Developing the Position Visualization

Improving the Grand Prix experience for F1 viewers at home

S8 Graduation FHICT

4 Sept 2023 - 16 Jan 2024

By Jordi Franssen

2 januari 2024

Introduction

The core business of the concept is finished. The datastream with Formula 1 data gets processed into events, sent to the CMS of RN365, after which the editorial can post the event right away or edit them to their liking.

Another part of the concept was a 3-dimensional visualization of the race track. This is to help users get a better idea of where events take place on the track. In an interview with RN365 it was agreed not to visualize actual location data, as it would be too obvious that we’re taking data from Formula 1. Instead, only the location of where an event took place will be visualized as a pin on the track. Therefore, this is more like a side issue instead of being part of the core business. Chances are that this won't be used in the final product after all, but I will add it to the MVP to give TDE advice on how to build it if they ever intend to add it to the final product.

Approach

What is exactly going to be built?

The 3D visualization is just going to be an extension of the MVP that was built for the core business of the concept. However, since there's a chance the 3D visualization won't be implemented, and the semester is coming to an end, I decided to keep it a bit more minimalistic.

I'm planning to have the 3D visualization work with only the development event that I have been using for developing the MVP and the development API, namely the 2023 Las Vegas GP. I will elaborate on the backend API to pass the location data to the CMS and modify the CMS and liveblog backend so the location data can be passed to the liveblog for the user.

Next I will build a simple 3D model to use as a racetrack. In the end product, it would be desirable to add some props to the model like trees and buildings so it's not just a black circle. At last, the logic for visualizing the pointers will have to be coded in the liveblog.

After a meeting with RN365, it became clear that the CMS needs a little map so the editor can manually input the location of an event. I have a rough idea of how I can build this and this shouldn't be too difficult.

Change of approach after RN365 feedback

In the feedback meeting with RN365, it was discussed that the position visualization is going to be a simple image. 3D doesn't provide any more context and it makes it a lot easier to develop as there is a challenge to match the scale of the 3D model with the scale Formula 1 is using in their data source.

We discussed solving this by generating the shape and measurements of the race track according to the data in the data stream. Earlier in the project, I made a POC that visualizes location data by placing a dot for every location update. This POC drew the shape of the track as a car completed a lap. As this is based on the data and scale in the data source, the shape matched the scale of the data source. I want to elaborate this POC into a tool that can generate the shape and scale of a racetrack. This can be done with data from the free practice on friday, so the shape is available for the qualifying and race later in the weekend.

The approach and end product is now completely different, the first step will be to make the location data from overtakes and potential crashes available in the liveblog. Then the tool to generate the shape of the racetrack will have to be built. This will have to be implemented in the CMS and liveblog.

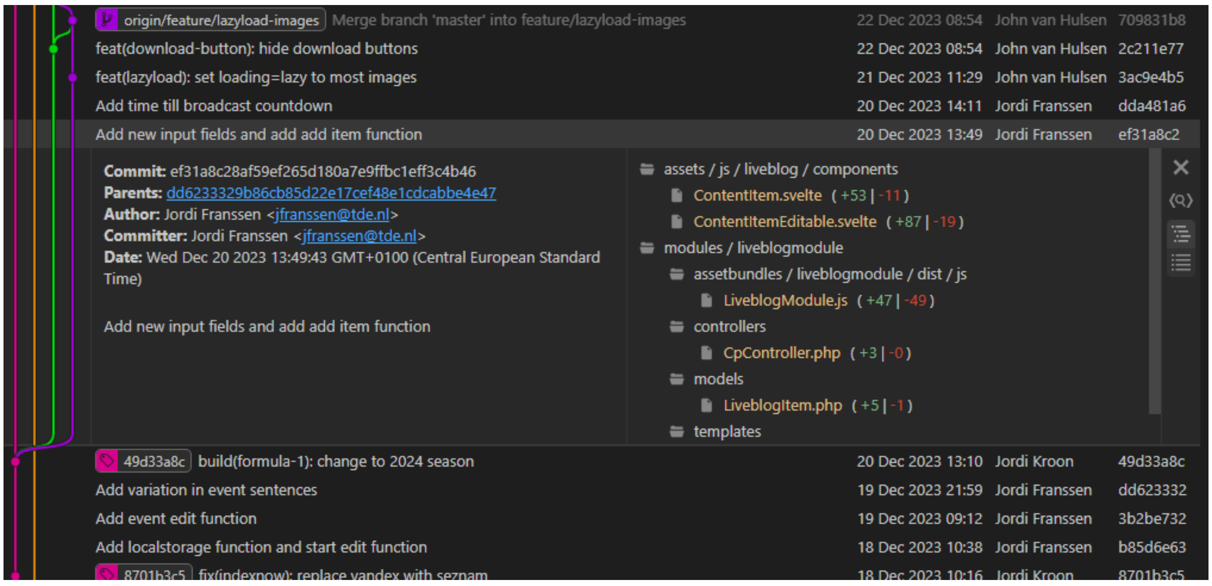
Methodology

I have been using the Ad Hoc method to build my MVP and want to apply this method again. The way of working won't differ much from how I built the MVP as it is now. I also don't have much experience in Blender and 3D on the web, so it's difficult to estimate the amount of time required for each step and I'm not entirely sure how smooth the development will go. Therefore, decisions will be made based on how progress is going.

Making the position data available in the front-end

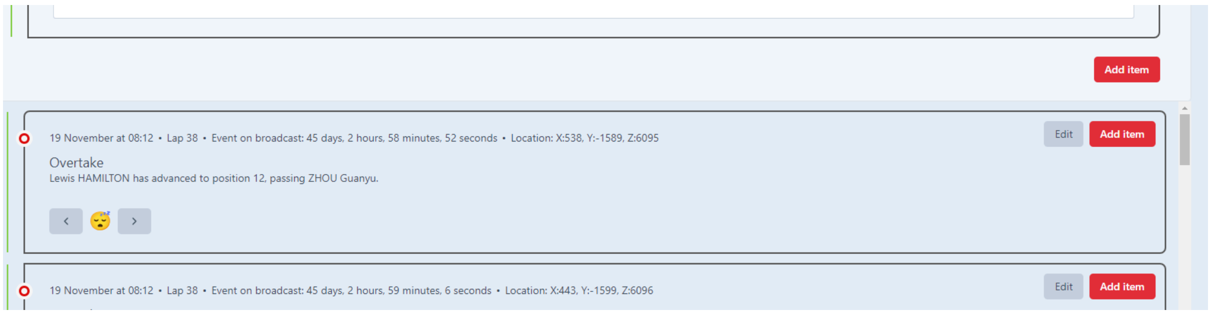
It turns out that I already made the location of the overtake available with the script that detects events, so this data is already available in the CMS, it only needs to be passed down with the liveblog item.

This required some backend development so this modification was quite complex for me. Fortunately, because I use Git I could see what changes Jordi made when he helped me with making the trackstatus and lap count available in the liveblog. All I needed was three integers for the X, Y and Z value to be passed down. I used the lap count as an example, as this is an integer as well.



```
origin/feature/lazyload-images  Merge branch 'master' into feature/lazyload-images   22 Dec 2023 08:54  John van Hulsen  709831b8
feat(download-button): hide download buttons                                         22 Dec 2023 08:54  John van Hulsen  2c211e77
feat(lazyload): set loading=lazy to most images                                      21 Dec 2023 11:29  John van Hulsen  3ac9e4b5
Add time till broadcast countdown                                                    20 Dec 2023 14:11  Jordi Franssen   dda481a6
Add new input fields and add add item function                                       20 Dec 2023 13:49  Jordi Franssen   ef31a8c2

  Commit: ef31a8c28af59ef265d180a7e9ffbc1eff3c4b46             assets / js / liveblog / components
  Parents: dd6233329b86cb85d22e17cef48e1cdcabbe4e47               ContentItem.svelte ( +53 | -11 )
  Author: Jordi Franssen <jfranssen@tde.nl>                       ContentItemEditable.svelte ( +87 | -19 )
  Committer: Jordi Franssen <jfranssen@tde.nl>                 modules / liveblogmodule
  Date: Wed Dec 20 2023 13:49:43 GMT+0100 (Central European Standard   assetbundles / liveblogmodule / dist / js
  Time)                                                                LiveblogModule.js ( +47 | -49 )
                                                                    controllers
  Add new input fields and add add item function                      CpController.php ( +3 | -0 )
                                                                    models
                                                                      LiveblogItem.php ( +5 | -1 )
                                                                    templates

49d33a8c  build(formula-1): change to 2024 season                                    20 Dec 2023 13:10  Jordi Kroon      49d33a8c
Add variation in event sentences                                                     19 Dec 2023 21:59  Jordi Franssen   dd623332
Add event edit function                                                              19 Dec 2023 09:12  Jordi Franssen   3b2be732
Add localstorage function and start edit function                                    18 Dec 2023 10:38  Jordi Franssen   b85d6e63
8701b3c5  fix(indexnow): replace yandex with seznam                                  18 Dec 2023 10:16  Jordi Kroon      8701b3c5
```

Used a previous git commit to determine which files to change and how to make the changes to add the location data.

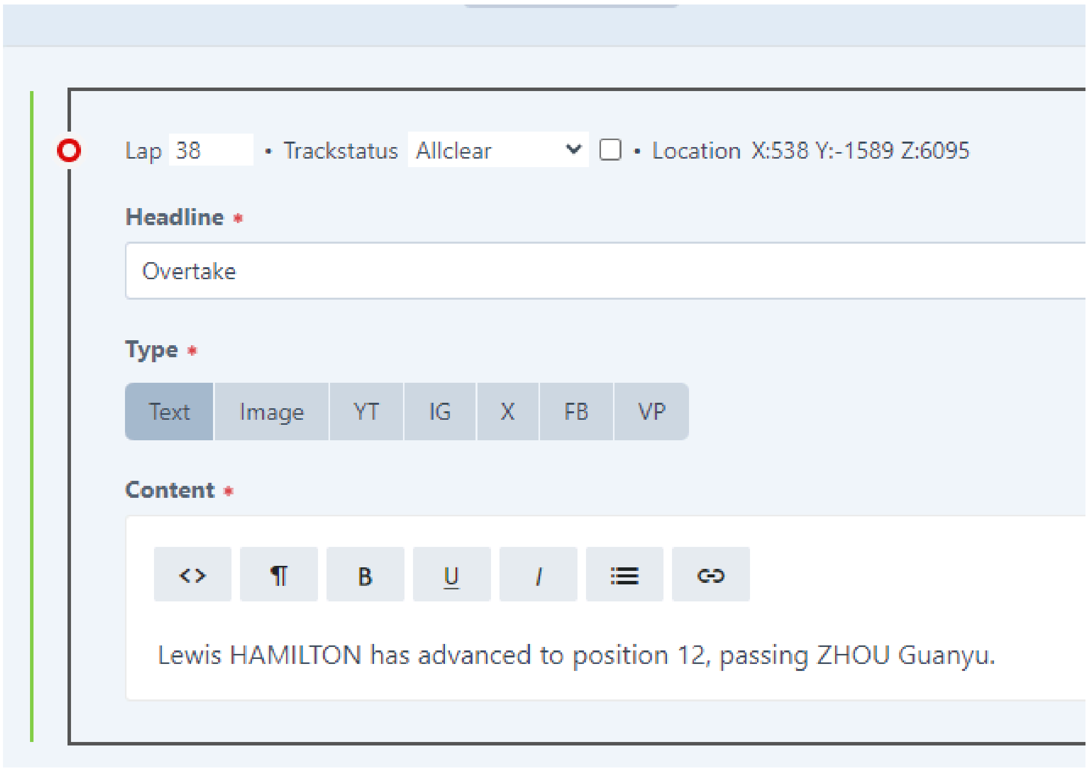


19 November at 08:12 • Lap 38 • Event on broadcast: 45 days, 2 hours, 58 minutes, 52 seconds • Location: X:538, Y:-1589, Z:6095

Overtake
Lewis HAMILTON has advanced to position 12, passing ZHOU Guanyu.

19 November at 08:12 • Lap 38 • Event on broadcast: 45 days, 2 hours, 59 minutes, 6 seconds • Location: X:443, Y:-1599, Z:6096

If an overtake happens, the X, Y and Z values will be displayed in the automated liveblog item, these values will be loaded in the editor when the item is added or edited.



Lap 38 • Trackstatus Allclear • Location X:538 Y:-1589 Z:6095

Headline *

Overtake

Type *

Content *

Lewis HAMILTON has advanced to position 12, passing ZHOU Guanyu.

I added three invisible input fields in the form to include the X, Y and Z values in the submit function. Now the location information is included with each liveblog item.

Making the tool for generating the shape of the track

The challenge in using the location data to point a location on the track is that the image used to visualize the location needs to have the exact same scale as the data in the data source. John and I discussed an approach and we figured that generating an image based on the data from the data source would be the perfect solution as this image will then always match with the data from the data source. This will be done in a separate tool.

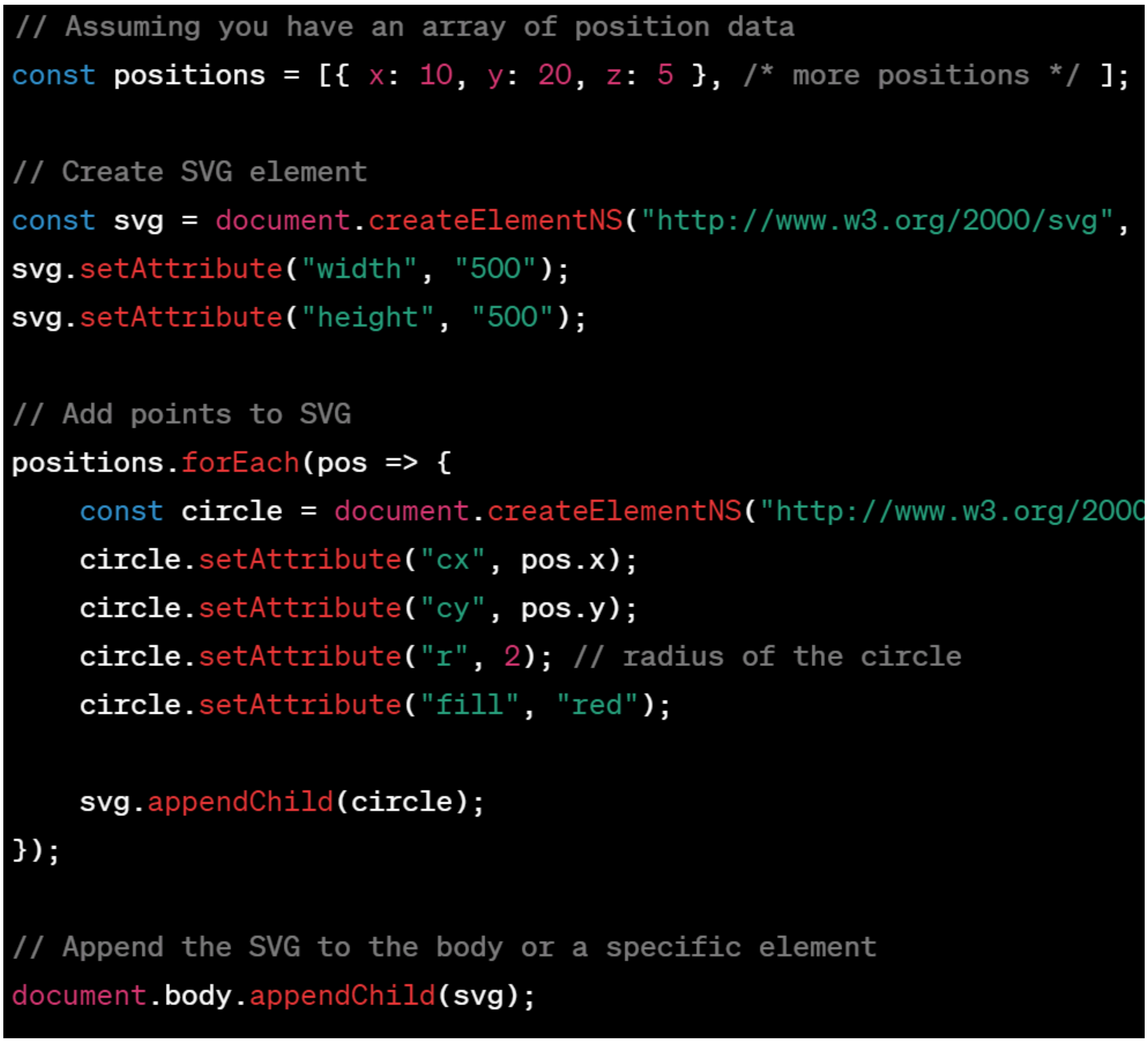
Selecting, downloading and processing the data

The user of the tool will need to make a selection of the track. This selection is based on the data available in the Formula 1 live timing static API. This way, the user can generate the shape of a track right away when data is available in the API.

Next, the logic for downloading and processing the data was already built in a POC I made earlier in the project. This POC downloads the jsonstream, reads each line and decrypts it to send it to the client to simulate the data source from the live API.

Generating an .svg image

I investigated if it's possible to generate an .svg image with javascript. With some input from my code assistant ChatGPT, this seemed like the way to go. I wanted to use .svg as this is a text based image. Therefore, it's easier to store in the database. However, ChatGPT warned of some performance issues. I gave it a try anyway and crashed my browser several times.

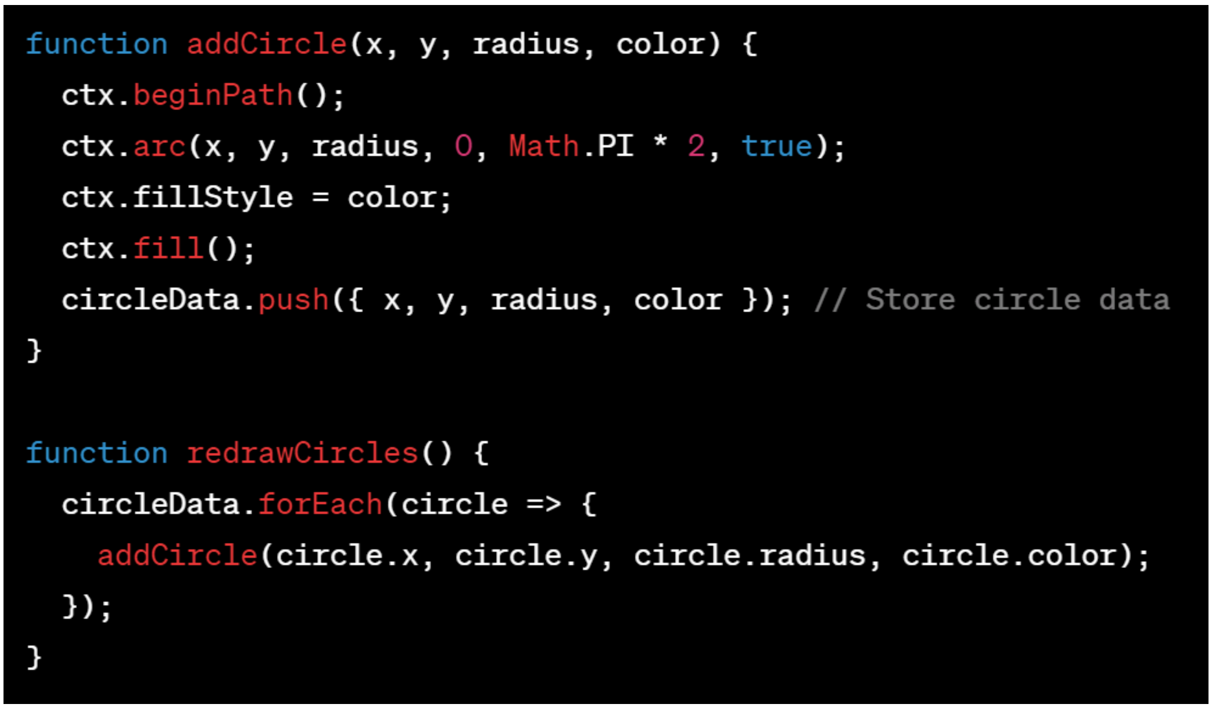


```javascript
// Assuming you have an array of position data
const positions = [{ x: 10, y: 20, z: 5 }, /* more positions */ ];

// Create SVG element
const svg = document.createElementNS("http://www.w3.org/2000/svg",
svg.setAttribute("width", "500");
svg.setAttribute("height", "500");

// Add points to SVG
positions.forEach(pos => {
    const circle = document.createElementNS("http://www.w3.org/2000
    circle.setAttribute("cx", pos.x);
    circle.setAttribute("cy", pos.y);
    circle.setAttribute("r", 2); // radius of the circle
    circle.setAttribute("fill", "red");

    svg.appendChild(circle);
});

// Append the SVG to the body or a specific element
document.body.appendChild(svg);
```

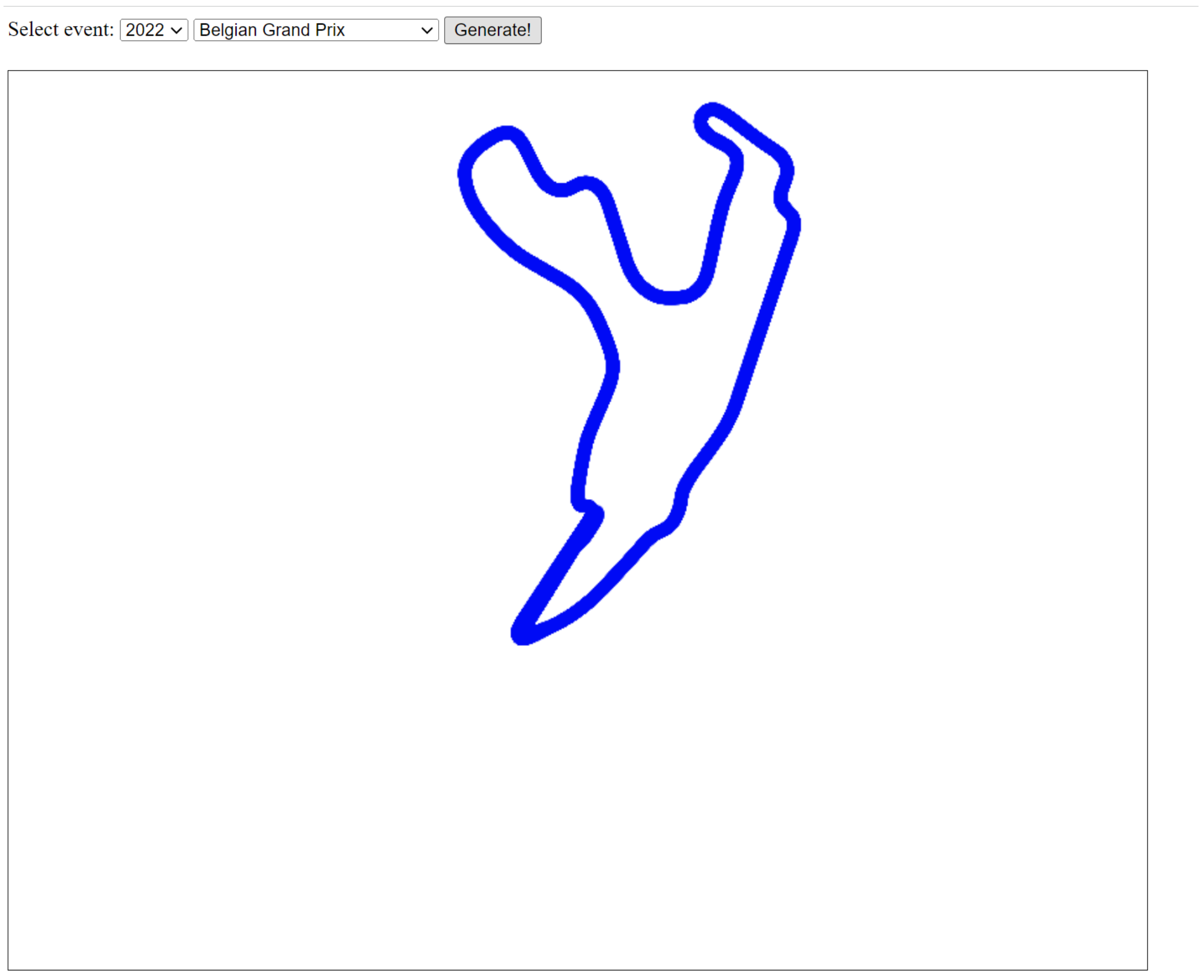
It seems like DOM manipulation on svg tags is just really heavy on memory for some reason. Add to the mix that the client needs to handle 20,000 position updates and the browser will crash. Therefore, I tried it by using HTML canvas instead.

Generating the shape in a canvas

John already suggested using canvas, so I tried this instead. This time the performance was good. By using the x and y values, I was able to place a dot on the position where a car was driving. I was eventually forced to scale the values from the data source, as the values were simply too large to display in pixels on the screen. I have divided all values by 20 and now the shapes fit easily on the screen.

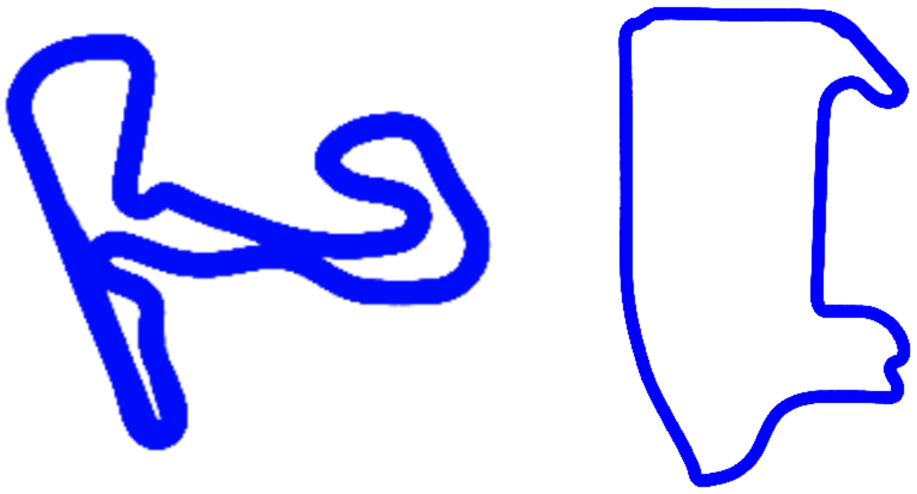


```
function addCircle(x, y, radius, color) {
  ctx.beginPath();
  ctx.arc(x, y, radius, 0, Math.PI * 2, true);
  ctx.fillStyle = color;
  ctx.fill();
  circleData.push({ x, y, radius, color }); // Store circle data
}

function redrawCircles() {
  circleData.forEach(circle => {
    addCircle(circle.x, circle.y, circle.radius, circle.color);
  });
}
```


Select event: 2022 Belgian Grand Prix
Generate!

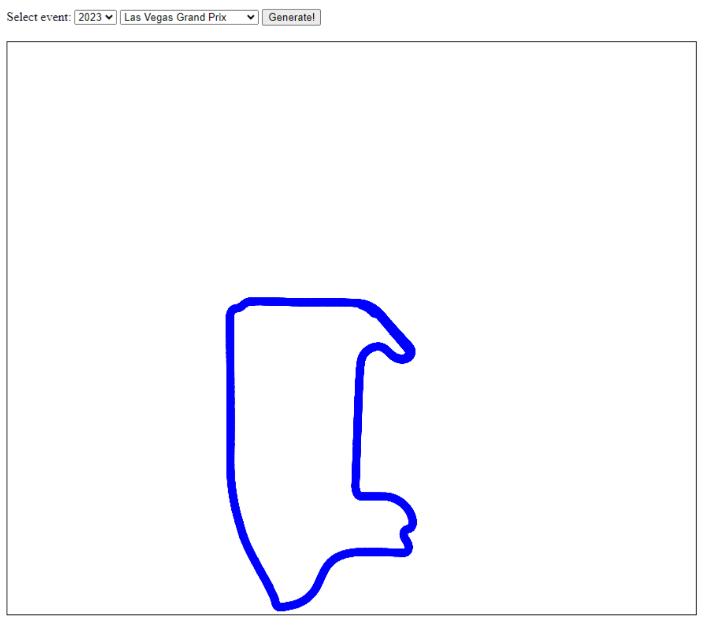
Challenges

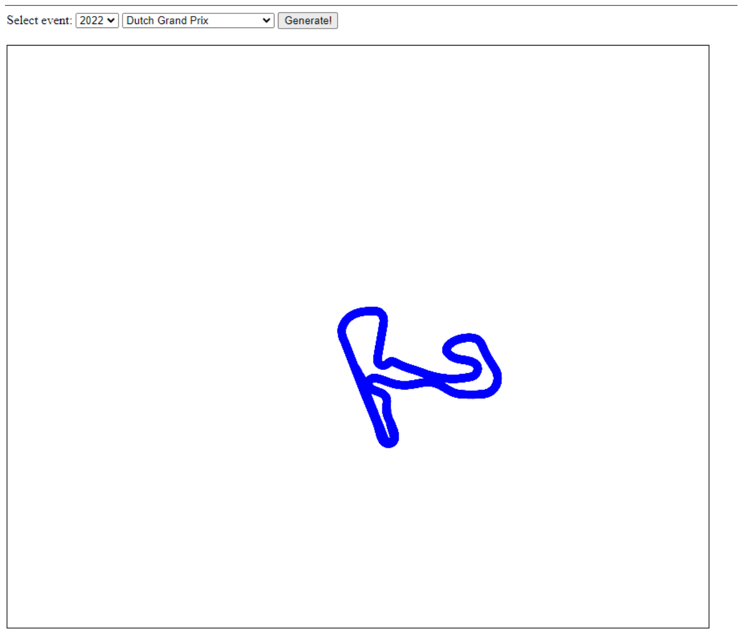
Generating the shape this way works well. However, there are some problems with the sizes of the shapes. For one, each track has a different size. The Zandvoort Dutch GP track is much smaller than the new Las Vegas GP track. The radius of the dots placed is too large for the Dutch GP, making that the lines touch each other where they shouldn't. So somehow, the radius of the dots needs to be adjusted based on the values in the datasource.



Dutch GP vs. Las Vegas GP track size.

The other challenge is the position of where the shape is drawn. X:0 and Y:0 are not always the center of the track. So, if the shape is cropped, it's likely impossible to preserve the location data as resolution of the image.





Feedback from Erik and John

Erik suggested that with some Mathematics and some clever thinking, these challenges would definitely be possible to overcome. For example, it should be possible to calculate the height and width of the drawn shape to resize the canvas. Next, the shape needs to be redrawn with the height and width of the shape subtracted from the data in the API. This should make it possible to crop the shape, and if the height and width values are saved somewhere, they can be used to pinpoint an event on the shape.

John suggested that with this information, I should try again to crop and scale the shape of the tracks, as this is valuable information for further development of the project.

What does this add to the project?

At first, I assumed it would be easy to make this tool and to generate an .svg based on the data in the data source. However, my plan fell into pieces when my browser crashed while trying to generate the svg tag with Javascript, resulting in a different approach which is to generate the shape in a canvas.

This method works, but it’s difficult to crop the canvas to make the shape fill the entire image. Erik suggested in an update that this is definitely possible with some javascript mathematics, but that it’s quite difficult.

Considering the challenges discovered and the closing deadline, I decided to deliver this tool as a Proof of Concept instead of including it as an MVP. The challenges that I discovered are simply too complex to fix before the deadline. Therefore, the elaboration of this POC to an end product will be included in the transfer document for the entire project.

Summary

In a feedback meeting with RN365, it was discussed that the positions of the events happening during a race will not be visualized in 3D, but simply in an image. This is because 3D doesn't provide any more context to the user and building it is quite complex.

To make the shape, we have to take the scale and position of the track into account. We discussed an approach and decided to make a separate tool that uses the location data in the data source to generate the shape of the track. This way the scale of the shape will always match with the data in the data source.

I started looping through the available data in the static API. This way a file can be selected that will be downloaded by the server. The logic for reading the file line by line, decrypting the content and sending it over to the client was taken from the PoC that I made earlier.

This works but I still found some challenges in cropping the canvas and I will deliver this tool as a PoC for generating the shapes instead of including it in the MVP with the liveblog.

Learning Outcome Clarification

- Learning Outcome 1: Professional Duties

Learning outcome 1 applies for this deliverable as this is a professional duty on a bachelor level in the activity of realization as I realized a POC for generating the shape of a racetrack.