Computer Graphics: Parametric Curves Website

By Shani Bar-Gera (ID: 314022989)

Supervisors: Boaz Sternfeld

Summer 2021

Table of Contents

Introduction……..…….……..……..…...3

Website Overview……...……...……...…….....4

Technologies and Platforms……..…….…….14

Development Process……..…….....….16

Designing the Layout.....16

Finding a Website Skeleton.....20

Mergin Skeleton with Initial Layout.....22

Creating the Curves.....24

Improving the UI.....32

Future Work…….……...….34

Conclusion……...…….....36

References……..……..…..37

Introduction

The goal of the project was to create a website, that enables to portray different parametric curves learned in the computer graphics course, in a way that makes it easier to learn them.

The user should be able view 6 different curves learned in computer graphics class:

- A monomial basis curve
- Lagrange Curve
- Cubic Hermite Spline
- Cubic Spline
- Bezier Curve
- B-Spline

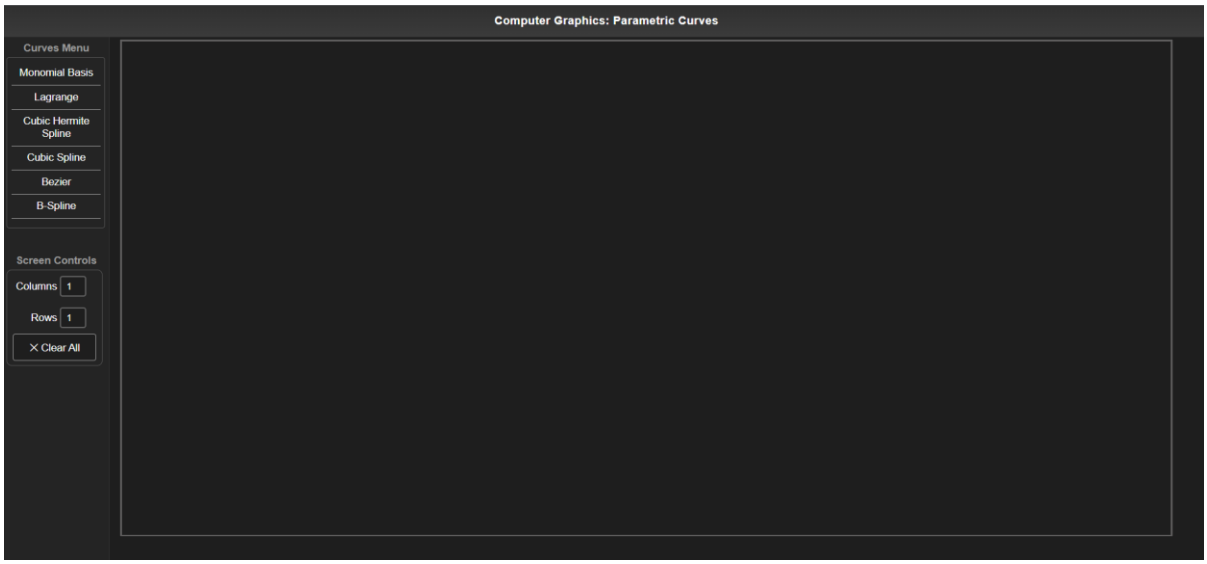
The user shuld be able to:

- Choose which curve to view
- Play with the curve’s control points by moving, adding and deleting them
- View the equation of the chosen curve and change it’s paramteres
- Split the screen in order to show multiple curves at once for comparison

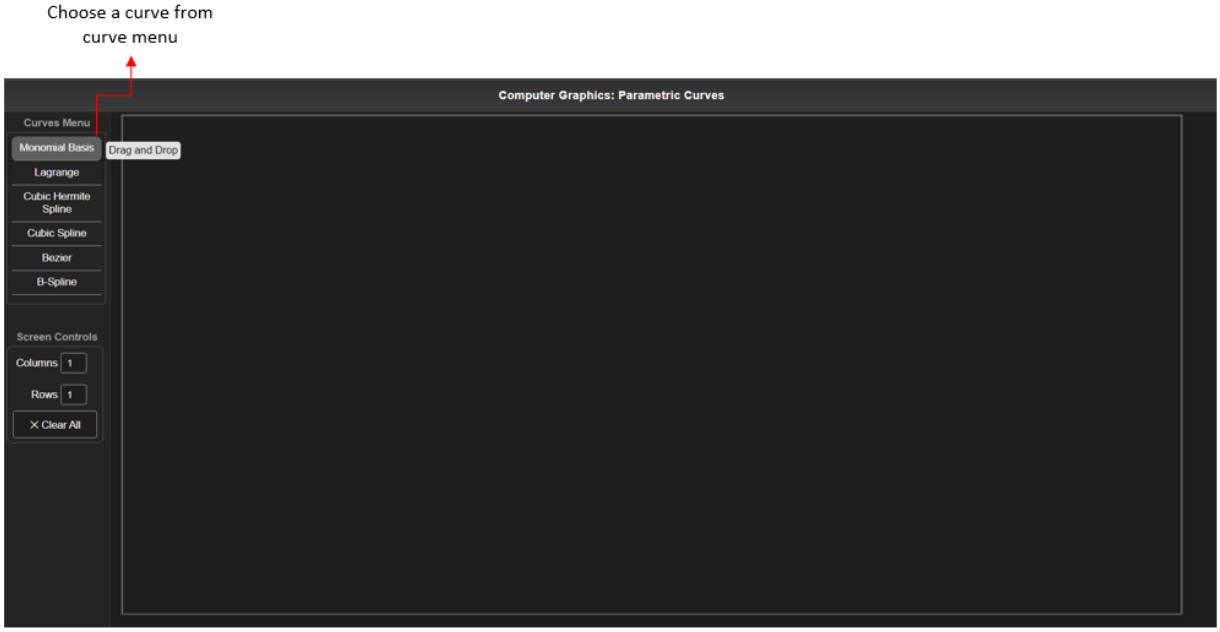
I decided to achieve this goal by creating a website using JavaScript that is based on a former existing website and adjusted it to the new demands.

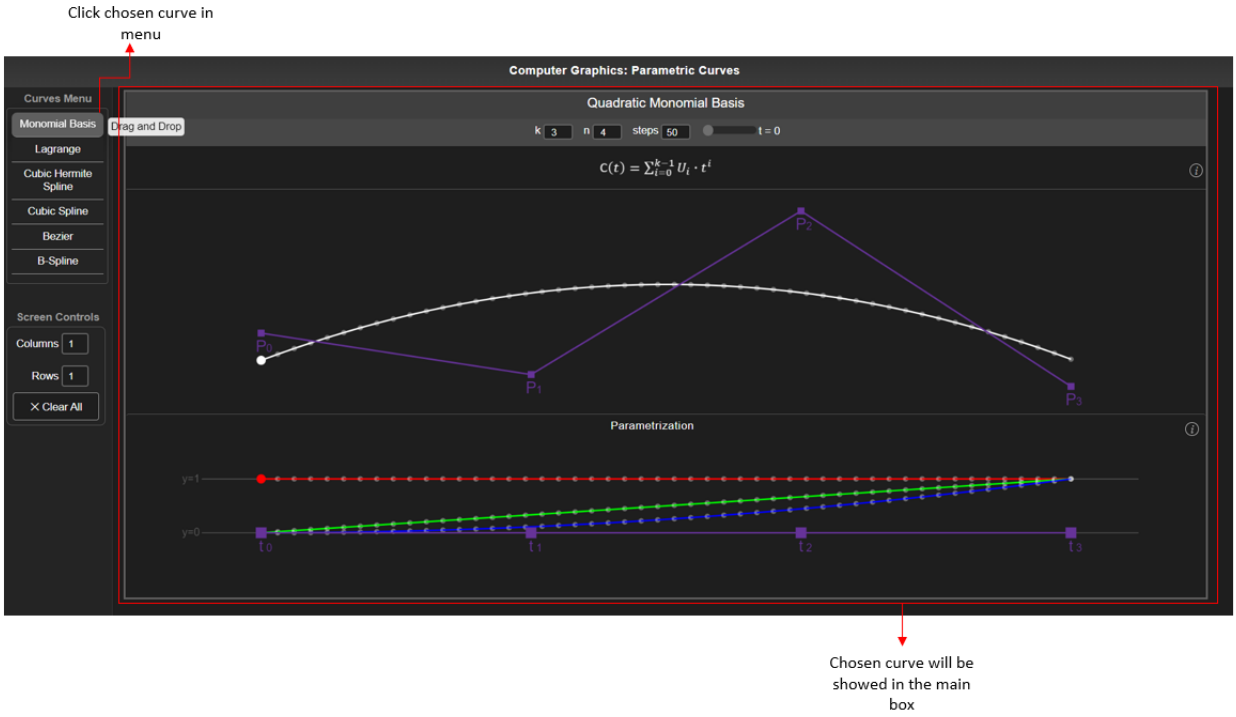
Website Overview

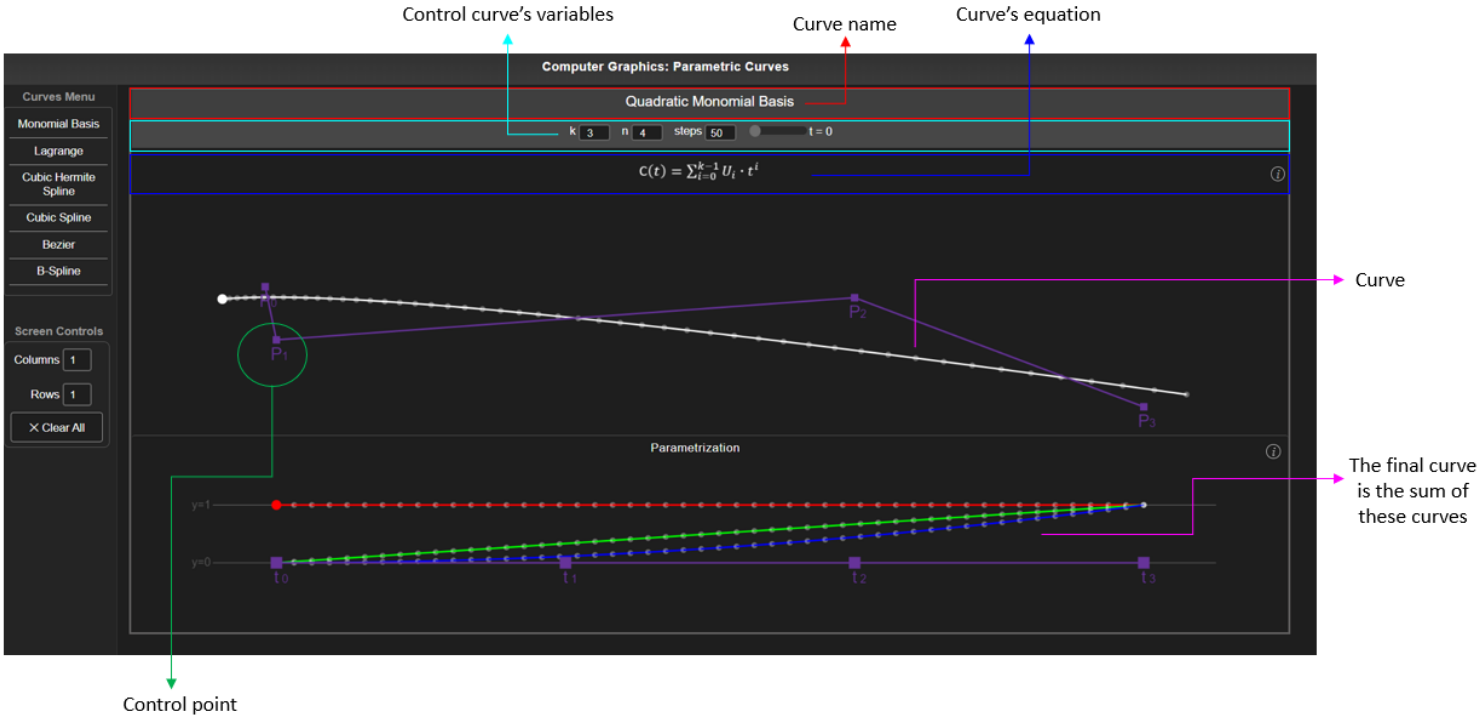
1. Opening Page



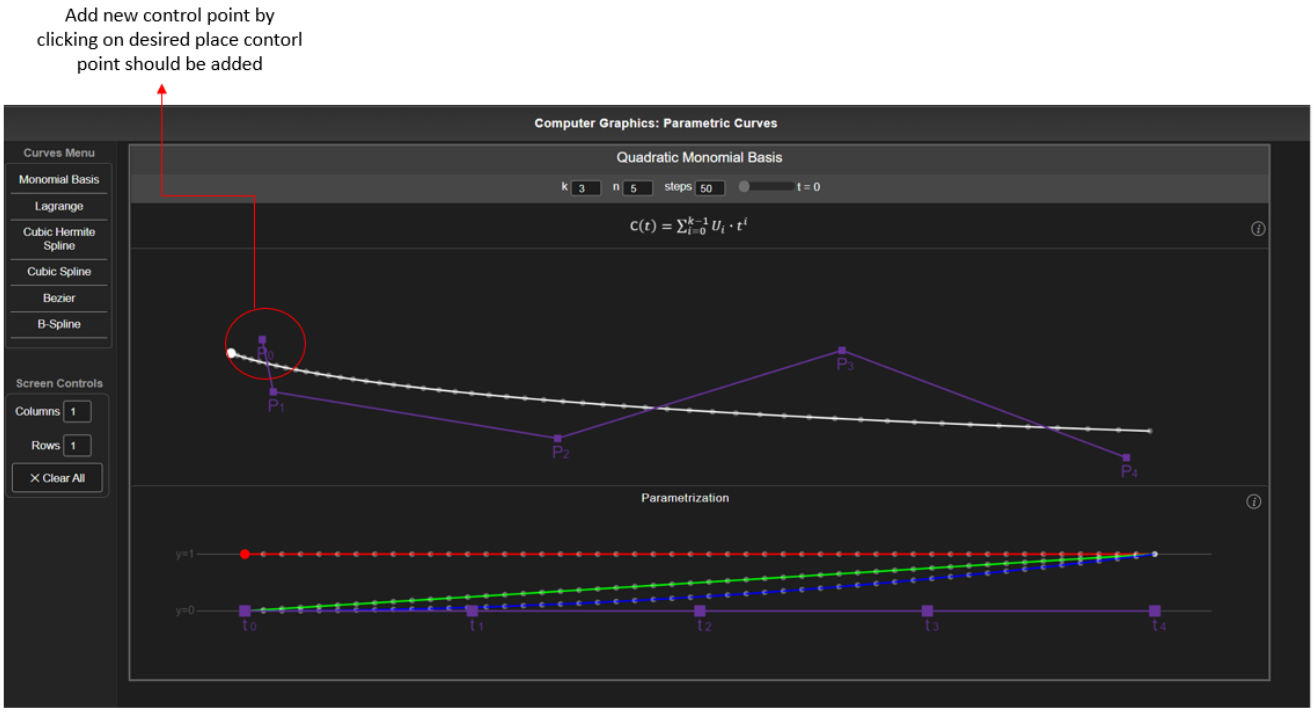
2. Select Curve - the user can choose a curve from the curve menu and click it or to drag and drop it to the main box.




Click chosen curve in menu
Computer Graphics: Parametric Curves
Curves Menu
Monomial Basis
Lagrange
Cubic Hermite Spline
Cubic Spline
Bezier
B-Spline
Drag and Drop
Quadratic Monomial Basis
k 3 n 4 steps 50 t = 0
$C(t) = \sum_{i=0}^{k-1} U_i \cdot t^i$
Screen Controls
Columns 1
Rows 1
✕ Clear All
Parametrization
Chosen curve will be showed in the main box

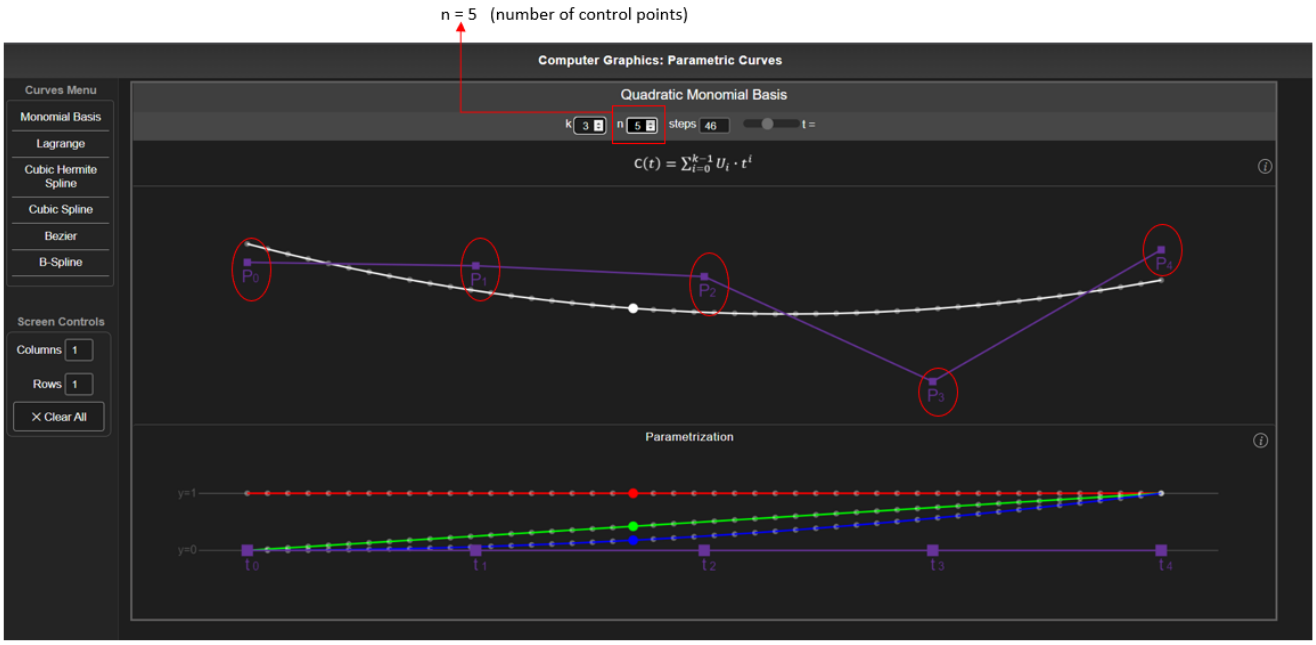

Control curve's variables
Curve name
Curve's equation
Computer Graphics: Parametric Curves
Curves Menu
Monomial Basis
Lagrange
Cubic Hermite Spline
Cubic Spline
Bezier
B-Spline
Quadratic Monomial Basis
k 3 n 4 steps 50 t = 0
$C(t) = \sum_{i=0}^{k-1} U_i \cdot t^i$
Screen Controls
Columns 1
Rows 1
✕ Clear All
Parametrization
Curve
The final curve is the sum of these curves
Control point

3. Play with control points – move control points by dragging them, add control point by clicking on the desired playce on screen it should be added, delete a control point by clicking on it

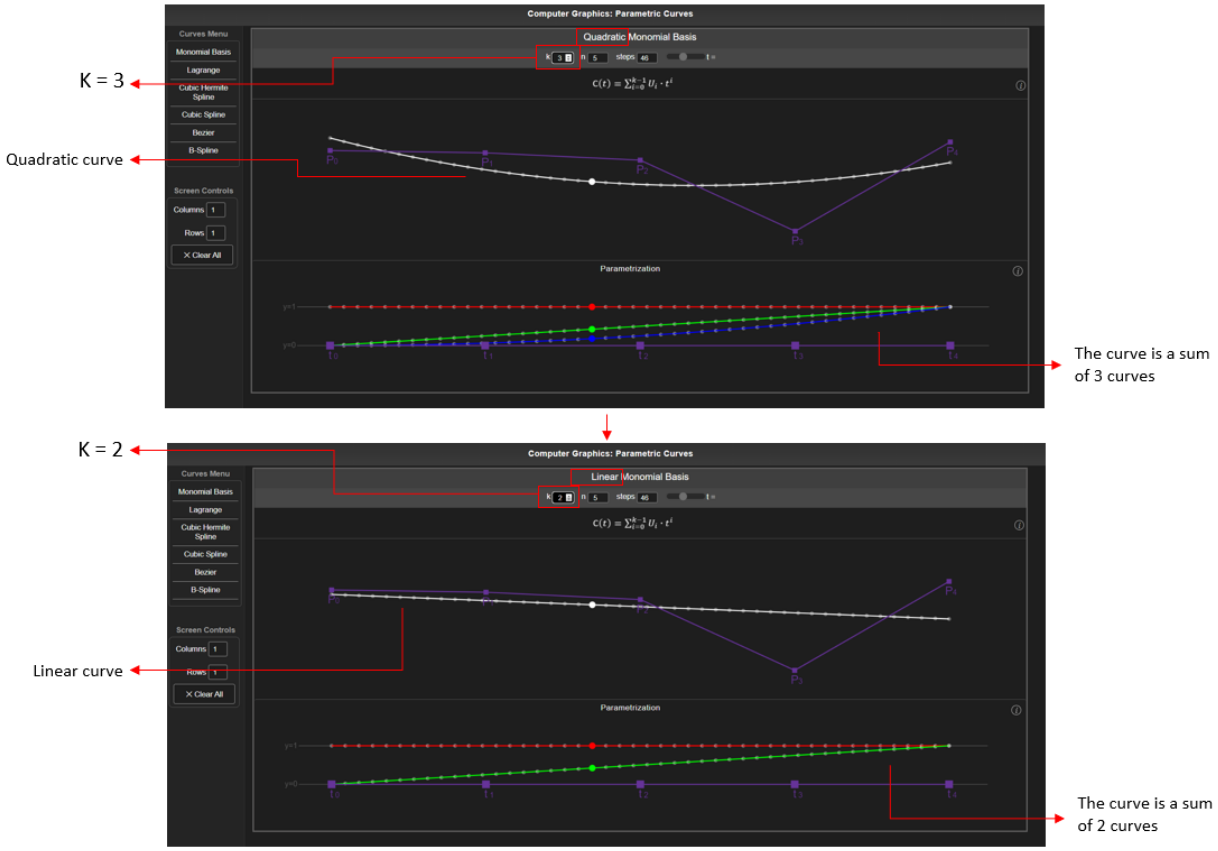


4. Control curve's equation variables:

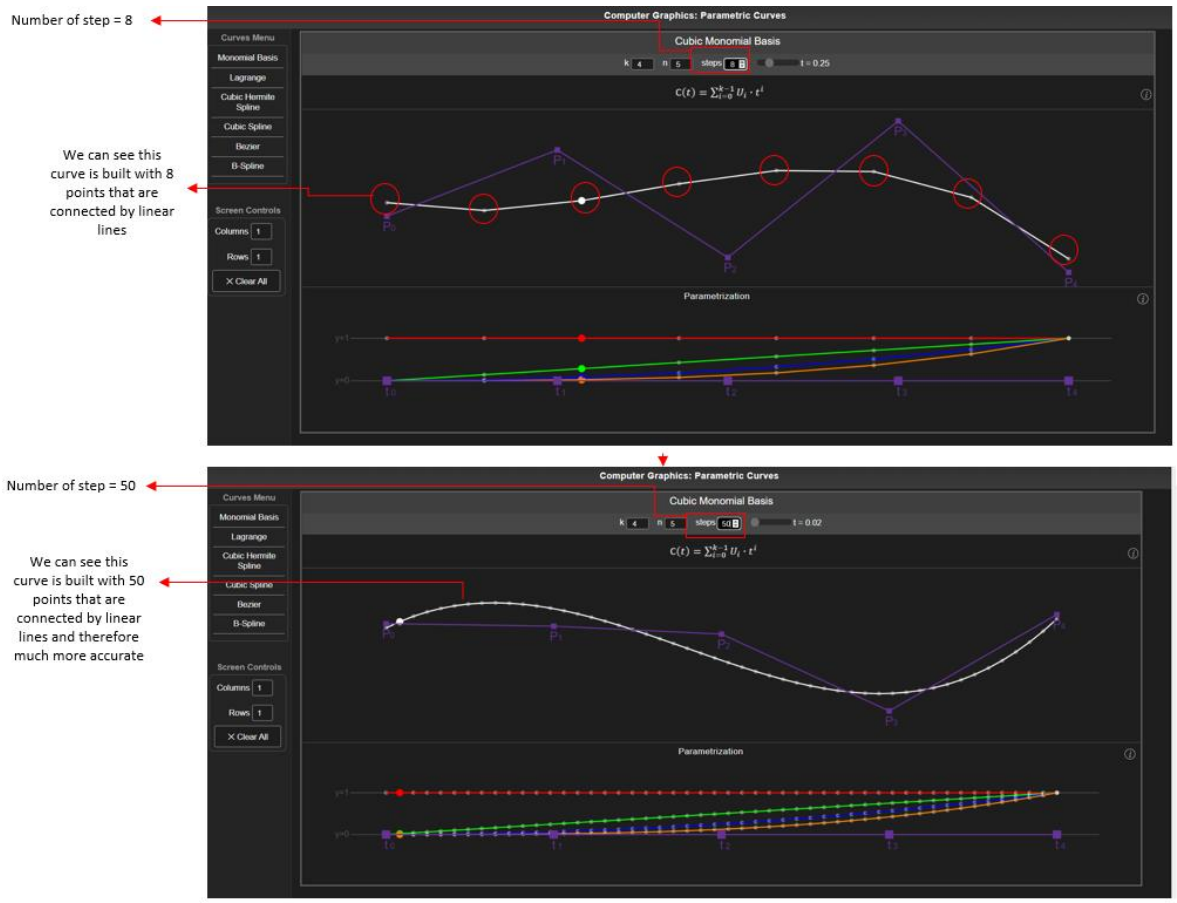
- n: number of control points



- k: Control curve's order, each curve's order is set differently. Monomial basis curve and B-Spline's order is set with a seperate variable k. Bezier and Lagrange's order is determined by n (number of control points) and for splines I enabled to view only cubic splines and cubic hermite splines.

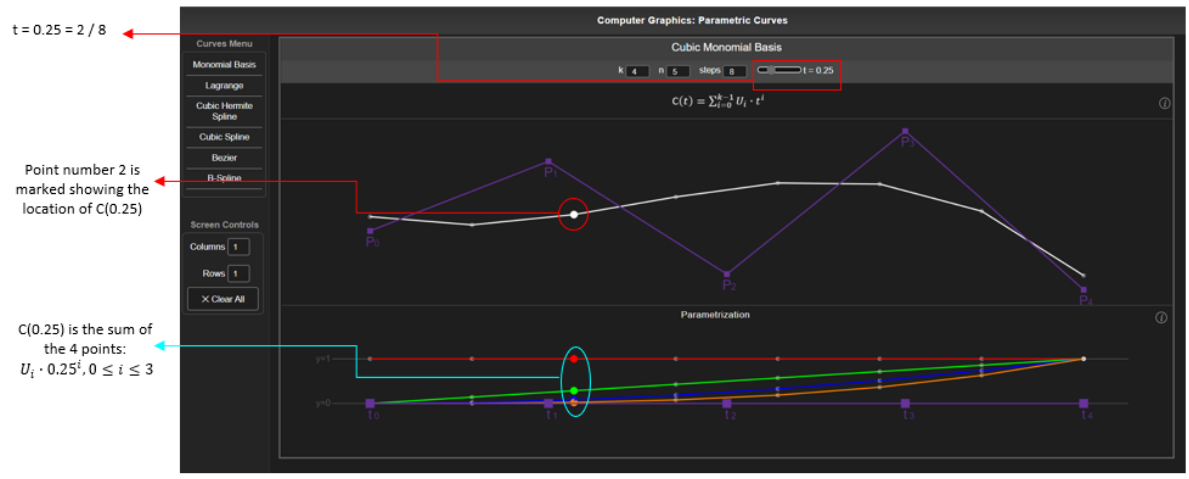


- steps: number of points to draw to make curve. The program generates for each stop a point on the screen and the draw linea lines between all points. The larger the number of steps the smoother the curve will be



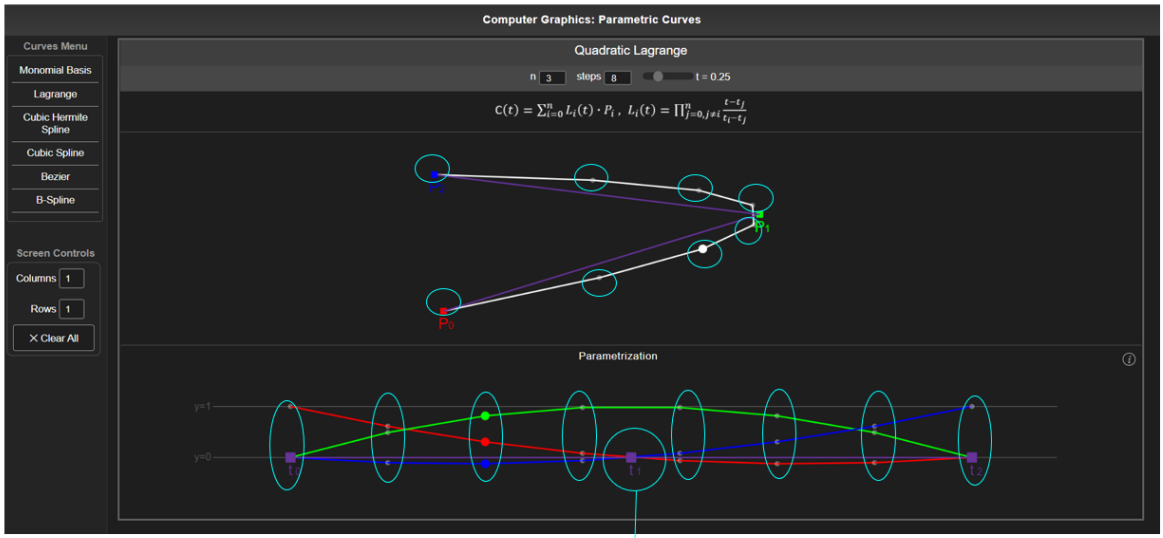
- t: show a specific point

$$t = \frac{step}{steps\#}, \quad 0 = t_0 \le t_1 \le \cdots \le t_i \le \cdots \le t_{n-1} = 1$$



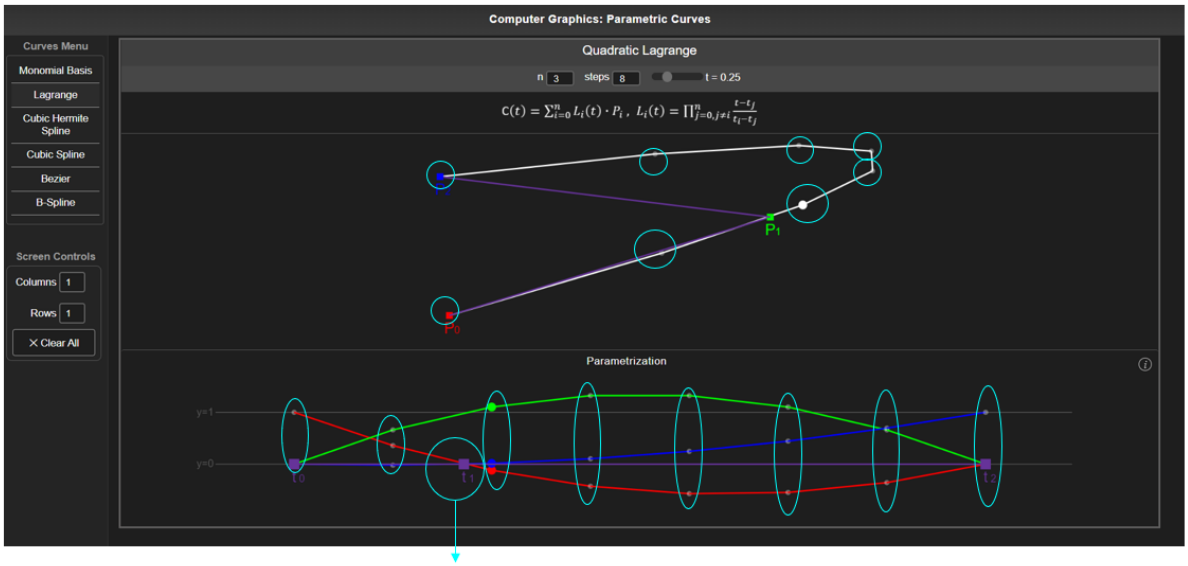
- Parametrization – change the number of points that build the curve between every two control points.

As seen in the example below, initially the points are distributed equally, between P_0 and P_1 there are 4/8 points and between P_1 and P_2 there are 4/8 points. In the parametrization section t_1 is exactly in the middle between t_0 and t_2 and there are 4 points on each side therefore will be 4 points between P_0 and P_1 and 4 points between P_1 and P_2.



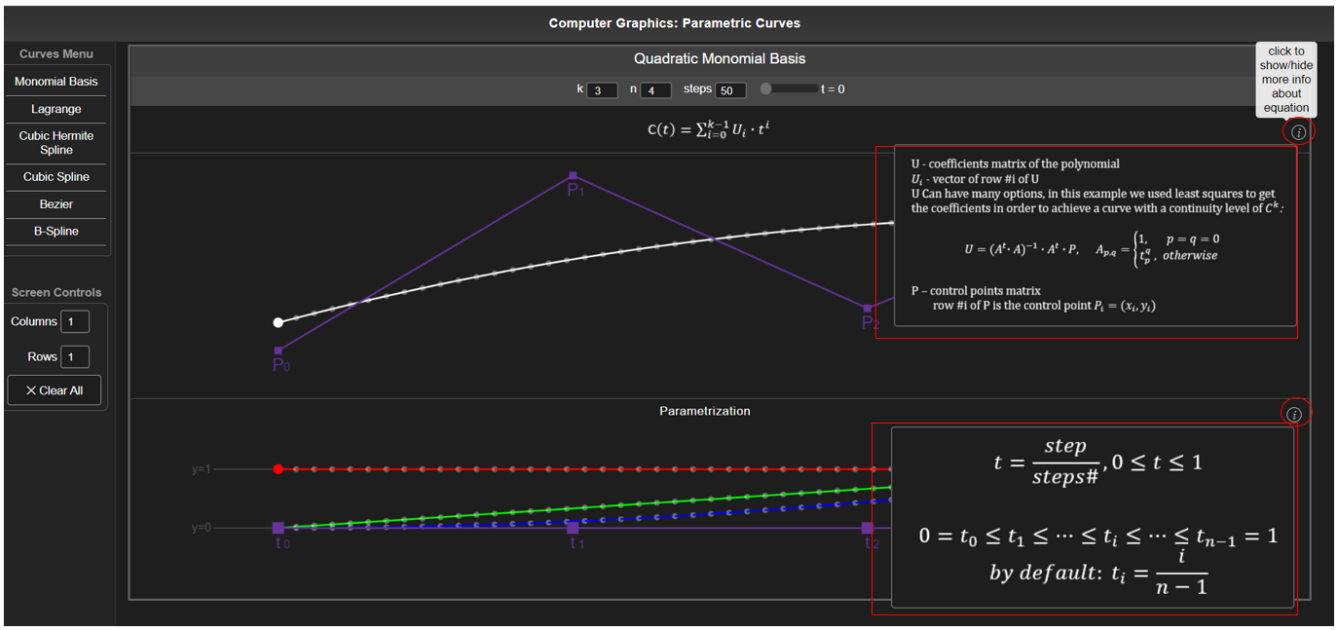
t_1 *is exactly in the middle between* t_0 *and* t_2, and there are 4 points in each secation

After moving t_1 in the parametrization section the points are distributed differently. In the example t_1 was moved closer to t_0 so there will be only 2 points between t_1 and t_0 and 6 points between t_1 and t_2 so therefore there will be 2 points between P_0 and P_1 and 6 points between P_1 and P_0.



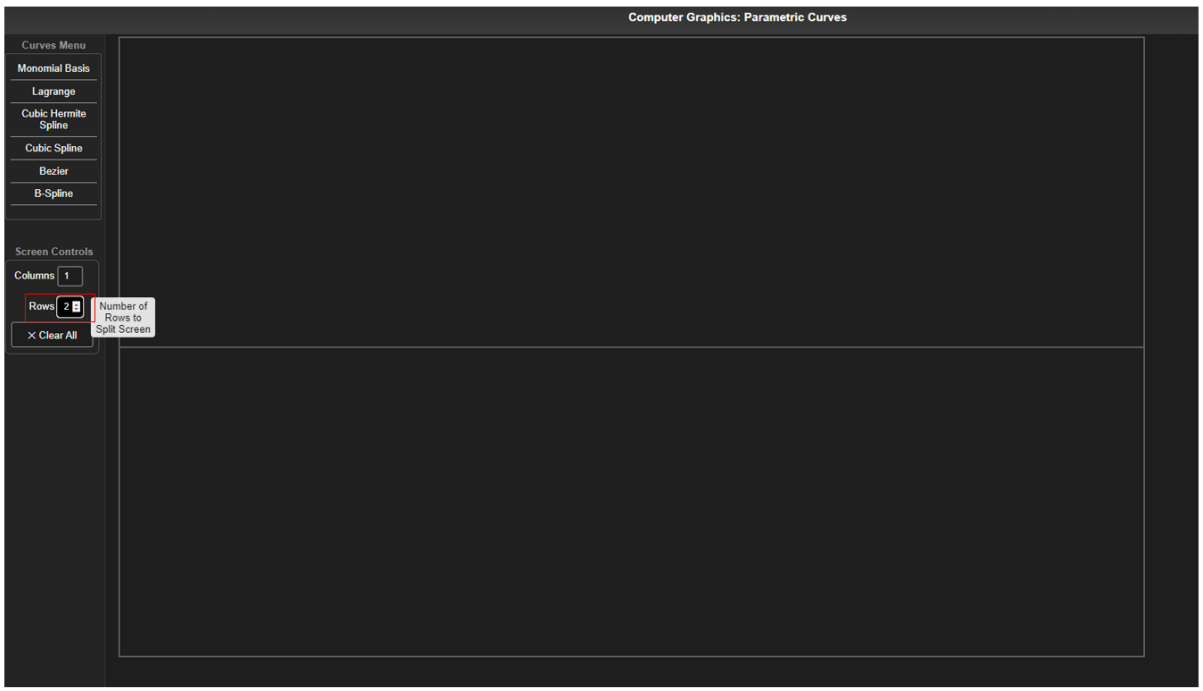
$moved\ t_1\ closer\ to\ t_0$

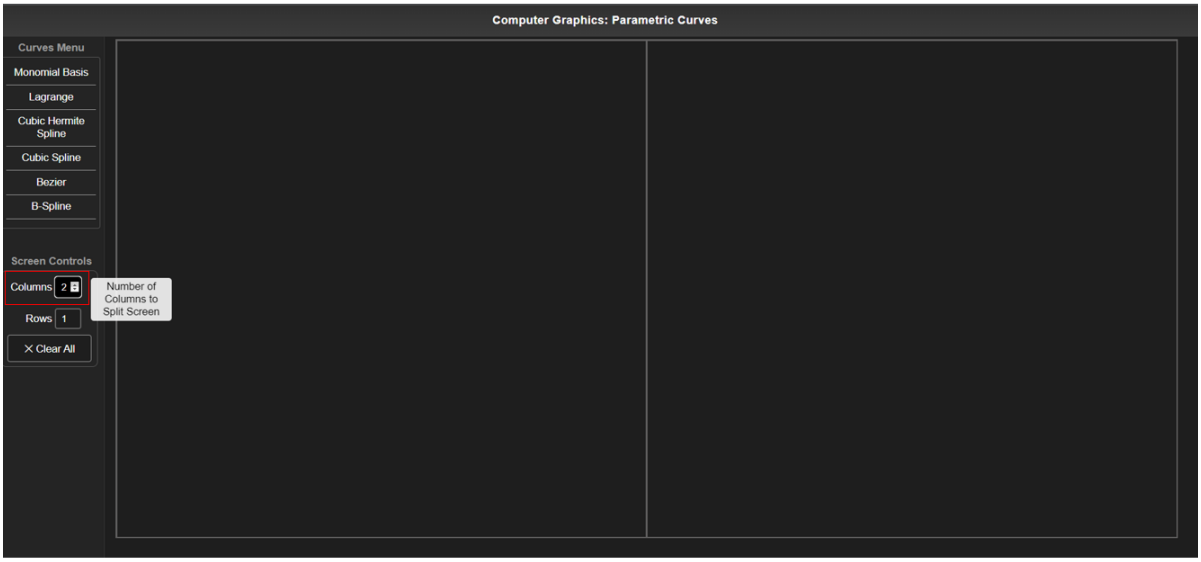
5. Learn more information about the curve's equation and parametrization by clicking the information button



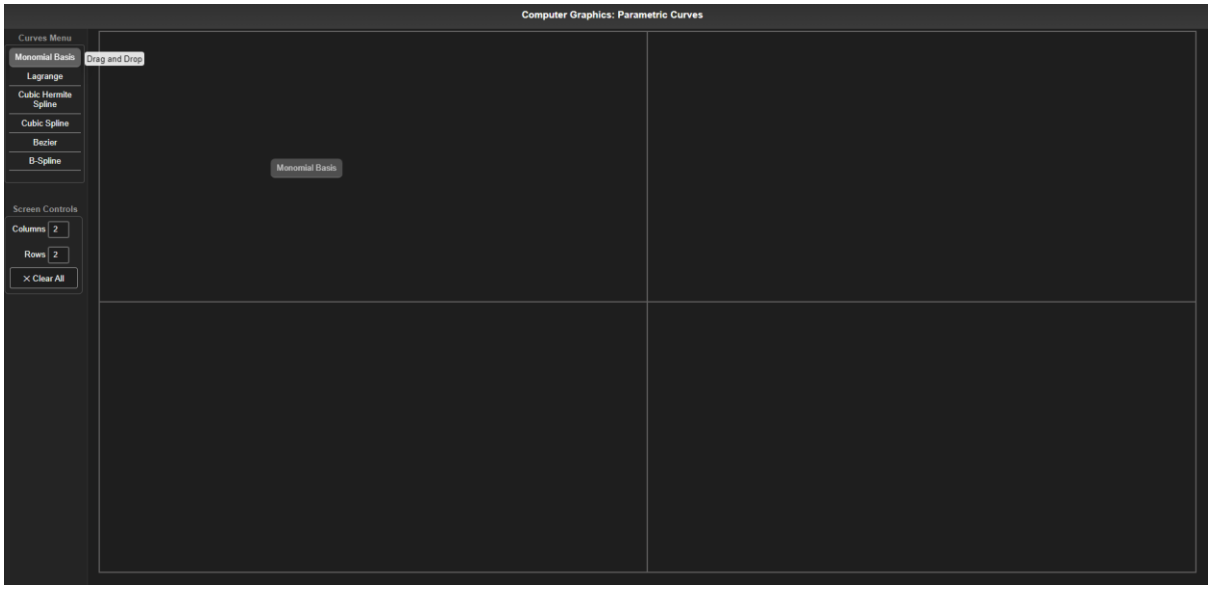
6. Split the screen to show multiple curves at once and then drag and drop a curve from the curve menu to the desired section of the screen

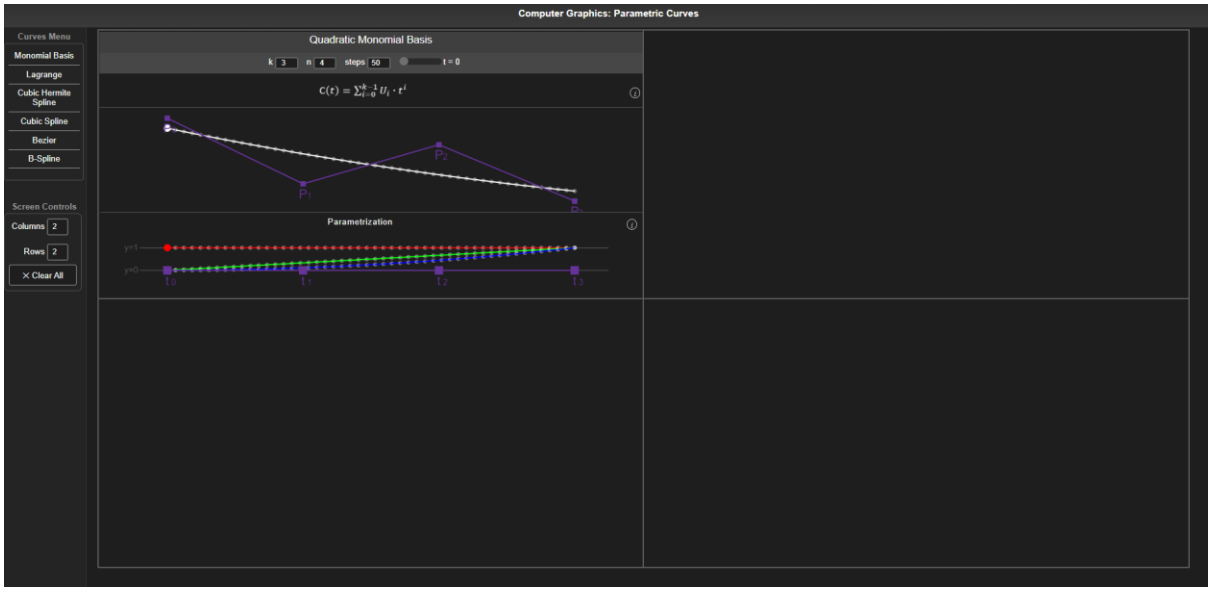
Change the number of rows and cloumns to split the screen.



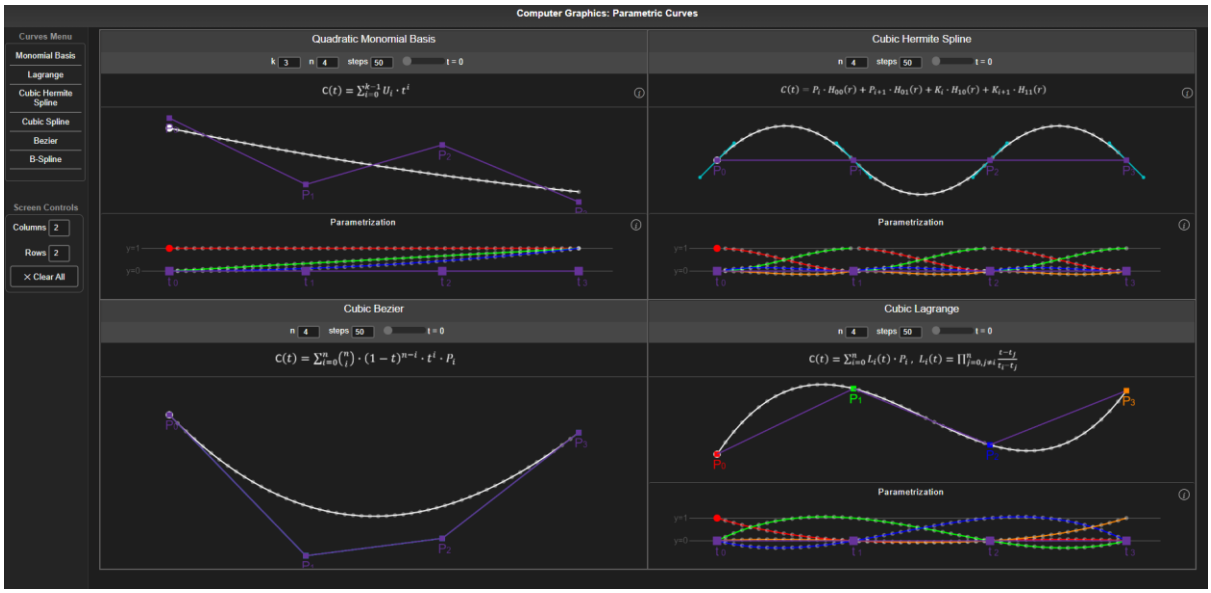


drag and drop a curve from the curve menu to the desired section of the screen at the example we are trying to put monomial basis curve to the top left scrtion of the screen

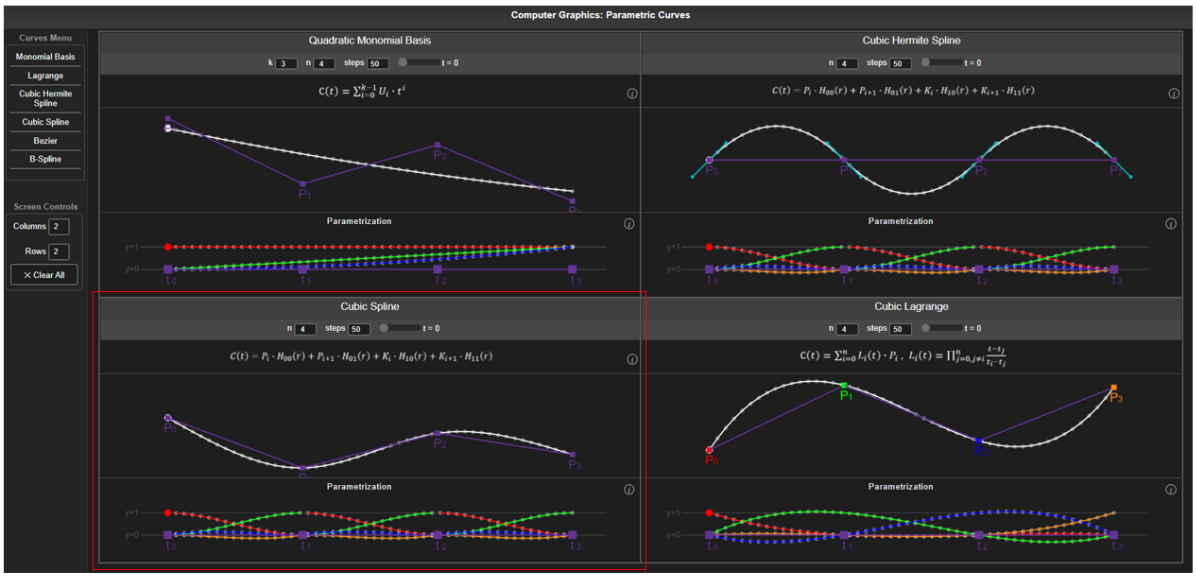




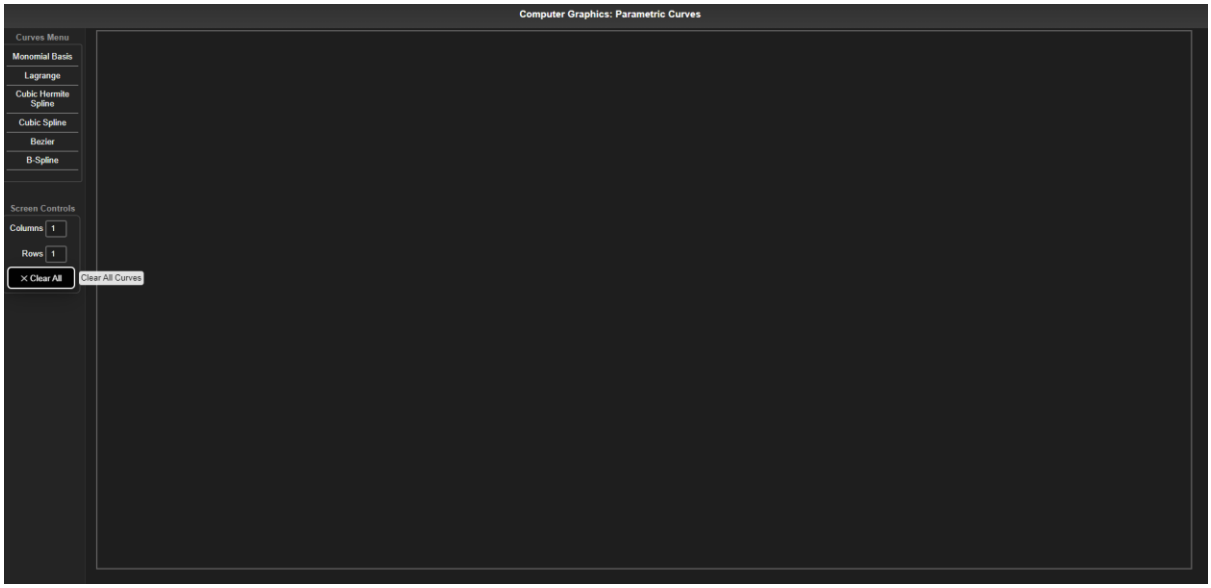
All the screen divides filled with curves:



It is also possible to drop a new curve on a square that already has a curve in it and by that replacing the old content. In the example Bezier curve was replaced with cubic spline.



7. Use “Clear All” button to clear all curves from screen and go back to starting point.



Technologies and Platforms

- **HTML**

HTML is short for HyperText Markup Language which is the standard markup language for documents designed to be displayed in a web browser and describes the structure of a web page. The HTML elements are the building blocks of HTML pages. The elements are delineated by tags, written using angle brackets for example <img></img> provides an image element. The curves are displayed in a <canvas> element. Browsers do not display the HTML tags, but use them to interpret the content of the page[1].
The main page of the project's website is written in HTML. It is the page that provides the website's layout and embeds the rest of code sections in CSS and JavaScript.

- **CSS**

CSS is short for Cascading Style Sheets and is a style sheet language used for describing the presentation of a document written in a markup language such as HTML and other markup languages. It is also a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.
Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods [2].

- **JavaScript**

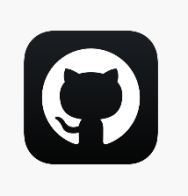
JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.
JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event.
It can function as both a procedural and an object oriented language. Objects are created programmatically, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in

compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint (or prototype) for creating similar objects.
JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation, object introspection, and source code recovery (JavaScript programs can decompile function bodies back into their source text)[3].
Most of the projects code was written in JavaScript. Most important functions that were done in JavaScript are the dynmaic splitting of the screen and the drawing of the curves on the screen.

- **Visual Studio Code**

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. It has features that support debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git [4]. It was the working envoirment that I chose to write the code of the project.

- **Github**



GitHub is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, continuous integration and wikis for every project [5].
I used git for my project to create backups to the layers of changes and in order to see clearly what were the changes that I made each time.

- **Chrome Inspect Tool**

A built in tool in Google Chrome web browser that is designed to help web developers check and edit the HTML, CSS and JS of the page displayed in the browser. Also allows to use a console that can get logging from the code and show errors. Similar tools are availabe in other browsers [6]. This tool was very beneficial to test and debug my project.

The Development Process

In order to create the website, we separated our development process into 5 main parts:

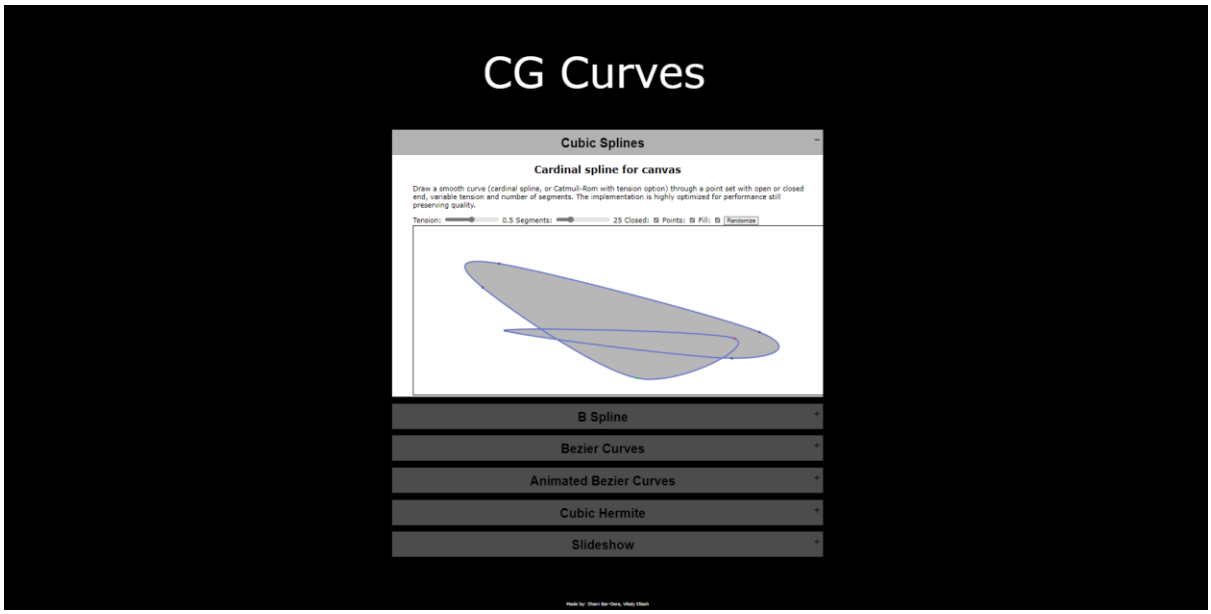
1. Designing the layout
2. Finding a website skeleton
3. Merging skeleton with initial layout
4. Adding the different curves algorithms to the code
5. Improving the UI

I will now talk about each part, my thought process, challenges and resolutions from each part.

Designing the Layout

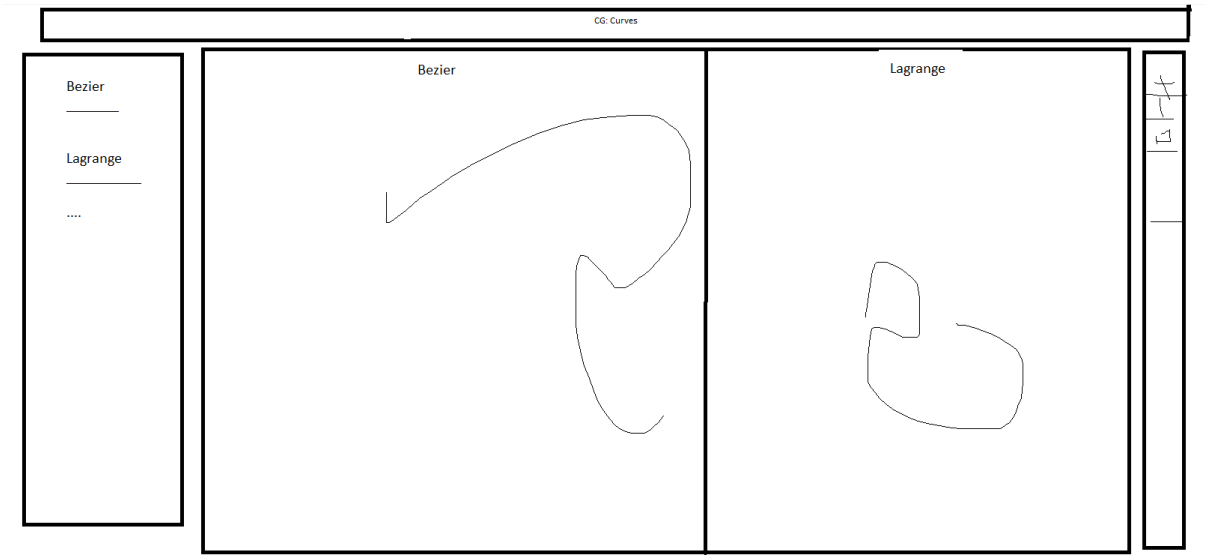
Firstly I had to learn how to build a website. I found w3school [7] to have very helpful tutorials. Then I wanted to decide how the website will look like. I was considering different possible layouts.

The first offer was this:



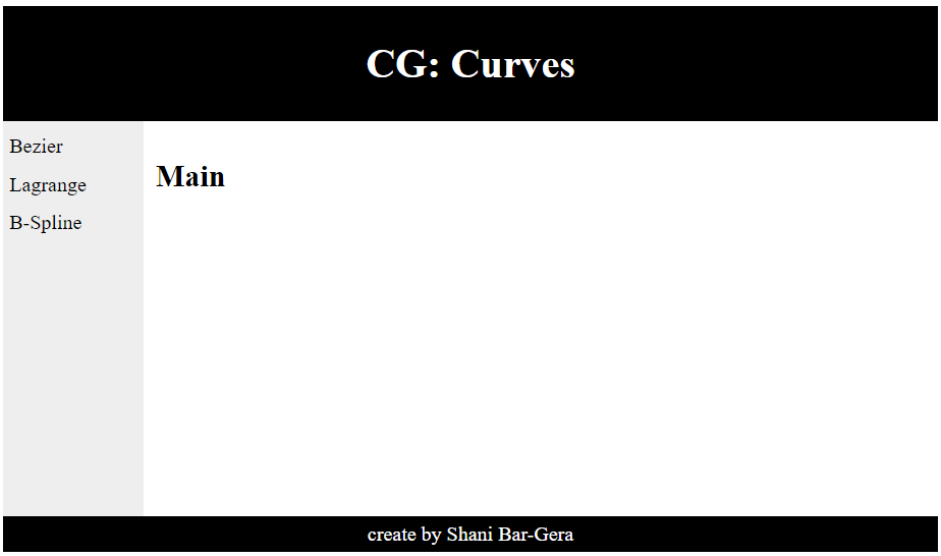
In this option I planed to have dropdown windows that each will have a curve to play with. Making this design made me learn a lot about using HTML and CSS and review the various websites that display curves and how they work. I learned about HTML canvas element that is used to display the curves in all the different websites.

But after discussing it with my supervisor we decided to go for something that looked like this:



The new design was intended to look more like a graphing tool and less like a slideshow, maximized the browser's space better to display the curves on most of it and allowed to split the screen in order compare different curves with one another.

I began learning how to create a basic layout using HTML. I learned how to create the header, a side navigation pane, a footer, and the main section. Later I added buttons to the navigation panes for the different curves. I also experimented with the CSS files trying to decide what will be the styling of my website. It looked something like this:



After creating a basic layout, I tried to understand how to add a split screen option. At first, I thought to dynamically create s in my main section and set them with a specific size and location. For that I had to start learning some JavaScript as well. After searching for examples online I learned about table element that fulfilled my requirements.

I created my first experiment with it.

Initial layout:

Click the button to add a new row at the first position of the table and then add cells and content.

Row1 cell1	Row1 cell2
Row2 cell1	Row2 cell2
Row3 cell1	Row3 cell2

Try it

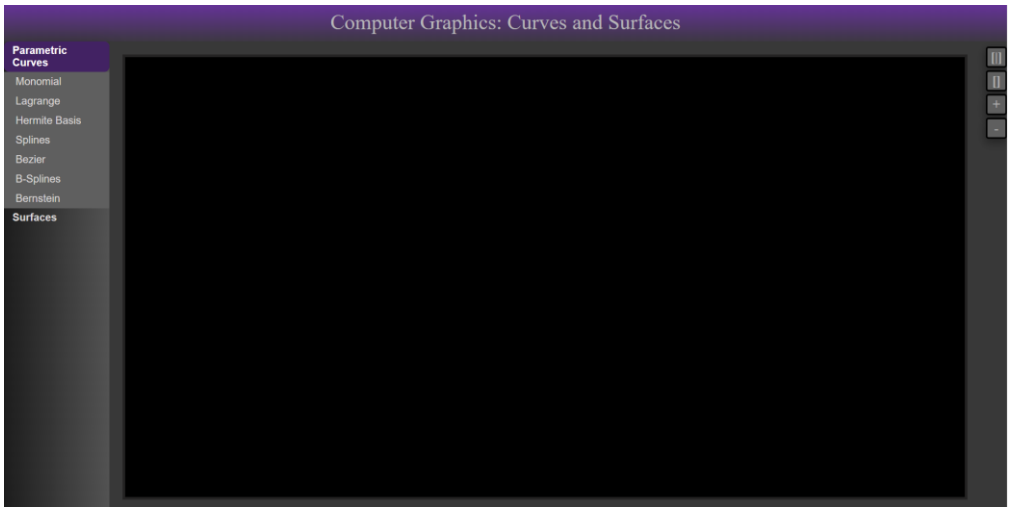
After Clicking the button:

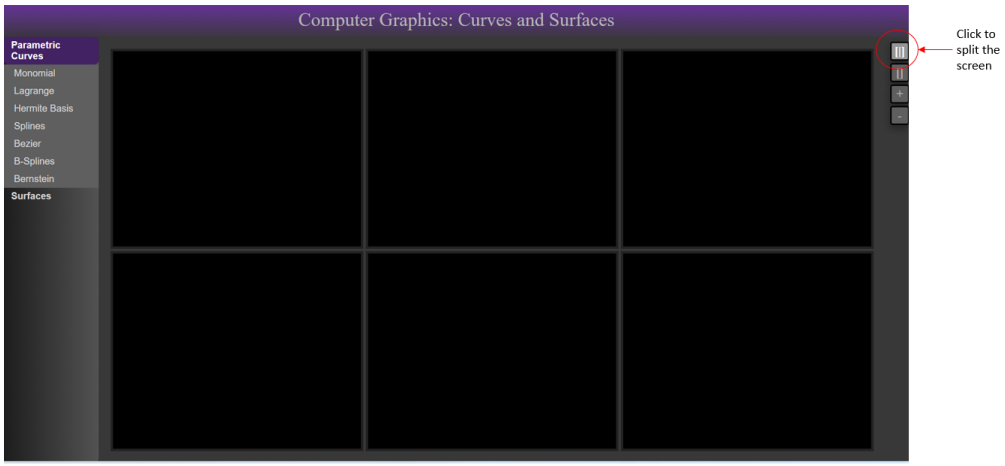
Click the button to add a new row at the first position of the table and then add cells and content.

NEW CELL1	NEW CELL2
Row1 cell1	Row1 cell2
Row2 cell1	Row2 cell2
Row3 cell1	Row3 cell2

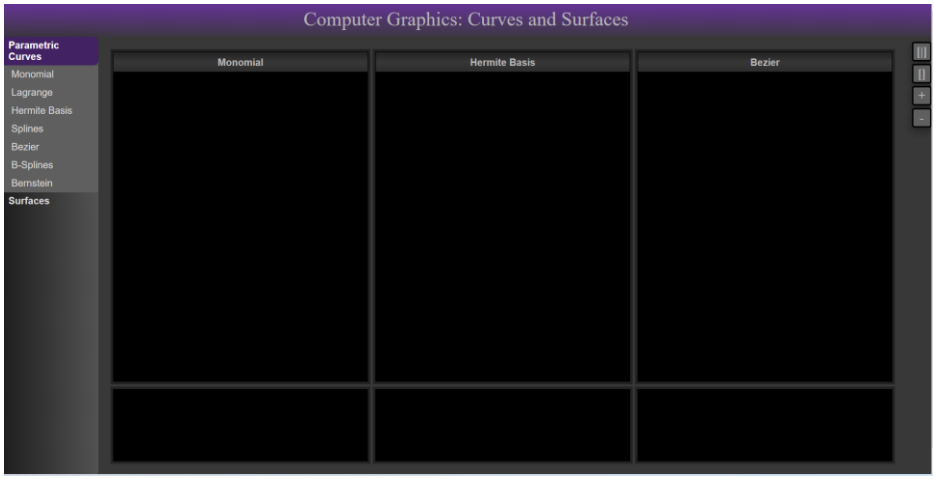
Try it

After it worked, I tried to add it to the main div of my layout and change the styling. This is how I set the initial table to look like:





After I succeeded to add the table, I wanted to be able to drag and drop an item from the curve menu to the desired table cell. In order to do that I had to learn how to make items draggable and how to listen to events in JavaScript. After many attempts I finally succeeded to enable the dragging and dropping option and transferring the data from the button's text to the table cell. I got something like this:



This was acceptable for the meantime, but I was unsatisfied with how the table's cells fit their size to the content and not the other way around. After spending a lot of time trying to understand how to stop this at the end, I dropped this and moved on to other things only to comeback to it later. In the end what that fixed this issue was adding an empty canvas element to the cells by default and replace it with a new one when a curve is being dropped.

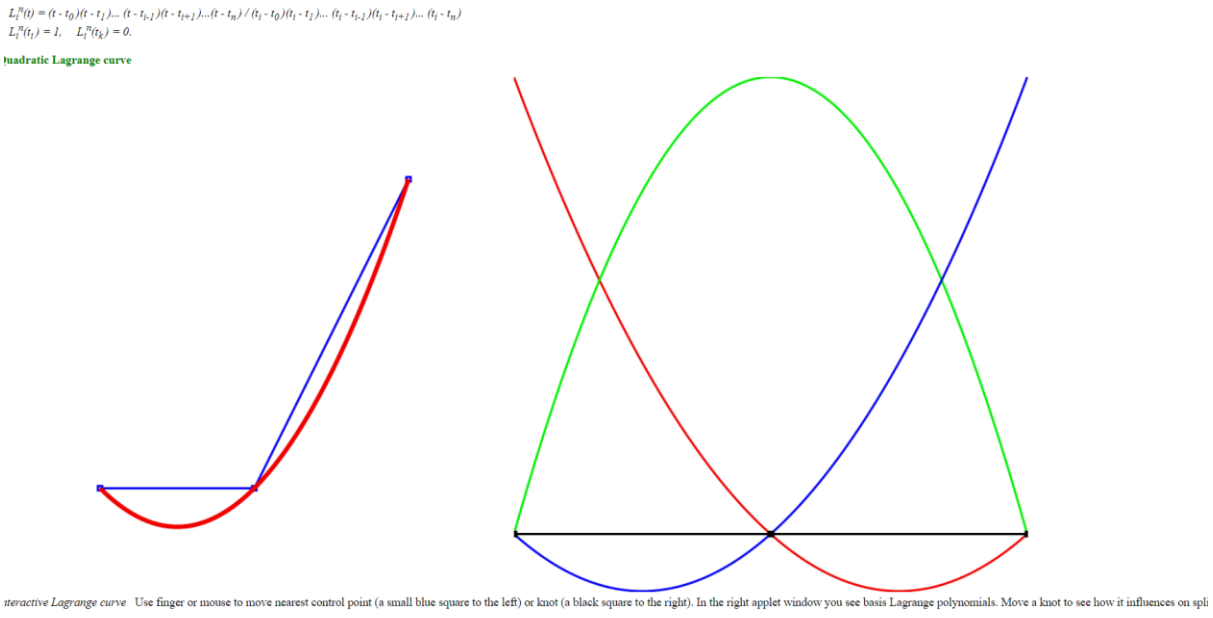
At this point I marked making the layout a success, learning in the process a lot about HTML, CSS and the JavaScript table element and how to dynamically change it's grid using JavaScript. I now decided to try and understand how to use the canvas element and how to display the curve in each table cell.

Finding a Website Skeleton

I wanted to get a uniform template on how I would like to display all the different curves that I could drop in my table cells. For that I reviewed many different websites the show parametric curves in different ways in hope I could use some of them as a skeleton for my website.

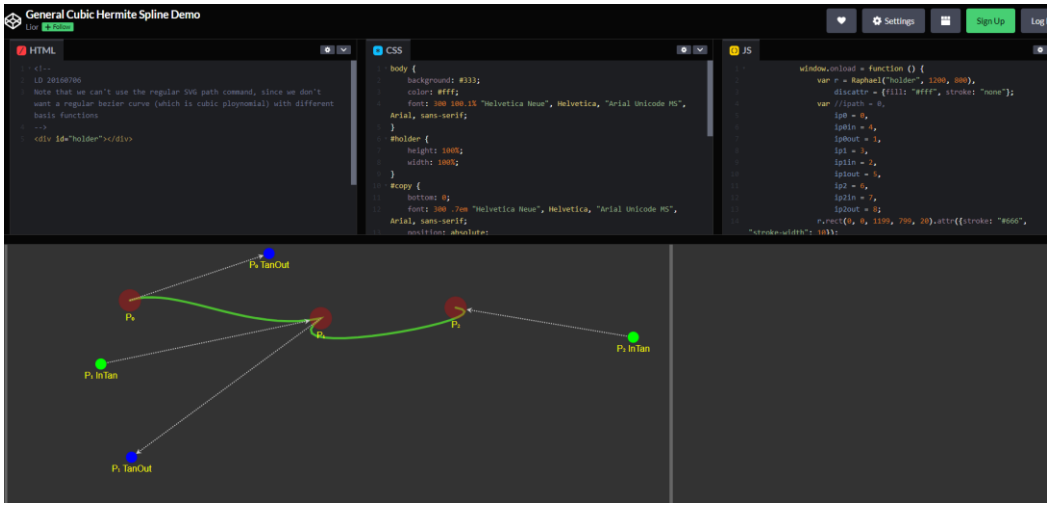
1st Website: Lagrange Interpolation [8]

The first website is the website currently used to display the Lagrange curve in the computer graphics course. It looks something like this:



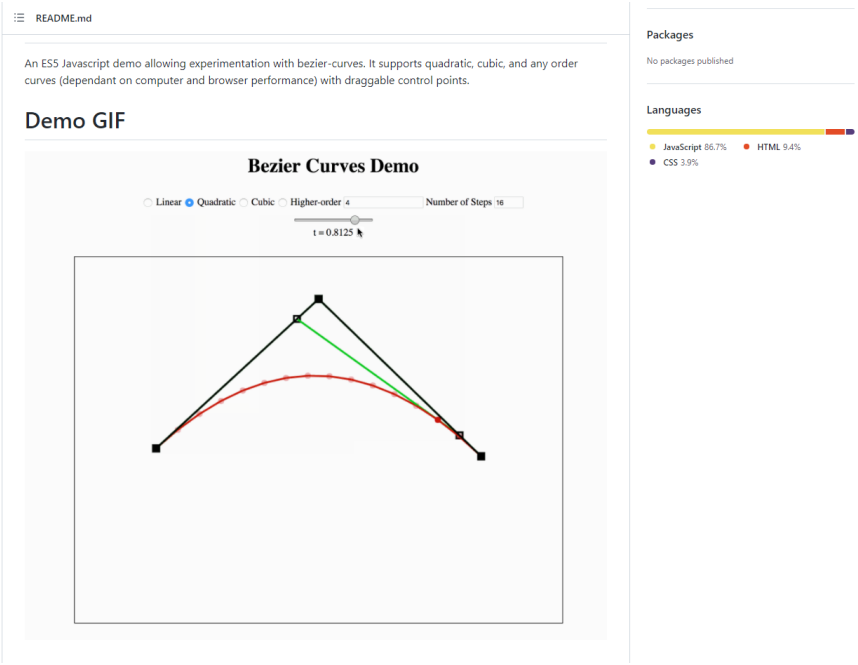
I was not very pleased with this website since its code was not very organized and very hard to understand. In addition, there were many glitches while playing with the control points, the experience was not very smooth. Also, I did not like how the information was organized. It was very hard to understand the color coding, what information refers to which curve, etc. And lastly, it's layout was very different from I was trying to do.
I ended up not using this website's code as a skeleton, but I did get some inspiration from it on how to display the parametrization curves in my website.

2nd Website: Cubic Hermite Spline Demo [9]



I had some issues with running the code of this website outside of the demo and in addition also had a hard time to understand how the tangents affect the curve from moving them. I figured that if I had a hard time probably other students would as well, so I decided not to go with this display. But I did get the inspiration from this website to use labels for my curves control points as well.

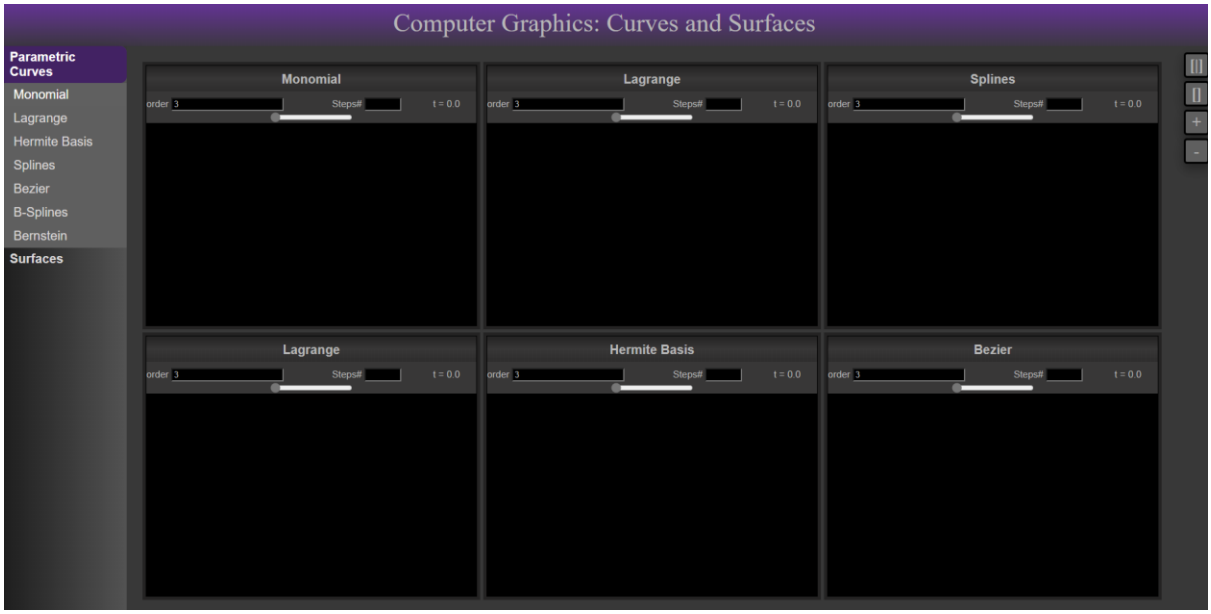
3rd Website: Bezier Demo [10]



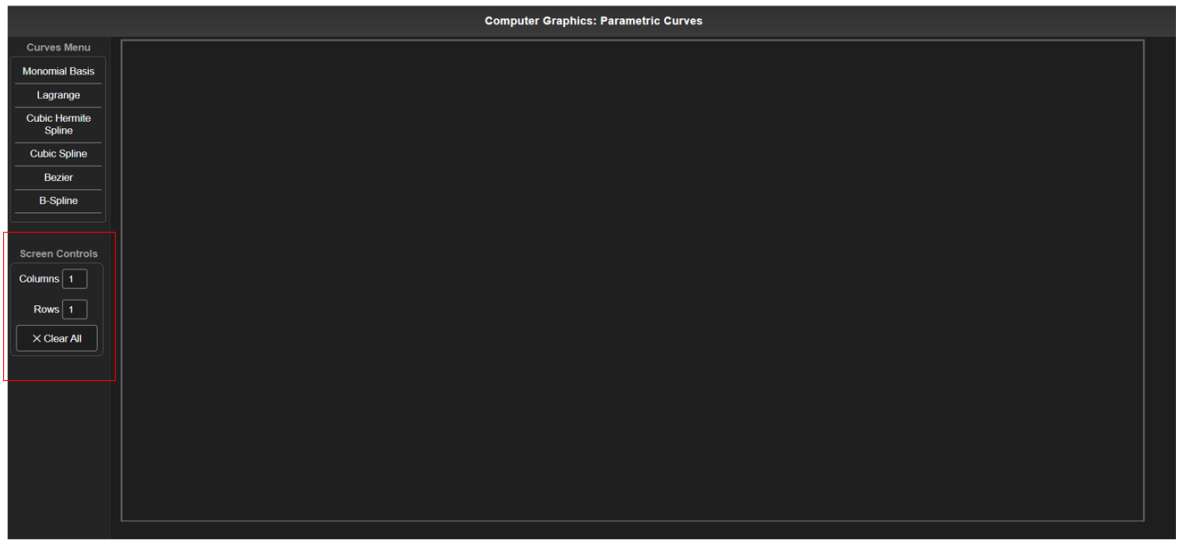
I was very pleased with this website, the git code was running smoothly, the code was very organized and easy to understand with a lot of documentation and generic functions. In addition, its layout was similar to what I was trying to create for my website. Also, I liked that it drew the dots showing me how the curve was drawn. I felt like this was very helpful in understanding how the curve was built. I decided to try and merge this code with my initial layout in hope that if this succeed, I could easily add more algorithms of the other curves.

Merging Skeleton with Initial Layout

Firstly, I wanted to add inputs to my layout similar to the ones in the skeleton I chose so the Bezier app of the existing website could get its inputs from my layout. After learning how HTML inputs work and how to create it in JavaScript (I wanted to create in JavaScript since I wanted them to be created dynamically depends on the given curve). After learning about inputs and attempting to implement them my website looked like this:

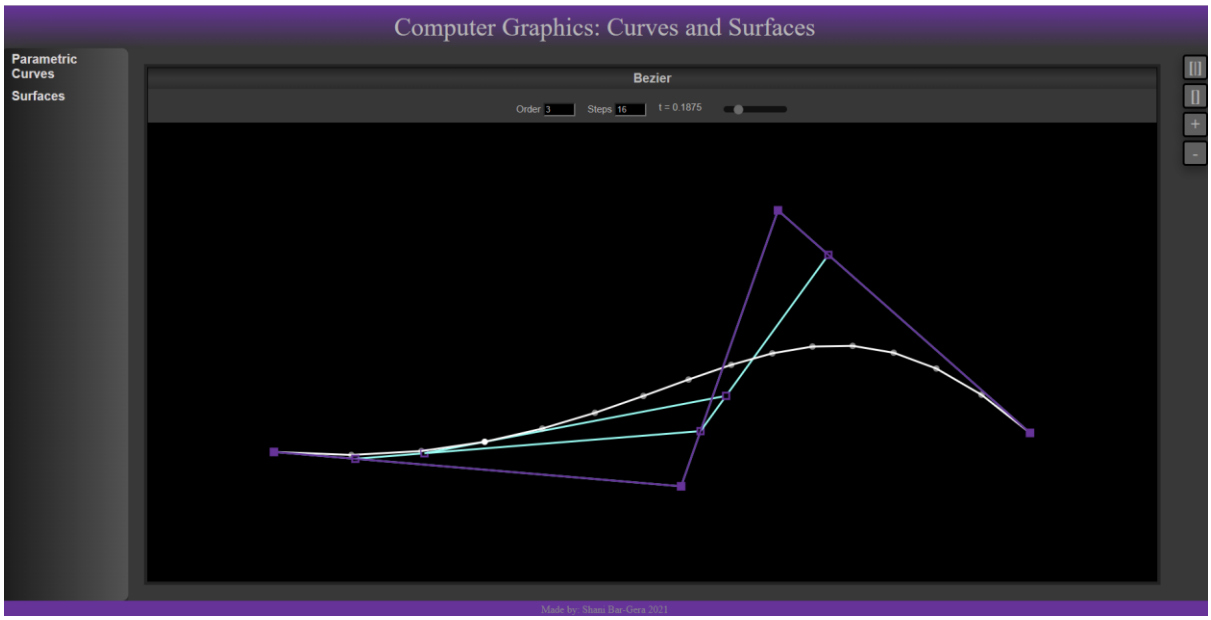


I felt like this was good enough in the meantime and will format it later to look better. My next step was to try and add to each table cell a canvas with a unique ID so the Bezier app I got from GitHub could write into it. In the original website the canvas was created through the HTML file and was with a fixed width and height. For my needs this did not fit since I wanted to create a split screen option, so I wanted canvases that fit their size to the table cells. This was an issue I had for a very long time and had a hard time to fit the canvas to the table cells that change sizes. The problem was the cells changed their size dynamically to fit to the content. I wanted the other way around; I wanted the content to fit to the table's cells. In addition, the I made the input bar and header of the curve fit to content as well which also caused issues in understanding the canvas' desired size. After a lot of issues, I decided to calculate manually the size I desire for the cells, set a fixed height for the curve's header and input bar and with that I was able to calculate the size I wanted for the canvas.
This still caused a small issue when trying to split the table dynamically when there is still content in it. I decided to solve this by removing from the user this option and created a screen control that clears the screen between table grid changes. The new menu looked like this:

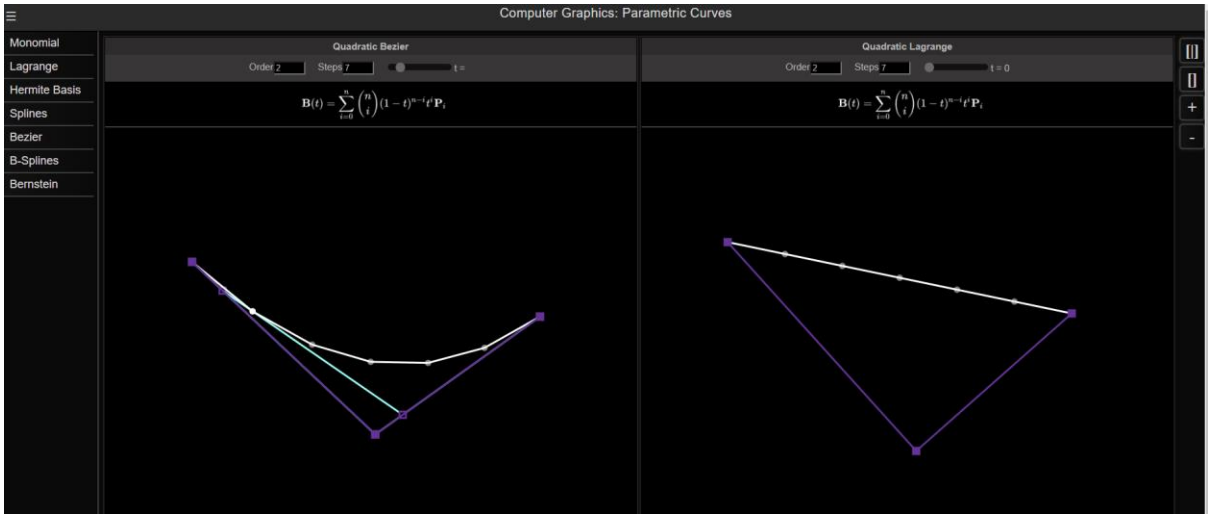


After being able to fit the canvas' to the table cell I began to try and connect the Bezier app to my canvas and inputs.

At the end of a long learning process on how the GitHub code worked I succeeded to portray it in my table cell:



After successfully showing the Bezier curve I wanted to see if I can add different curves to the existing app. I implemented an easy linear line curve that simply connect the first and last control point and tried by a click of a button from the curves menu to switch between the Bezier curve and the basic linear curve.
After succeeding this is what it looked like:



An issue I had with the inputs was to connect the correct input to the correct canvas. I solved this by using Ids for all my elements where the ID startes with the location of the cell its in for example “t_0_0_canvas” or “t_0_1_header”.

Creating the Curves

My next step was to go thorough each of the curves in the geometric modeling lecture [11], understand it better and implement its algorithm.

I reviewed various website and how they calculate their curves. At first, I used from a C-Spline demo website [12] use of functions. I later decided that this creates a messy code and preferred to use inheritance.

I created a base class called Curve that all the other curves will inherit from it. In each step a Curve class is called to interpolate its coordinate. The class has three crucial members:

- t – the input for the algorithm
- base – an array of points of each base curve at value t
- point – the interpolated point C(t)

The code iterates on the steps and for each step it calls the relevant curve class to interpolate the coordinates of the point in the specific step. The class’s constructor receives the control points, the ts of the parametrization, and the relative step (step / number of steps – 1). It calculates the t value for that curve (by default the t is just the relative step), the base curves and then interpolates the point C(t) with the calculated t and base curves.
In the end the application draws linear lines between all the interpolated points.

I will now explain how I created each one of the curves, excluding Bezier since I had that implemented from the demo I found and used as skeleton.

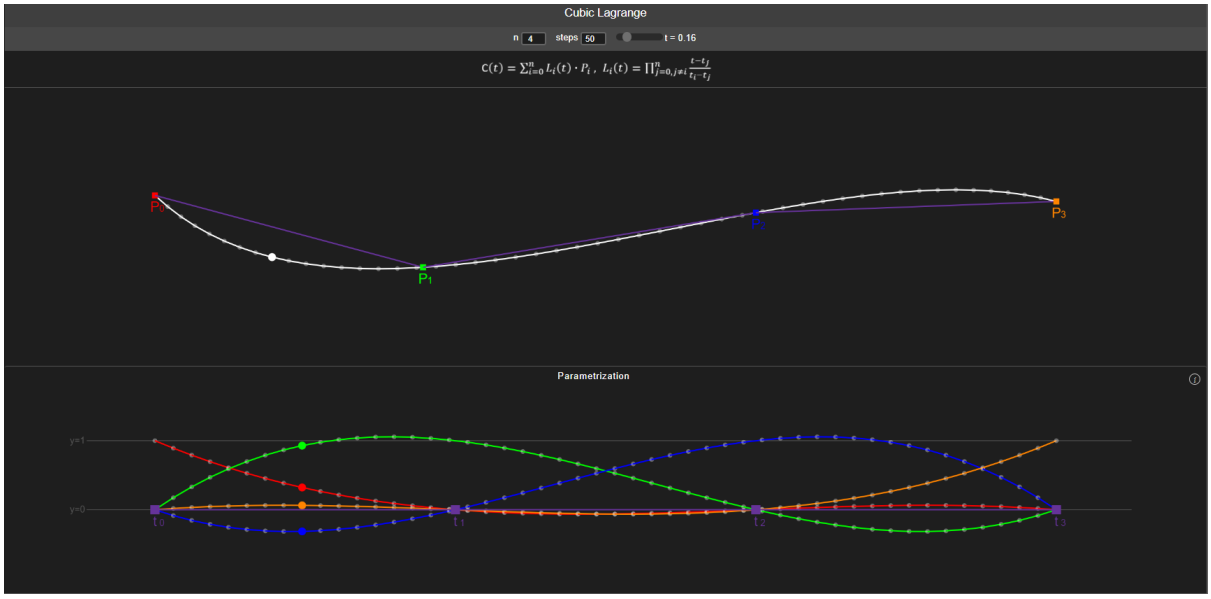
Lagrange Curve

I started with the easiest algorithm which was the Lagrange using this equation I got from the lecture[11]:

$C(t) = \sum_{i=0}^{n} L_i(t) \cdot P_i\ ,\ \ L_i(t) = \prod_{j=0, j\neq i}^{n} \frac{t-t_j}{t_i-t_j}$

For the members of the Lagrange class I set t as the default relative step/(steps# -1), for the base I calculated the $L_i(t)$ for all n is and then set the point to be C(t).

This is what it looked like:



In the picture above we can see for t=0.16 (8/50), n=4 in the main curve the white point marked and in the parametrization section 4 points marked on 4 different curves, each curve represents a specific L_i . It's important to mention that the order of the Lagrange curve is set by n so for n=4 this is cubic Lagrange.
In general the parametrization is :

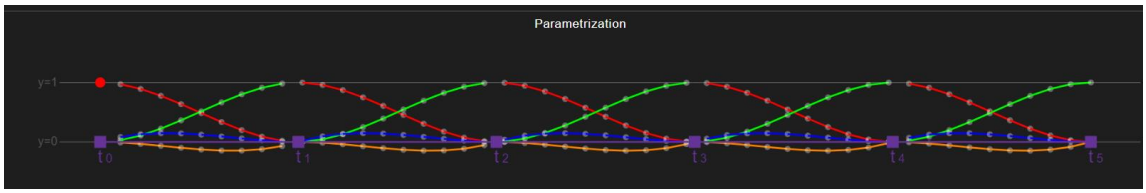
$$L_i(t_i) = 1,\ L_i(t_j) = 0 \Rightarrow C(t_i) = P_i$$
$$by\ default\colon\ t_i = \frac{i}{n-1},\ 0 \leq t \leq 1$$

I also colored the control points of the main curve with the color of the matching base curve in the parametrization section. In other words P_i and L_i are in the same color.

Cubic Spline

I got the base functions for the lecture[11] and created with them the base curves for the parametrization:

$$H_{00}(r) = r^2 \cdot (2 \cdot r - 3) + 1$$
$$H_{01}(r) = -r^2 \cdot (2 \cdot r - 3)$$
$$H_{10}(r) = r \cdot (r - 1)^2$$
$$H_{11}(r) = r^2 \cdot (r - 1)$$



There are 2 different ways to implement C-Spline interpolation with the base functions, a non-parametric approach and a parametric approach.
In the non-parametric approach, each interpolated point is of the form p = (x, y(x)). This approach was my first attempt to implement C-Spline interpolation and for it I used a code example I found online [12].

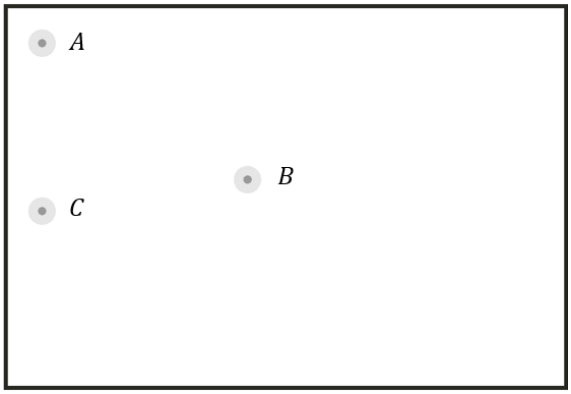
```
CSPL.evalSpline = function(x, xs, ys, ks)
{
   var i = 1;
   while(xs[i]<x) i++;

   var r = (x - xs[i-1]) / (xs[i] - xs[i-1]);

   var a =  ks[i-1]*(xs[i]-xs[i-1]) - (ys[i]-ys[i-1]);
   var b = -ks[i  ]*(xs[i]-xs[i-1]) + (ys[i]-ys[i-1]);

   var q = (1-r)*ys[i-1] + r*ys[i] + r*(1-r)*(a*(1-r)+b*t);
   return q;
}
```

```
CSPL.getNaturalKs = function(xs, ys, ks)    // in x values, in y values, out k values
{
   var n = xs.length-1;
   var A = CSPL._gaussJ.zerosMat(n+1, n+2);

   for(var i=1; i<n; i++)    // rows
   {
      A[i][i-1] = 1/(xs[i] - xs[i-1]);

      A[i][i  ] = 2 * (1/(xs[i] - xs[i-1]) + 1/(xs[i+1] - xs[i])) ;

      A[i][i+1] = 1/(xs[i+1] - xs[i]);

      A[i][n+1] = 3*   ( (ys[i]-ys[i-1])/ ((xs[i] - xs[i-1])*(xs[i] - xs[i-1]))
                     +  (ys[i+1]-ys[i])/ ((xs[i+1] - xs[i])*(xs[i+1] - xs[i])) );
   }
```

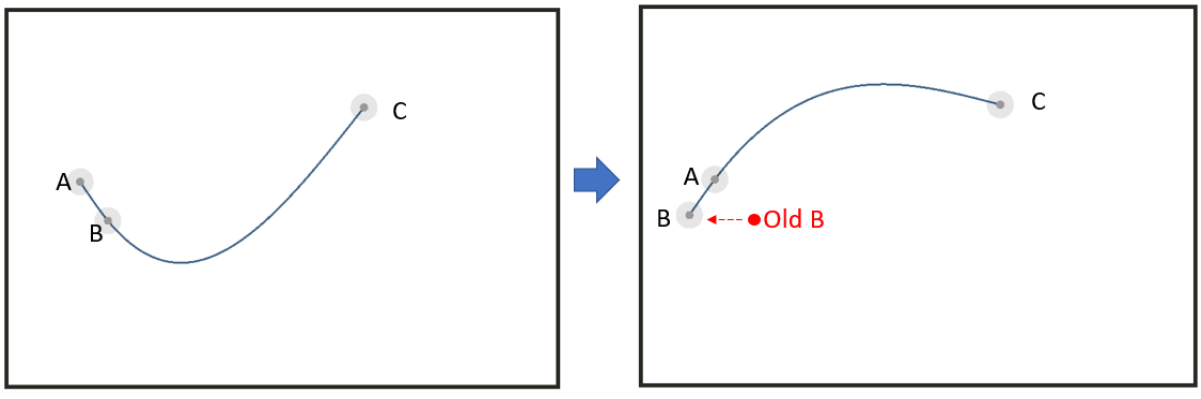
```
    A[0][0  ] = 2/(xs[1] - xs[0]);
    A[0][1  ] = 1/(xs[1] - xs[0]);
    A[0][n+1] = 3 * (ys[1] - ys[0]) / ((xs[1]-xs[0])*(xs[1]-xs[0]));

    A[n][n-1] = 1/(xs[n] - xs[n-1]);
    A[n][n  ] = 2/(xs[n] - xs[n-1]);
    A[n][n+1] = 3 * (ys[n] - ys[n-1]) / ((xs[n]-xs[n-1])*(xs[n]-xs[n-1]));

    CSPL._gaussJ.solve(A, ks);
}
```

In this implementation the control points are first ordered by their x coordinate. This creates a few restrictions on the curve. Equal x coordinates are not allowed; the connection between the control points depends on their x values and the role of x and y is not symmetrical. These restrictions cause the experience of playing with the curve's control points to be less smooth as seen in the following examples:

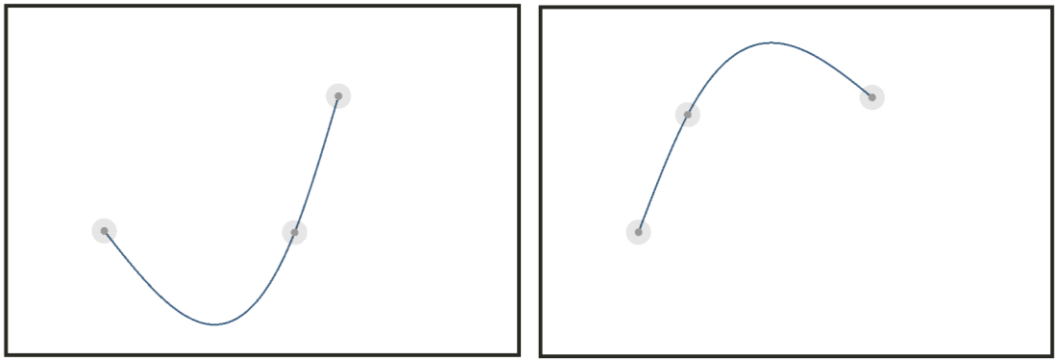
Example 1: the x of point A and C are equal - the curve is not drawn when



Example 2: point B is shifted slightly to the left and the curve changes entirely since now the order changed from A-B-C to B-A-C



Example 3: although the control points are symmetrical the curve is not



For these reasons I chose to use a second approach, the parametric approach. In this approach each point p is of the form p = (x(t),y(t)).

$$C(t(i,r)) = P_i \cdot H_{00}(r) + P_{i+1} \cdot H_{01}(r) + K_i \cdot H_{10}(r) + K_{i+1} \cdot H_{11}(r)$$

$$\mathbf{t}(i,r) = t_i + r \cdot (t_{i+1} - t_i),\ 0 \le t(i,r) \le 1 \quad ,\ 0 \le r \le 1$$

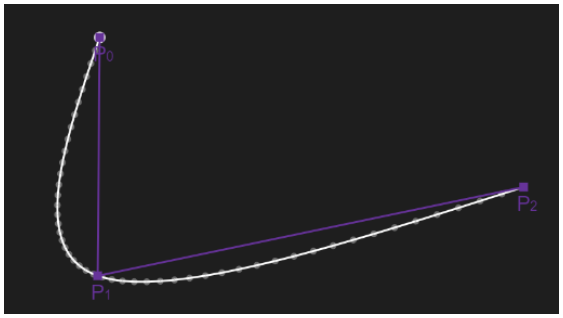
$$K = \begin{pmatrix} K_{0,x} & K_{0,y} \\ \vdots & \vdots \\ K_{i,x} & K_{i,y} \\ \vdots & \vdots \\ K_{n-1,x} & K_{n-1,y} \end{pmatrix} = A^{-1} \cdot \begin{pmatrix} B_{0,x} & B_{0,y} \\ \vdots & \vdots \\ B_{i,x} & B_{i,y} \\ \vdots & \vdots \\ B_{n-1,x} & B_{n-1,y} \end{pmatrix}$$

$$A_{i,j} = \begin{cases} \frac{1}{|t_i - t_j|},\ |i-j| = 1 \\ \frac{1}{t_i - t_{i-1}} + \frac{1}{t_{i+1} - t_i},\ i = j \\ 0,\ otherwise \end{cases},$$

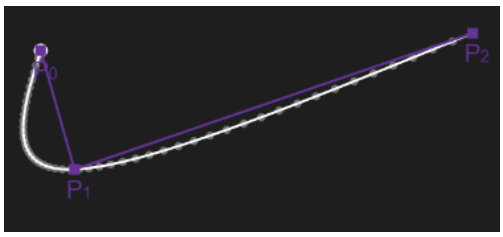
$$D_{i,x} = 3 \cdot \frac{x_{i+1} - x_i}{(t_{i+1} - t_i)^2}, \quad B_{i,x} = \begin{cases} D_{0,x} & ,\ i = 0 \\ D_{n-2,x} & ,\ i = n-1 \\ D_{i-1,x} + D_{i,x} & ,\ otherwise \end{cases}$$

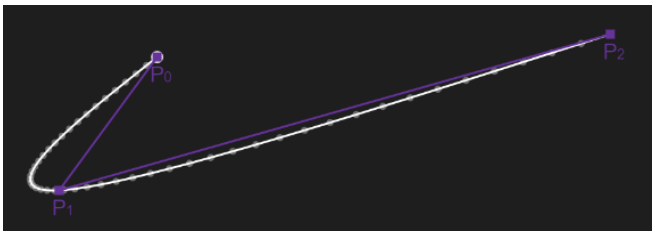
This way the treatment of x and y is symmetrical, the order of the control points remains unchanged, and equal values of x are allowed.

Example: $P_0.x = P_1.x$, we can see the curve remains drawn

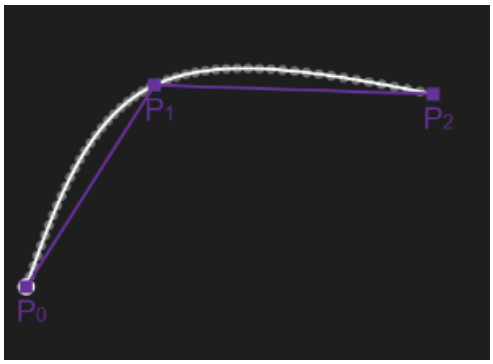


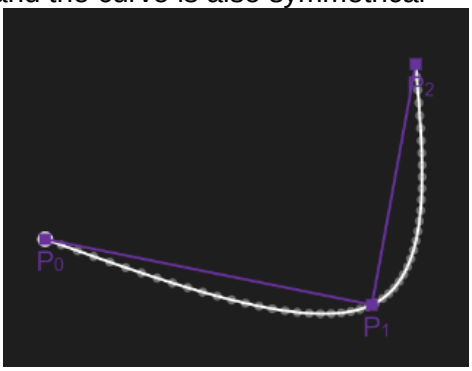
Example: P_1 has shifted to the left but the order of the points connections $P_0 - P_1 - P_2$ remained.





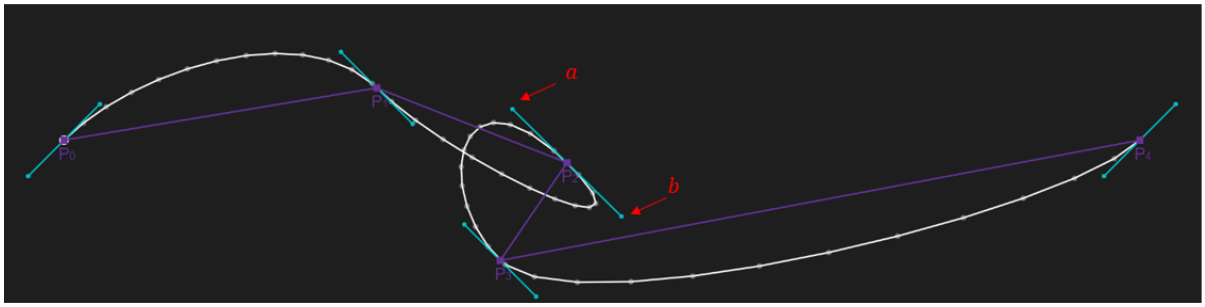
Example: the points are symmetrical and the curve is also symmetrical





Cubic Hermite Spline

After creating the Cubic Spline, I wanted to create the Cubic Hermite Spline. I Could have easily used the same functions I used for Cubic Spline but instead of calculating the Ks I wanted to get them as input. I viewed how it was done in the Hermite spline demo I mentioned before [9]. I didn't really like how it was portrayed so I decided instead to create control lines for each control point. I learned a lot about accessing the mouse movement and controlling the points through the mouse in the process.



In the picture above we can see that the control point P_2 has a light blue control line connected to it that can be controlled either by point a or point b.

$$K_i = (b_i.x - a_i.x, b_i.y - a_i.y)$$

B-Spline

I took the equation for B-Spline the current B-Spline website used in the course [13]:

$$C(t) = \sum_{i=0}^{n} N_{i,k}(t) \cdot P_i, \quad t_{k-1} \le t \le t_n$$

In a B-spline each control point is associated with a basis function $N_{i,k}$ which is given by the recurrence relations:

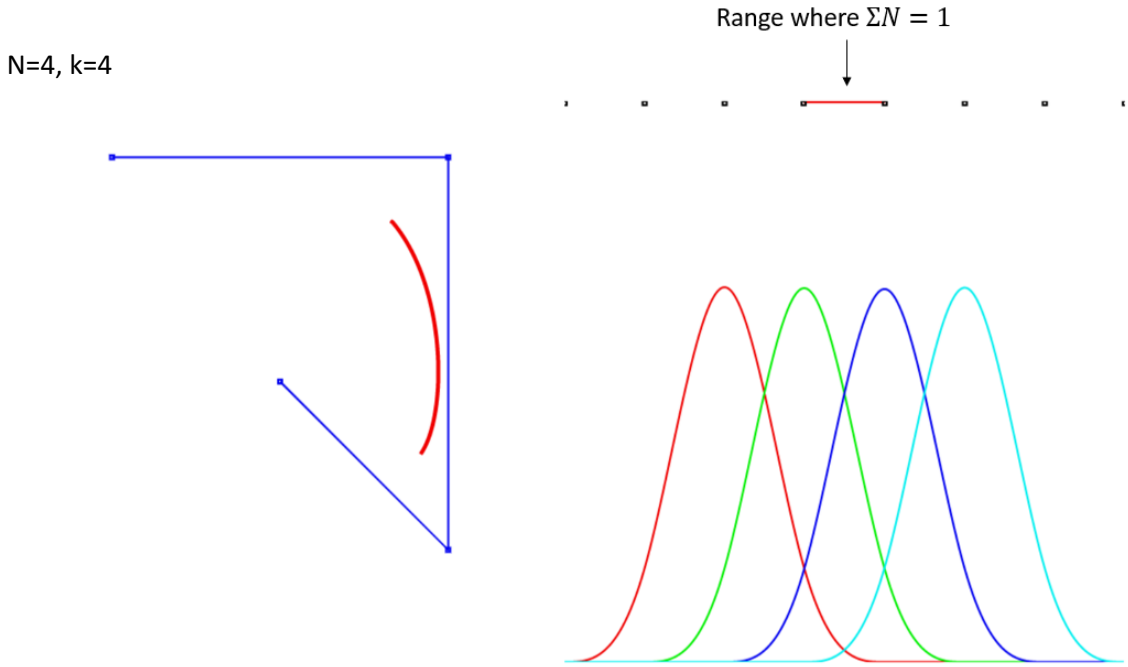
$$N_{i,k}(t) = N_{i,k-1}(t) \cdot \frac{t - t_i}{t_{i+k-1} - t_i} + N_{i+1,k-1}(t) \cdot \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}}$$

$$N_{i,0}(t) = \begin{cases} 1, & t_i \le t \le t_{i+1} \\ 0, & otherwise \end{cases}$$

B-Spline is eventually a weighted average of all the control points, where each basis function is the weight. In order for the point C(t) to be valid the summation of all the basis weights must be 1.

$$\sum_{i=0}^{n} N_{i,k}(t) = 1 \implies t \; is \; valid$$

In the old website[13] where we sum n weights for n control points the range of valid ts may be small in comparison to the control lines. For example:



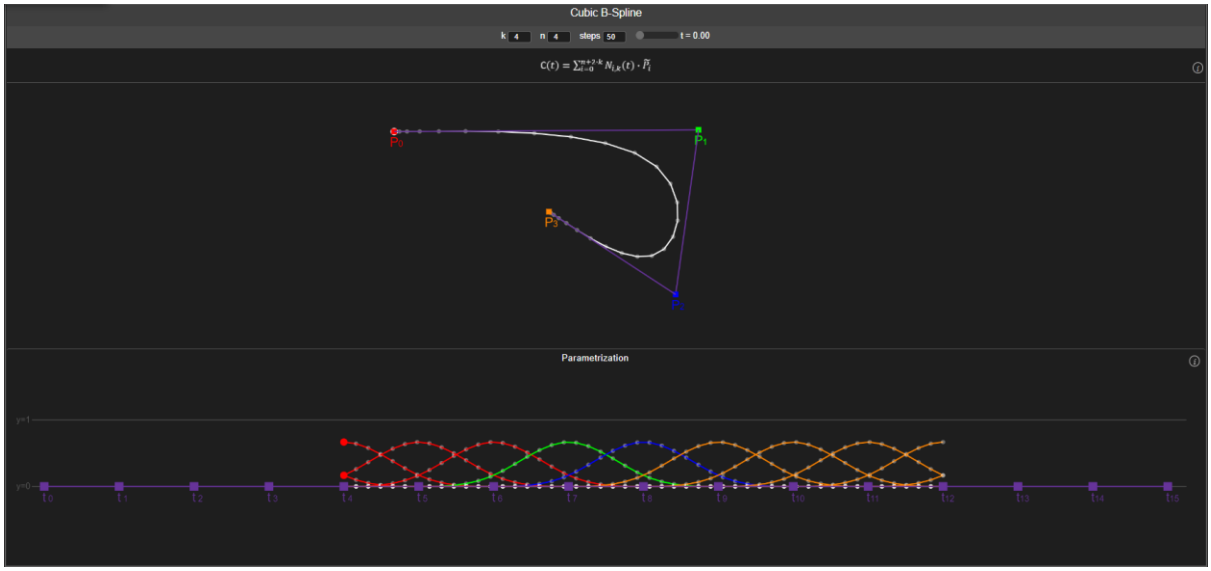
As seen in the picture above the valid range is only between t_3 and t_4 and because of that only a very small portion of the curve is drawn.

To address this issue I added a buffer of k parameter points from each side.

$$C(t) = \sum_{i=0}^{n+2\cdot k} N_{i,k}(t) \cdot \widetilde{P_i}$$

$$\widetilde{P_i} = \begin{cases} P_0, & 0 \le i < k \\ P_{i-k}, & k \le i < n+k \\ P_{n-1}, & n+k \le i \le n+2\cdot k \end{cases}$$

The extended parameterization control points gave a full curve:



The empty buffer ts in the parametrization section are not showing any weight basis curves since the summation is smaller than 1 and those ranges are invalid to draw by them. They are necessary though to set the basis curves.
Similarly to the Lagrange in this curve I also color coded the control points to match the matching base curve (P_i *and* N_i *are in the same color*) only that here there is a buffer so the first k base curves are in the same color as P_0 and the last k sub-curves are in the same color as P_n .

Monomial Basis

A curve with a monomial basis gets an equation that looks like this:

$$C(t) = \sum_{i=0}^{k-1} U_i \cdot t^i$$

In order that C(t) will be a parametric curve we need to have a continuity of C^k.
There are many options to decide how to get the coefficients U_i, I chose to use least squares[14]. I got the coefficients with the following calculations:

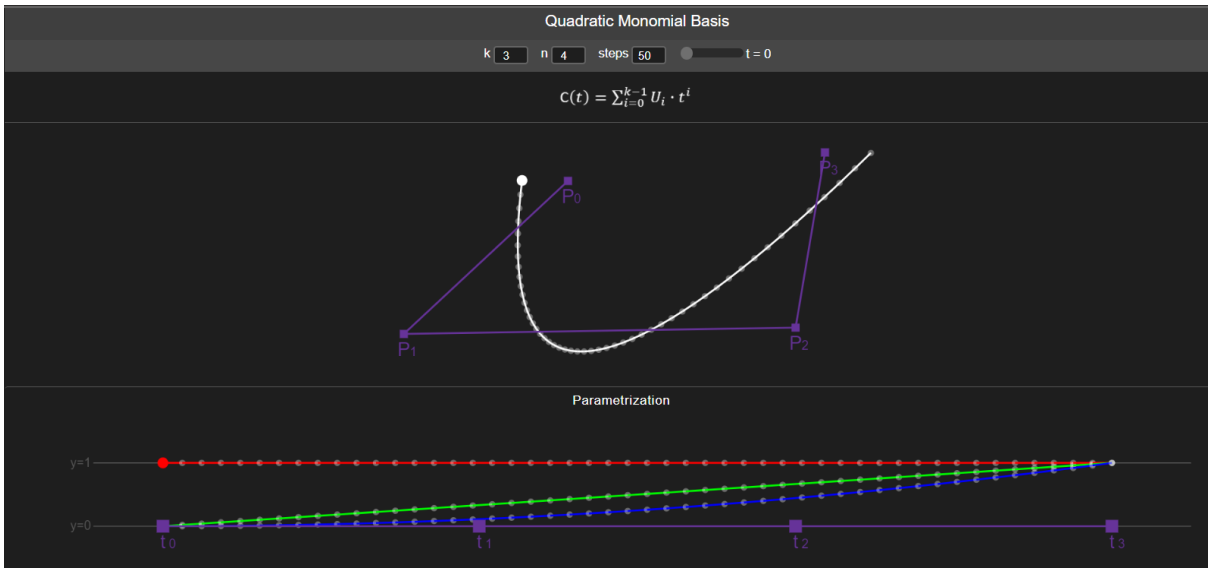
U - coefficients matrix of the polynomial
U_i - vector of row #i of U

$$U = (A^t \cdot A)^{-1} \cdot A^t \cdot P, \qquad A_{p,q} = \begin{cases} 1, & p = q = 0 \\ t_p^q, & otherwise \end{cases}$$

P – control points matrix

row #i of P is the control point $P_i = (x_i, y_i)$

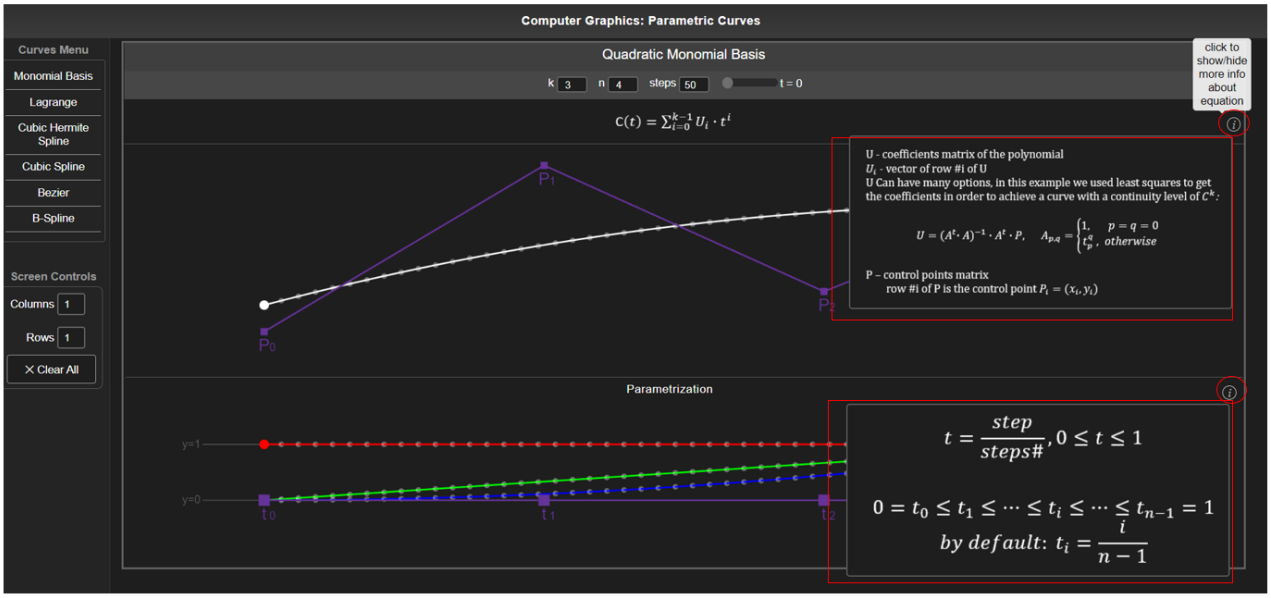
I got a curve that looks like this:



Improving the UI

After finishing to add all the curves I added some finishing improvements to make it easier for the user.

- Information button to all curves equations where I specified my calculations for the specific curve:



- Adding a curve to the table cell not only by dragging the curve name and dropping in the desired cell but also by simply clicking on the curve name in the menu and then adding it to the top left table cell.

- Adding a control point by clicking the desired position for it on the screen. At the beginning I only connected the new control points to the edges of the curve, afterwards I improved it to add the new control point P_i to be between the Ps with the closest x coordinates, or:

$$if\ P_i.x < P_0.x\ push\ P_i\ to\ the\ front\ of\ the\ control\ points\ array$$
$$and\ if\ P_i.x < P_n.x\ push\ it\ to\ the\ end\ of\ the\ array$$

- Deleting a control point by clicking on it
-
- I had many issues with positioning of all the elements on screen that I had to sort out and took a long time to get right

- Alerts when n < 2 for all curves, alert when n < k in monomial basis

- Set 2 as min value for n, k

- A README page to the GitHub project I created

- Make header and sidebar float when scrolling

Future Work

Add More Features

I have worked a lot to try and make the website as learning friendly as possible but during my work there were a few features a I thought be good to add and didn't have time to.

Here are the following features i thought of:

- An information button that will pop a page with extra general information about parametric curves
- Add an option to open all curves at once and save common curves displays for future use, maybe add an option to set a few fixed grid displays that are frequently used
- Add option to match all control points from different screens in order to better compare the curves. For example the user could play with control points of Lagrange and it will change simultaneously also the control points of Bezier in the part of the splitted screen. Like that the student could see more clearly the differences between the different curves.
- Add option to change color schemes. I based my current color scheme on Visual Studio Code dark mode which I feel most comftrable working with but this might not be the color scheme that is comftarble for everyone. In the future it could be nice to add a "Change Color Scheme" control.

Add Tutorial Popups

Many website these days have popup messages when site is first used as tutorials that walk the user through the different buttons and options of the website. It could be nice to add something like this to the website in the future for students who want to use the website for learning and have never used it before or find it not as intuitive.

Make Code More Efficient and Generic

Although I have put a lot of thought to make the code as efficient and generic as possible so other students could improve the website in the future with ease there are still a few ways to make it even better, probably with a bigger use of inheritance. For example currently there is a class called Curve that all the other curves classes inherit from but Cubic Hermite Spline for example is very similar to Cubic Spline. By making it inherit from C-Spline it could save some code duplication.

Add Surfaces

Currently the website presents only parametric curves. In the futuer it might improve the information the websites offers by allowing also to present the variouse surfaces learned in the computer graphics course.

Write Code with React

From reading and learning about web development I leared that most web developers use React or Angular for their web development. Since I wanted to relay on an existing code to build on that did not use those I chose to not do that as well. But a good improvement for the future could be to move the code to be written with React (or Angular). It could make future developement of the website much easier and the code much more elegant and efficient.

Improve Screen Change

One of the biggest issues I faced during my project was how to resize the canvas with the curves in it dynamically. The biggest issue with this was that I need to save all the important data from the previous run delete the running curve object and redraw it on the new canvas with the saved data. I had trouble finding out how to save the data and access all of the different occasions the canvas size should change. In the future it could be good to add a “save curve data” function inorder to make the screen change a smoother experience for the user.

Conclusion

In conclusion, the project granted me an opportunity to learn many new and interesting things in the programming world and develop useful skills.

I got my first experience with web development that included mastering the use of HTML, CSS, JavaScript and web inspection tools. I got familiar with useful elements like Table and Canvas.

I got a chance to have a deeper understanding of the various parametric curves and their algorithms.

I also learned about the challenges integrating pieces of code written by different people together and using existing git projects that were not written by me to serve me as a helpful tool.

I got first hand experience with being a software architect, making technological decisions alone, front-end testing, web design and algorithm implementation.

Overall, this project gave me a wider base of knowledge in diverse fields and a useful experience that could serve me greatly in the future.

References

1. About HTML - https://www.investopedia.com/terms/h/html.asp
2. About CSS - https://developer.mozilla.org/en-US/docs/Web/CSS
3. About JavaScript - https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
4. About Visual Studio Code - https://code.visualstudio.com/docs/supporting/FAQ
5. About GitHub - https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/
6. Chorme Inspecting Tool Tutorial - https://www.browserstack.com/guide/inspect-element-in-chrome
7. Web Development Tutorials - https://www.w3schools.com/
8. Lagrange Website - https://www.ibiblio.org/e-notes/Splines/lagrange.html
9. Cubic Hermite Demo - General Cubic Hermite Spline Demo (codepen.io)
10. Bezier demo (used as base code) - https://github.com/ryanmid/bezier-curves
11. Computer graphics course lecture slides – https://grades.cs.technion.ac.il/grades.cgi?dbeaccjij4f02e421bab04c8399a27+2+234325+Winter2020-2021+ho/WCFiles/08_Modeling-part2.pdf https://grades.cs.technion.ac.il/grades.cgi?dbeaccjij4f02e421bab04c8399a27+2+234325+Winter2020-2021+ho/WCFiles/09_Modeling-part3.pdf https://grades.cs.technion.ac.il/grades.cgi?dbeaccjij4f02e421bab04c8399a27+2+234325+Winter2020-2021+ho/WCFiles/09_Modeling-part4.pdf
12. C-Spline Demo - http://blog.ivank.net/interpolation-with-cubic-splines.html
13. B-Spline Website - B-spline basis functions (ibiblio.org)
14. Least Squares - https://mathworld.wolfram.com/LeastSquaresFitting.html
15. Splines Website - http://scaledinnovation.com/analytics/splines/aboutSplines.html