

04/09:

- Vector finished and merged with main

- Web Service and Qt-Backend API done and merged with main

04/10:

- Fixed draw bugs
- Added destructors to polygon and polyline

04/11:

- Inheritance (shape base class and derived classes) done and merged with main
- Bugs fixed
- Project moved to Qt

04/12:

- Reorganized files
- Working on parser

04/13:

- Working on render area
- Updated all derived classes
- Parser done and merged with main

2

04/15:

- Fixed parser bugs
- Completed reverse parser

04/16:

- Working on rendering area
- Working on UI
- Fixed docker and other bugs
- Working on user accounts (guest & admin)

04/17:

- Working on shape sorting
- Built basic slots and signals
- Added shape selection
- Fixed text rendering

04/18:

- Created and finished move mouse event
- Fixed polygon and polyline

Scrum Log: Sprint 2

04/19:

- Added style sheet
- Working on UI

04/20:

- Working on tree widget and UI
- Fixed vector

04/21:

- Working on tree widget

04/22:

- Working on tree widget
- Fixed vector
- Done and merged shape sorting to main
- Done and merged render-area to main
- Updated docker
- Fixed lots of bugs

04/23:

- Display shape ID in render area
- Fixed more bugs

04/24:

- Working on shape selector
- Added shape buttons to UI
- Fixed shape rendering
- Slots to main window

04/25:

- Done darkmode version
- Login done
- Fixed bugs: darkmode, selection sort, few others
- Added properties in tree widget

04/26:

- Fixed memory leaks & cleaned code
- Updated darkmode
- Added icons
- Updated and fixed: login, tree widget, tracker id and a few others

04/27:

- Clean up frontend and backend
- Working on mouse double click event
- Done shape editing
- Fixed bugs
- Updated darkmode, render area, shape buttons and few more

04/28:

- Fixed tree widget to update when shapes are moved
- Done shape editing
- Done admin features (only admin can create shapes)

04/29:

- Done testimonials
- Fixed bugs
- Selection sort done and merged with main
- Updated render area
- Done contact us and merged with main