Subdivision алгоритъм за криви на Bézier

Описание на проекта

Проектът представлява интерактивен уебсайт, който демонстрира subdivision алгоритъм на крива Bézier. Потребителите могат да добавят контролни точки чрез кликване върху платното (canvas) и могат да регулират разделянето на кривата чрез плъзгач (slider). В процеса на интеракция, потребителите могат да визуализират различни стъпки от разделянето на кривата и да изчистят платното (canvas).

Интерфейс

1. Платно (canvas)

   Елементът <canvas> с id "bezierCanvas" представлява областта, в която се рисува кривата Bézier и контролните точки (набор от 2„контролни точки“ дефинира гладка, непрекъсната крива с помощта на формула).

   Интерактивността с канваса включва добавяне на контролни точки и регулиране на тяхната позиция.

2. Плъзгач (slider) и стойност на плъзгача (Value)

   Елементът <input> с id "slider" представлява плъзгач, чрез който потребителите могат да регулират степента на разделяне на кривата. Над него е видима и стойността, която заема той.

3. Бутон за изчистване

   Бутона "Clear Canvas" се използва за изчистване на канваса.

4. Бутон за премахване на последната добавена точка

   Бутона “Delete last point” се използва за премахване на последната контролна точка, добавена от потребителя.

5. “User guide”
   Инструкции за използване на програмата.

6. Checkbox
   При кликване върху checkbox-а се визуализира кривата на Bézier (при стартиране на програмата checkbox-а е неотбелязан).

JavaScript функционалности

1. Инициализация

   Функцията initializeControlPoints се използва за инициализиране на контролните точки, задавайки празен масив.

2. Рисуване на крива Bézier

   Функцията drawBezier се използва за рисуване на кривата Bézier върху канваса.

   Използва се алгоритъмът на deCasteljau за изчисляване на точките от кривата.

3. Рисуване на контролни точки и линии

   Функциите drawControlPoints и drawLines се използват за рисуване на контролните точки и линиите между тях върху канваса.

4. Обработка на интеракции

   Функциите handleMouseDown, handleMouseMove и handleMouseUp обработват интеракциите с мишката, като позволяват теглене и преместване на последната добавена контролна точка.

   Функцията handleCanvasClick добавя нова контролна точка при кликване върху канваса.

5. Анимация/ визуализация

   Функцията animate се използва за анимиране (визуализация) на промените в контролните точки и разделянето на кривата.

6. Изчистване на платното

   Функцията clearCanvas изчиства канваса и нулира контролните точки.

7. Изчистване на последната добавена контролна точка

   Функцията deleteLastPoint премахва последната добавена точка в масива от контролни точки.

8. Смяна на видимостта на кривата

   Функцията toogleBezierVisibility се използва за промяна на флага на променливата, която се използва за да следи каква е видимостта на кривата (заема стойности true или false).

9. Линейна интерполация

(Интерполация в числовия анализ е метод на конструиране на нови числови стойности в област от множество на изолирани точки от известни числови стойности.)

10. Разделяне на кривата на Bézier

    Въз основа на контролните точки, функцията subdivideBezier, интерполира нови точки между всяка двойка такива, като използва параметъра t.

11. Промяна на стойността на плъзгача

    Функцията updateSlider се използва за актуализиране на стойността на slider-а (плъзгача) и след това стартира функцията animate, която отговаря за анимирането на Bézier кривата в зависимост от новата стойност на slider-а.

Използвани математически алгоритми

- deCasteljau алгоритъм за построяване на кривата на Bézier.

Нека n ∈ N, $b_0, b_1, \ldots, b_n$ са n + 1 различни точки в $\mathbb{R}^3$ и t ∈ [0, 1].
Алгоритъмът на deCasteljau използва последователни линейни интерполации и след n стъпки построява точка $b^n{}_0(t)$ върху полиномиална крива B от степен n.
Кривата B се нарича крива на Bézier.
Точките b_i, i = 0, . . ., n, се наричат контролни точки или точки на Bézier.
Полигонът с върхове $b_0, b_1, \ldots, b_n$ се нарича контролен полигон или полигон на Безие на кривата.

Вход:

$b_0, b_1, \ldots, b_n \in \mathbb{R}^3$,
t ∈ $\mathbb{R}$
$b^r{}_i(t) = (1 - t)b^{r-1}{}_i(t) + tb^{r-1}{}_{i+1}(t)$, за $r = 1, \ldots, n$; $i = 0, \ldots, n - r$
$b^0 i = b_i$

Изход:

$b^n{}_0(t)$ е точка от кривата B, съответстваща на параметъра t.

Схема на deCasteljau:

b_0

b_1 $b^1{}_0$

b_2 $b^1{}_1$ $b^2{}_0$

b_3 $b^1{}_2$ $b^2{}_1$ $b^3{}_0$

Алгоритъмът на deCasteljau може да се обобщи по следния начин:

b_0

b_1 $b^1{}_0[t1]$

b_2 $b^1{}_1[t1]$ $b^2{}_0[t1, t2]$

b_3 $b^1{}_2[t1]$ $b^2{}_1[t1, t2]$ $b^3{}_0[t1, t2, t3]$

Точката $b^3{}_0[t1, t2, t3]$ е функция на три независими променливи, следователно вече не описва крива, а област в $\mathbb{R}^3$.

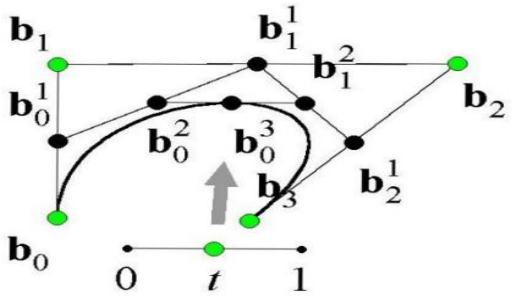
Пример:

$(0, 0)$

$(0, 2)$ $(0, 1)$

$(8, 2)$ $(4, 2)$ $(2, 3/2)$

$(4, 0)$ $(6, 1)$ $(5, 3/2)$ $(7/2, 3/2)$

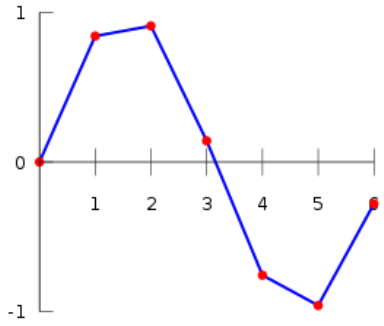


Алгоритъм на deCasteljau за n = 3.

- Линейна интерполация за изчисляване на точки между контролните точки.

Линейната интерполация е форма на интерполация, която включва генериране на нови стойности въз основа на съществуващ набор от стойности. Линейната

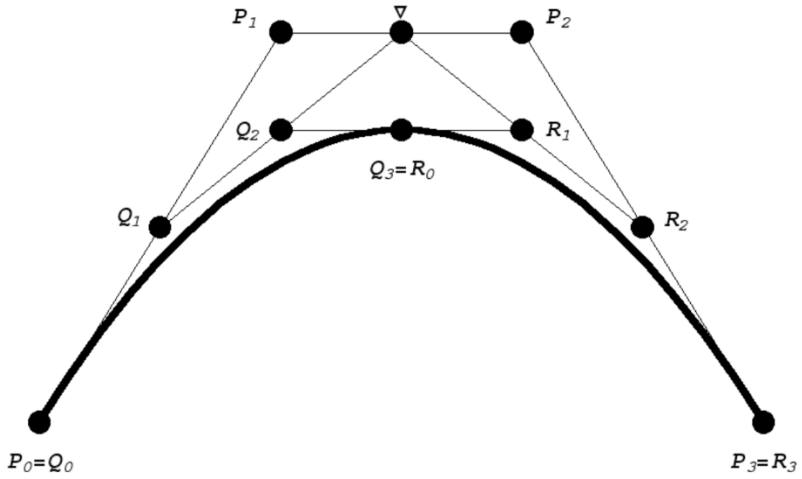
интерполация се постига чрез геометрично изобразяване на права линия между две съседни точки на графика или равнина. Всички точки на линията, различни от оригиналните две, могат да се считат за интерполирани стойности.



Линейна интерполация

- Subdivision алгоритъм за криви на Bézier.

Нека кривата на Bézier b^n има контролни точки b_i , $i = 0, \ldots, n$ и е дефинирана в интервала [0, 1].
Нека 0 < c < 1.
Частта от кривата, съответстваща на [0, c] може също да се дефинира с полигон на Bézier.
Намирането на този полигон се нарича subdivision (подразделяне) на кривата на Bézier.



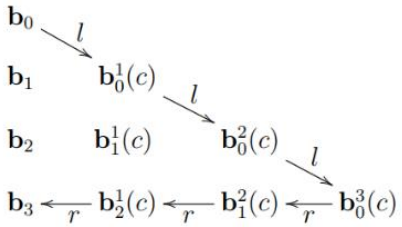
Subdivision на крива на Bézier.
(за c = $\frac{1}{2}$ и $n = 3$)

Неизвестните контролни точки c_i се намират лесно като използваме функцията blossom.

Знаем, че $b_i = b(0^{<n-i>}, 1^{<i>})$, а междинните точки са: $b^r{}_i = b(0^{<n-r-i>}, t^{<r>}, 1^{<i>})$, където $t^{<r>}$ означава, че t участва r пъти като аргумент.

Ако кривата на Bézier е дефинирана в интервала *[a, b]*, тогава $b_i = b(a^{<n-i>}, b^{<i>})$. Сега се интересуваме от интервала *[0, c]* (т. е. $a = 0$, $b = c$). В този случай имаме, че контролните точки са $b(0^{<n-i>}, c^{<i>})$.
От алгоритъма на deCasteljau, приложен за t = c следва
$b^i{}_0 = b(0^{<n-i>}, c^{<i>}, 1^{<0>}), = b(0^{<n-i>}, c^{<i>})$.
Следователно $c_i = b^i{}_0$ (c). Поради свойството „симетрия“ на крива на Bézier, контролните точки за частта от кривата, съответстваща на интервала *[c, 1]* са $b^{n-1}{}_i$ (c).

Може да извършим многократно subdivision. Първо за $t = \frac{1}{2}$, след това всяка от получените две криви подразделяме пак на две и т. н. След k нива на subdivision получаваме 2^k полигони на Bézier, като всеки от тях описва малка дъга от кривата. Ако $k \to \infty$, тези полигони клонят към кривата.



Контролните точки на лявата (l) и дясната (r) кубични криви на Bézier, получени след subdivision. Стрелките показват реда на съответните контролни точки.

Обяснено по друг начин, алгоритъмът за подразделяне на криви на Bézier, играе важна роля в апроксимирането на криви. Този алгоритъм позволява разделянето на кривите на Bézier на по-малки сегменти, които са все още криви на Bézier, но с по-малка степен.

Ето обяснение на стъпките на алгоритъма за подразделяне:

1. Инициализация:

Първоначално имаме точки контрол, които представляват кривата на Bézier.
Ако имаме n+1 контролни точки, тогава имаме крива на Bézier от степен n.

2. Подразделяне на сегменти:

За да подразделим кривата на Bézier на два сегмента, използваме т. нар. "точка на разцепване" (splitting point).
Този точка се намира чрез линейна интерполация между контролните точки, които представляват началото и края на кривата.

3. Създаване на нови контролни точки:

Новите контролни точки се изчисляват, като комбинираме оригиналните контролни точки по следния начин:
Новите точки от дясната страна на подразделящата точка се изчисляват чрез линейна интерполация между съседните точки от дясната страна на оригиналната точка на разцепване.
Същото се прави и за лявата страна.

4. Повторение:

След като сме разделили кривата на Bézier на два нови сегмента, процесът се рекурсивно повтаря за всяка половина.
Рекурсията продължава, докато стигнем до криви на Bézier от ниска степен (например, линии или криви от степен 1).

Инструкции за използване

- Добавянето на контролни точки става чрез кликване върху канваса.
- Регулирането на степента на разделяне е с плъзгача, чрез плъзгане наляво и надясно.

  Анимацията автоматично реагира на промените в контролните точки и степента на разделяне.

- Може да се мести последната добавена от потребителя, контролна точка, чрез double-tap върху нея.
- Изтриването на досегашното платно става при натискане на бутона Clear Canvas, а изтриването само на последната добавена контролна точка е при натискането на бутна Delete Last Point.
- Показването на кривата на Bézier става при кликване върху празния checkbox под платното.

Използвани технологии

- HTML5 за структурата на уебсайта.
- CSS3 за стилизацията на елементите.
- JavaScript за логиката на интеракцията и рисуването.

Заключение

Уебсайтът предоставя интерактивен начин за изучаване на разделянето на крива Bézier. Потребителите могат да манипулират контролните точки и да визуализират как кривата се дели в реално време.

Източници

https://learn.fmi.uni-sofia.bg/pluginfile.php/458081/mod_resource/content/5/lecture_notes_CAGD.pdf

https://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D0%B0%D1%86%D0%B8%D1%8F

https://bg.theastrologypage.com/linear-interpolation