Крива на Безие и нейната първа поляра

курсов проект по Компютърно геометрично моделиране (CAGD)

Лектор: доц. Красимира Влъчкова

Цветомир Стайков - 1MI0800469

2 курс, Компютърни науки

12 февруари 2025 година

Съдържание

1. Въведение

Настоящият проект представя двуизмерна крива на Безие и нейната първа поляра. За генерирането на кривите се използва алгоритъмът на de Casteljau. Реализирани са стандартни функции за създаване на криви. Програмата работи в уеб браузър и е тествана на настолни компютри.

2. Структура и технологии

Проектът представлява уеб приложение, разработено с HTML, CSS и JavaScript, като използва панел за рисуване, още познат като Canvas, за визуализация на кривите. Няма външно извикани библиотеки, което позволява проекта да работи без нуждата от интернет. Изборът на тези технологии позволява лесна адаптация и използване на програмата на различни устройства.

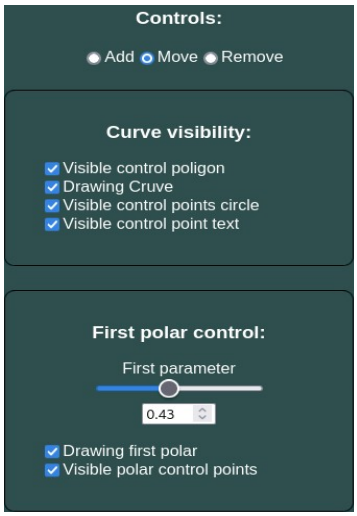
Компресирана структура на проекта

- *CAGD-1MI0800469.html* – Целия проект компресиран в един файл.

Оригинална структура на проекта

- *index.html* – Начална страница на сайта.
- *styles.css* – Файл с дизайн на интерфейса.
- *Interface.js* – Основни функции за създаване, управление и изтриване на контролните точки, както и за визуализация на крайната крива.
- *Algorithms.js* – Алгоритми за изчисление на кривата и нейната първа поляра.
- *Debug.js* – Помощни функции за извеждане на информация и грешки.

3. Интерфейс



Потребителският интерфейс се състои от две основни части:

- Ляво, контролен панел - Панела има настройки за създаване на крива. Позволява добавяне, местяне и премахване на контролни точки. Има кутийки с тикчета, с които може да се променя видимоста на контролния полигон, точките, кривата, надписите, както на полярата и неините точки. Даден е слайдер, който контролира първия параметър на поларята. До

него има текстова кутия, която приема числа, за по-прецизен контрол на параметъра до втори знак след запетаята. Имплементирано е ограничения в интервала [0,1]. За оправление на контролните точки, най-горе в контролния панел има 3 настройки – Add, Move и Remove. Когато програмата се влючи за първи път е избрана настройката Add.

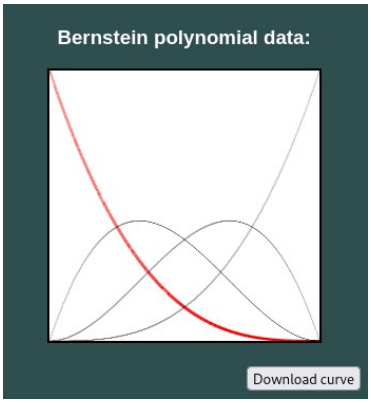
- Дясно, панела за рисуване (Canvas) – Следкато е избрана една от настройките за контролните точки чрез натискане на дясно копче на мишката може да се манипулират точките върху този панел. Когато мишката е върху някоя контролна точка, при настройка Move и Remove, то тази точка ще се огради в червено, което синвулизира, че може да бъде избрана.

Основни функции върху контролните точки

- Add – Може да се добавят нови точки, като най-ново добавената ще бъде следващата по-ред. Чрез ляв бутон върху панела за рисуване може да се добавя точката.
- Move – Като мишката се сложи върху дадена контролна точка, тя ще се огради с червен кръг. Чрез натискане и задържане на ляв бутон, този кръг изчезва и контролната точка ще следва мишката.
- Remove – Използва същата функционалност като на настройката „Move“, но вместо да мести контролната точка, я изтрива при натискане на ляв бутон.

Допълнително

- Програмата позволява да се изтегли нарисуваната крива без фон, чрез бутона „Download curve“ най-долу в контролния панел.
- Представени са и полиномите на полином на Бернщайн в малко панелче най-долу в контролния панел. Ако сложим мишка на върху дадена контролна точка при избрана опция Move или Remove, полинома на тази точка ще се маркира в червено.



4. Алгоритъм

За пресмятане на кривата е използван алгоритъм на de Casteljau. След като потребителят добави контролни точки, автоматично програмата генерира кривата. Използва се цикъл и малка стъпка на t, за да изглежда кривата гладка.

Програмта използва две функции за генериране на кривата и нейната поляра и три помощни подфункции.

Главната функцията има за цел да нарисуват кривата със стъпка 0.01. Проверява се дали е възможно да се генерира крива според това дали има достатъчно контролни точки. Тези функции се възползват от подфункции, които имат за цел да изчислят междинните точки. Резултата от тази помощна функция е нужната точка за чертаене на кривата. Приема като аргумент t и контролни точки, след което се използва алгоритъма на de Casteljau.

```
function calculateAtPos(t,LocalControlPoints){
    let count = LocalControlPoints.length;

    let intermediatePoints = [LocalControlPoints]

    for(let i = 0;i<count-1;++i){
        let prev = intermediatePoints[i];
        let newPoints = new Array(prev.length-1);

        for(let j =0;j<prev.length-1;++j){
            let first = prev[j];
            let second = prev[j+1];
            newPoints[j] = {
                x: (1-t)* first.x + t * second.x,
                y: (1-t)* first.y + t * second.y
            }
        }
        intermediatePoints.push(newPoints);

    }

    return intermediatePoints[intermediatePoints.length-1][0];
}
```

```
function drawCurve(){
    if(controlPoints.length <=1){
        return;
    }
    ctx.lineWidth = 2;
    ctx.strokeStyle = "grey";

    for(let i =0;i<=1;i+=0.001){
        pos =calculateAtPos(i,controlPoints);

        ctx.beginPath();
        ctx.moveTo(pos.x, pos.y);

        pos =calculateAtPos(i+0.001,controlPoints);

        ctx.lineTo(pos.x, pos.y);
        ctx.stroke();
    }

    ctx.strokeStyle = "black";
}
```

Функцията за поляра използва същия метод, но тя допълнително приема променливата t_1, подадена от интерфейса. Тази променлива се подава на подфункцията, която изчислява полярата.

```
function calculateAtPosFirstPolar(t0,t,LocalControlPoints){
    let count = LocalControlPoints.length;

    let intermediatePoints = [LocalControlPoints]

    for(let i = 0;i<count-1;++i){
        let prev = intermediatePoints[i];
        let newPoints = new Array(prev.length-1);
        for(let j =0;j<prev.length-1;++j){

            let first = prev[j];
            let second = prev[j+1];

            if(i == 0){
                newPoints[j] = {
                    x: (1-t0)* first.x + t0 * second.x,
                    y: (1-t0)* first.y + t0 * second.y
                }
            }else{
                newPoints[j] = {
                    x: (1-t)* first.x + t * second.x,
                    y: (1-t)* first.y + t * second.y
                }
            }
        }
        intermediatePoints.push(newPoints);
    }
    return intermediatePoints[intermediatePoints.length-1][0];
}
```

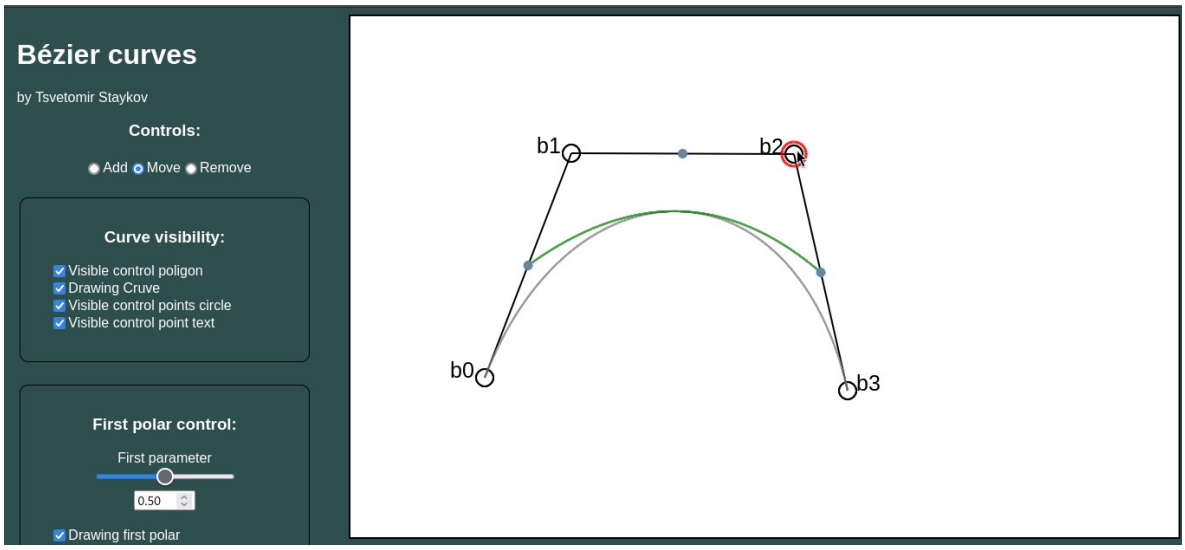
```
function drawFirstPolar(t0, drawControlPointOfPolar){

    if(t0 < 0 || t0 > 1){
        DebugError("Polar t0 is out of [0,1] bound");
        return;
    }

    if(controlPoints.length <=2){
        return;
    }

    for(let i =0;i<=1;i+=0.001){
        let pos =calculateAtPosFirstPolar(t0,i,controlPoints, drawControlPointOfPolar,5,i)

        ctx.beginPath();
        ctx.moveTo(pos.x, pos.y);

        pos =calculateAtPosFirstPolar(t0,i+0.001,controlPoints,false);

        ctx.lineTo(pos.x, pos.y);
        ctx.lineWidth = 2;
        ctx.strokeStyle = "green";
        ctx.stroke();
    }
```

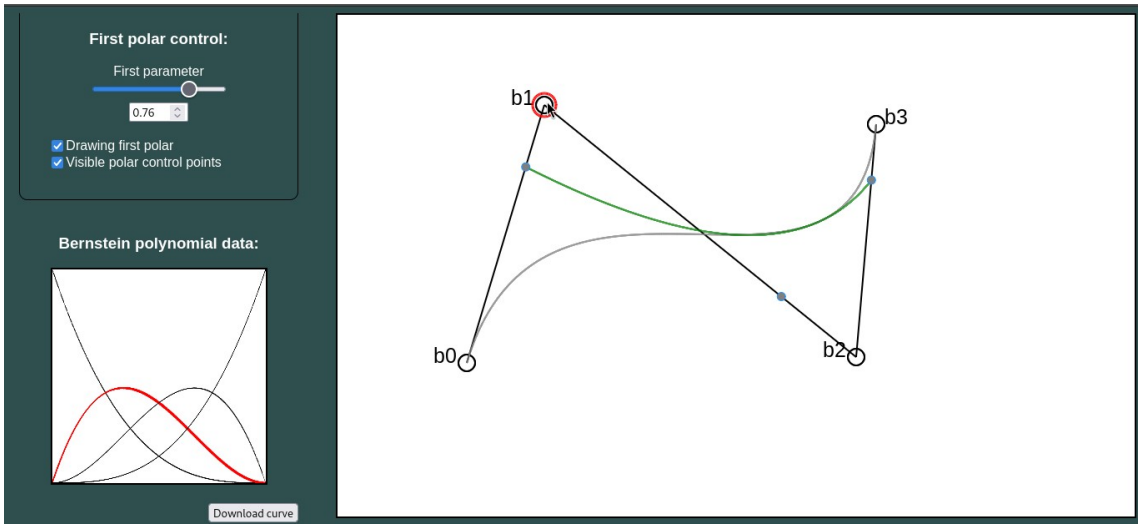
Има отделна функция, която изчисля един път алгоритъма на de Casteljau и връща { $b_0^1(t_1)$, ... $b_{n-1}^1(t_1)$ }, които се изобразяват на контролния полигон. Те могат да се разгледат като контролни точки на полярата.

```
if(document.getElementById("DrawPolarControlPointsCheckbox").checked){
  let polarControlPoints = PolarControlPoints(t0,controlPoints);
  for(let i = 0; i<polarControlPoints.length; ++i){

      ctx.beginPath();
      ctx.arc(polarControlPoints[i].x,
              polarControlPoints[i].y,
               5, 0, 2 * Math.PI);
      ctx.fillStyle = "gray";
      ctx.fill();
      ctx.lineWidth = 1;
      ctx.strokeStyle = "DodgerBlue";
      ctx.stroke();
  }
```

```
function PolarControlPoints(t0,LocalControlPoints){

    let count = LocalControlPoints.length;
    let PolarControlPoints = new Array(count-1);

    for(let j =0;j<count-1;++j){

        let first = LocalControlPoints[j];
        let second = LocalControlPoints[j+1];

        PolarControlPoints[j] = {
            x: (1-t0)* first.x + t0 * second.x,
            y: (1-t0)* first.y + t0 * second.y
        }

    }

    return PolarControlPoints;
}
```

5. Снимки на проекта





6. Краен продукт

Крайния продукт е един Html файл, който съдържа всичко нужно да работи и може да се пусне през стандарните браузъри.

Работеща версия на проекта е качена на сайта: eguardian.dev/CAGD

Оригиналния код е качен в Github, където може да се видят и различни итерации и прогреса на кода чрез контрол на версията.

7. Използвани Ресурси

- Записки по курса Компютърно геометрично моделиране (CAGD) на лектор доц. Красимира Влъчкова, София, 2024 г.
- Mozilla’s Javascript canvas tutorial/documentation - https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial

Допълнителни връзки

- https://www.eguardian.dev/CAGD/
- GitHub https://github.com/eGuardianDev/University-Work/tree/main/(%202%20)%20second%20Year/(%203%20)%20%20Third%20semester/CAGD/