



# Курсова работа

на тема

Реализиране на алгоритъма за намиране на cross plot на крива на Bezier

за курса

Компютърно геометрично моделиране (CAGD)

Лектор: доц. Красимира Влъчкова

Изготвена от

Байрям Байрямов Дерменджиев

II курс, Софтуерно инженерство

# Съдържание

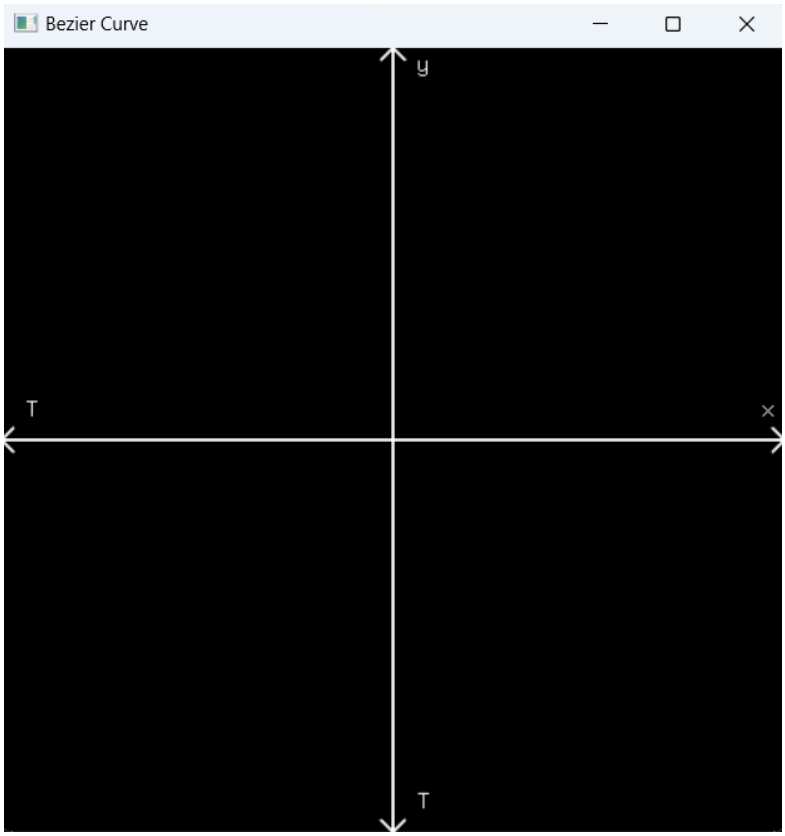
# Условие

Да се създаде софтуер, чиито основни функционалности са:

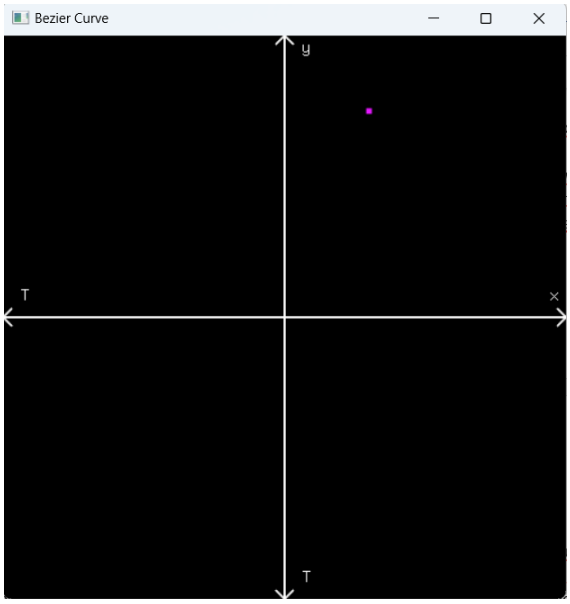
- при зададени от потребителя точки, те да се възприемат като контролни такива и спрямо тях да се начертае кривата на Bézier
- спрямо вече начертаната крива на Bézier, софтуерът трябва да има възможността при желание на потребителя да начертае cross plot-а на тази крива.

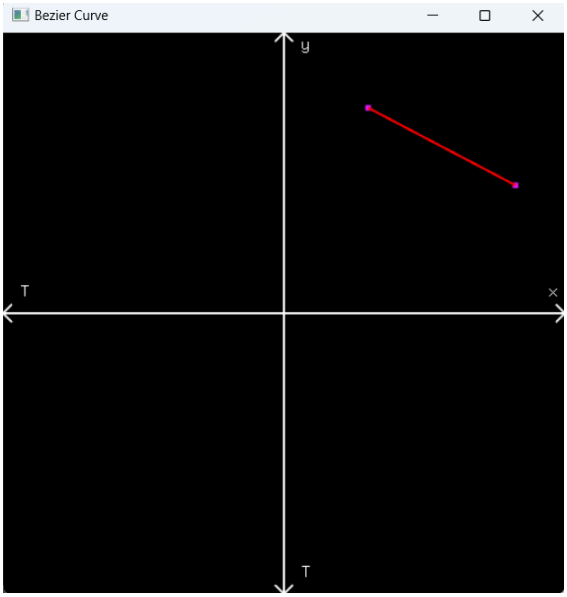
# Функционалноссти на програмата

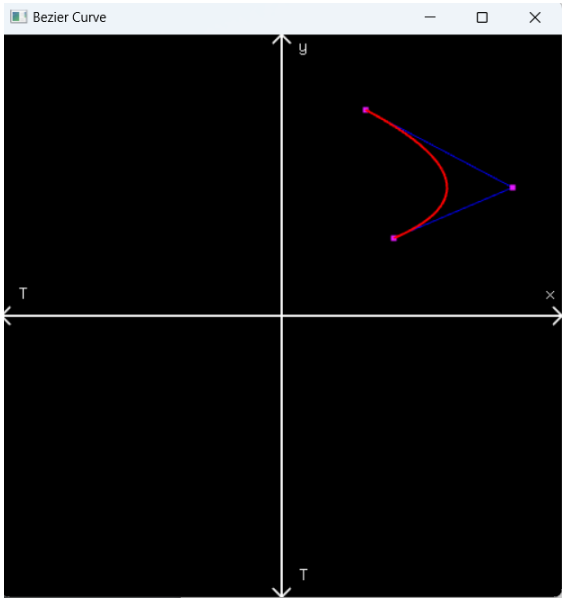
Първото нещо, с което трябва да се запознае потребителят, който използва софтуера е Декартовата система, която е първото нещо, което излиза на прозореца при пускане на програмата. Потребителят може да си взаимодейства с програмата само и единствено чрез първи квадрант.

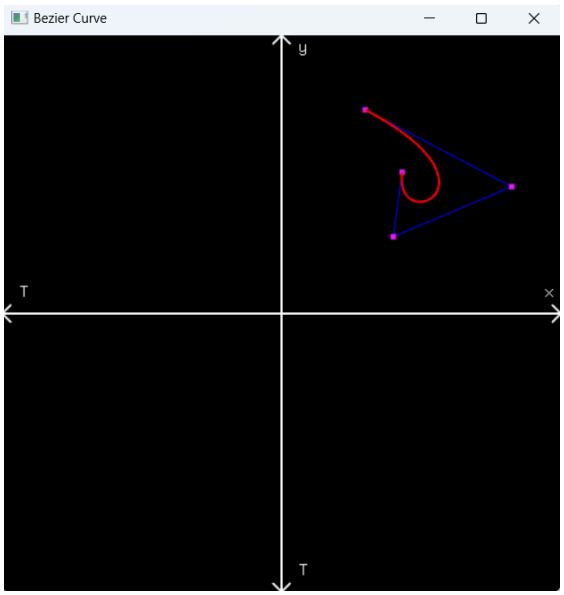



1. Когато потребителят кликне левия бутон на мишката върху първи квадрант, се създава нова контролна точка точно на текущата позиция на курсора. След всяко такова добавяне на нова точка автоматично се изчертава кривата на Bézier, използвайки актуализираните контролни точки. Този процес включва добавянето на новата точка към предходните, което води до промяна в броя на контролните точки и, съответно, до създаването на нова крива на Bézier с променени характеристики.

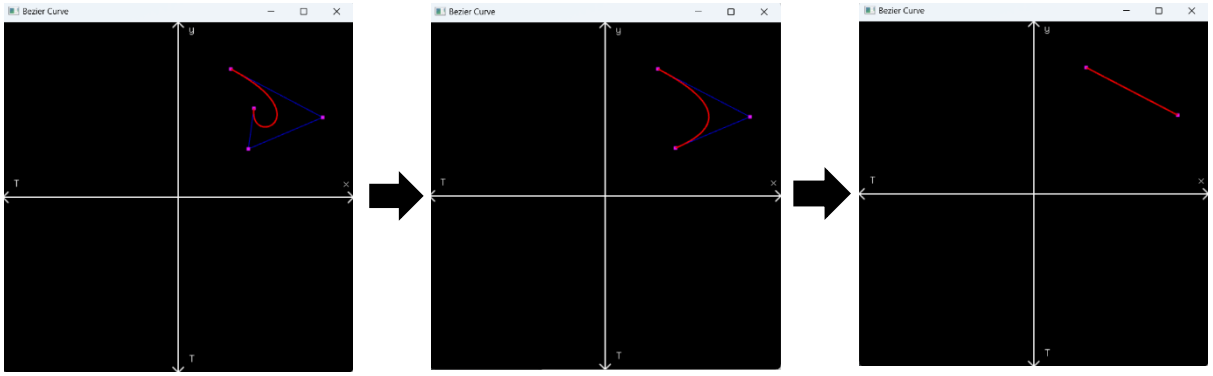


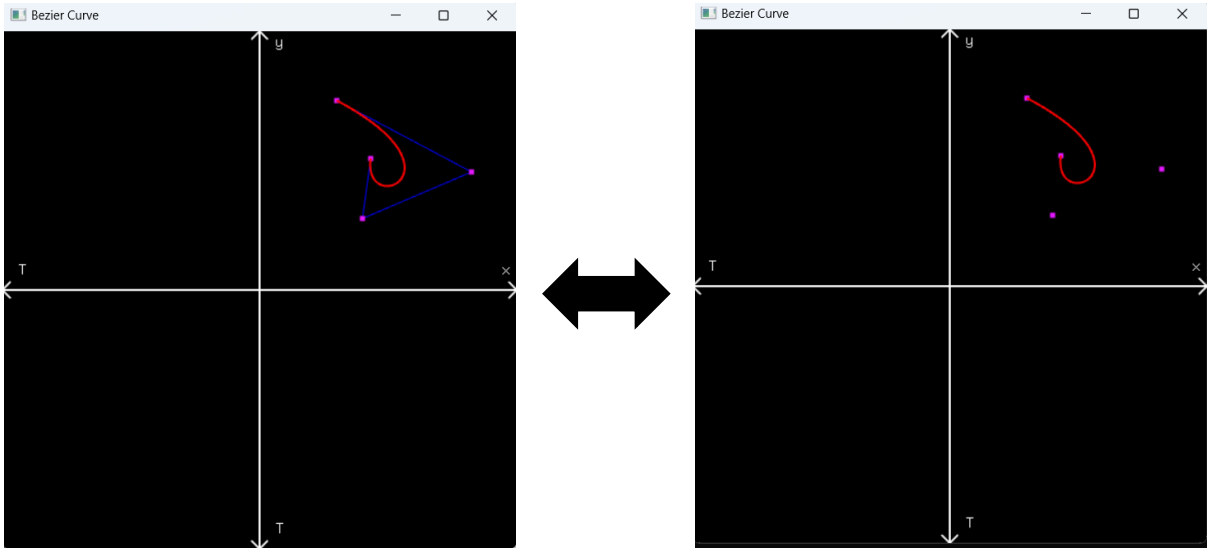



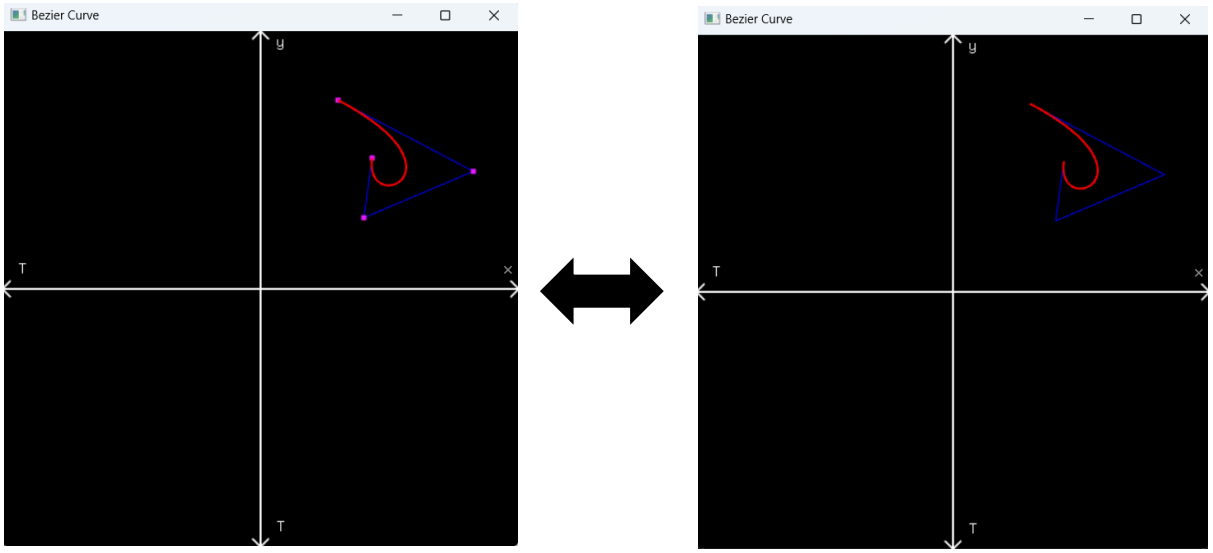

2. При щракване на десния бутон на мишката в първия квадрант, потребителят изтрива последно добавената контролна точка (ако такава съществува). След всяко такова изтриване автоматично се чертае нова крива на Bézier, като се използват обновените контролни точки. Точката се премахва от контейнера с контролни точки, което довежда до създаването на нова крива на Bézier с актуализирани характеристики.



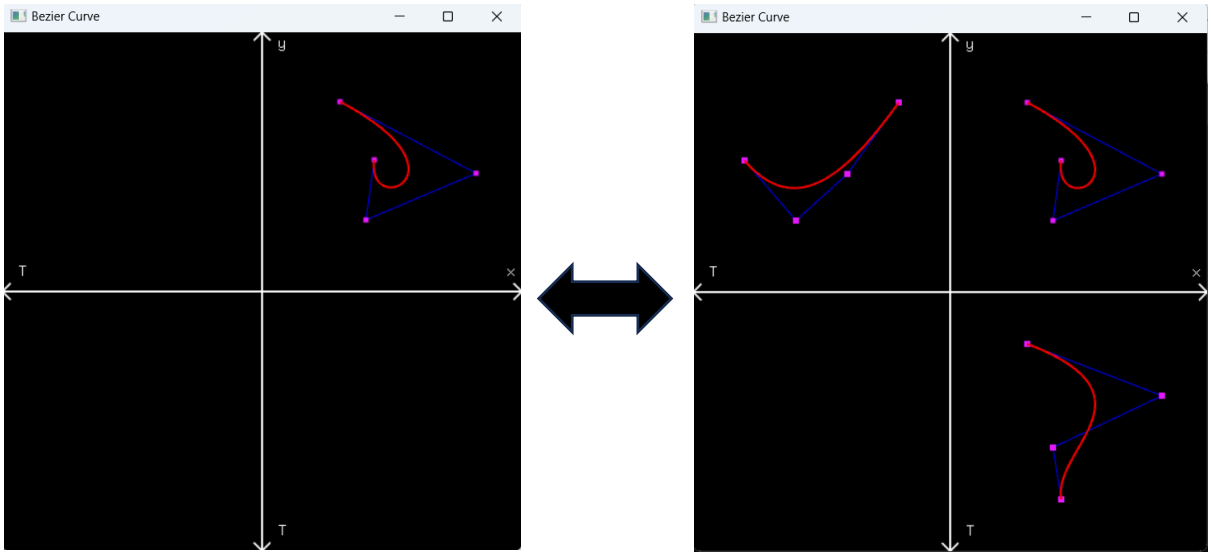
3. При натискане на клавиша ’-’ се начертава/скрива контролния полигон.




4. При натискане на клавиша ‘.’ се начертават/скриват контролните точки.




5. При натискане на клавиша ‘f’ се начертава/скрива cross plot-а на кривата на Bézier. Във втори квадрант ще бъде нарисувана/скрита y(t) функцията, а в четвърти квадрант ще бъде нарисувана/скрита x(t) функцията.




6. При натискане на клавиша ‘r’ се намалява стойността на червения цвят с 0.1 в цвета на кривата на Бézier.

7. При натискане на клавиша ‘b’ се намалява стойността на синия цвят с 0.1 в цвета на кривата на Бézier.

8. При натискане на клавиша ‘g’ се намалява стойността на зеления цвят с 0.1 в цвета на кривата на Бézier.

9. При натискане на клавиша ‘1’ се намалява стойността на червения цвят с 0.1 в цвета на контролния полигон.

10. При натискане на клавиша ‘2’ се намалява стойността на синия цвят с 0.1 в цвета на контролния полигон.

11. При натискане на клавиша ‘3’ се намалява стойността на зеления цвят с 0.1 в цвета на контролния полигон.

# По – комплекси функции

1. computeBezierPoint - чрез нея изчисляваме точка от кривата на Бézier при подадени контролни точки и дадена стойност t. Използваният алгоритъм е рекурсивният алгоритъм на de Casteljau:
    1. На първоначалната стъпка се използват контролните точки на Бézier кривата.
    2. За всяка двойка последователни контролни точки се изчислява нова точка на отсечка между тях, спрямо параметъра t (0 <= t <= 1).
    3. Новите точки се използват като контролни точки за следващата стъпка.
    4. Процесът се повтаря, като се изчисляват нови точки до стъпка n, където n е степента на Бézier кривата.
    5. Накрая, остава една финална точка, която е крайната точка на Бézier кривата.

$$b_i^r(t) = (1 - t)\, b_i^{r-1}(t) + t b_{i+1}^{r-1}(t),$$
$$r = 1,\ldots, n;\ i = 0,\ldots, n - r$$
$$b_i^0 = b_i$$

Функцията работи рекурсивно като изчисленията вървят надолу по йерархията, докато не се изчислят всички междинни точки нужни за изчислението на финалната точка.

**2.** computeBezier - чрез тази функция се чертае кривата на Bézier. Използва се цикъл, в който параметърът t изменя стойността си в интервала [0,1], като на всяка итерация след първата t = t + 0.01 и този параметър се подава във функцията computeBezPoint. По този начин на всяка итерация получаваме точка от кривата, която се намира съвсем близо до предишната. След което функцията начертава права между двете точки. Така след 100 на брой итерации, ще сме получили 100 на брой прави, които представляват кривата на Bézier. Големият брой прави, чрез които се представя кривата са причината тя да изглежда гладка. Затова параметърът t се изменя с толкова малка стойност, а именно 0.01.

**3.** computeXFuncPoints – чрез тази функция се изчисляват координатите на контролните точки на функционалната крива x(t).

**Алгоритъм:**

За всяка контролна точка на функционалната крива:

- всяка една от точките има същата координата x като i-та по ред на поява контролна точка на кривата на Bézier.
- y координатата на всяка точка се намира като интервалът на четвърти квадрант за y, като при нас това е [windowHeight / 2, 0], се разбие на броя контролни точки. От това разбиване се получават n + 1 равни подинтервала, като n е броят на контролните точки. Всеки край на интервал, без последния, е y координатата на i-тата контролна точка на функционалната крива. По опростено чрез формула за $yi$ :

$yi$ = (windowHeight / 2) – i * ((windowHeight / 2) / (n + 1)), за i = 1, . . ., n

**4.** computeYFuncPoints - чрез тази функция се изчисляват координатите на контролните точки на функционалната крива y(t). Като това става много подобно на computeXFuncPoints.

**Алгоритъм:**

За всяка контролна точка на функционалната крива:

- всяка една от точките има същата координата y като i-та по ред на поява контролна точка на кривата на Bézier.
- x координатата на всяка точка се намира като интервалът на втори квадрант за x, като при нас това е [0, windowWidth / 2], се разбие на броя контролни точки. От това разбиване се получават n + 1 равни подинтервала, като n е броят на контролните точки. Всеки край на интервал, без последния, е x координатата на i-тата контролна точка на функционалната крива. По опростено чрез формула за $xi$ :

$xi$ = (windowWidth / 2) – i * ((windowWidth / 2) / n + 1), за i = 1, . . ., n

**5.** drawBezier – чрез тази функция се изчертава текущата крива на Bézier, координатната система и опционално - cross plot-а, контролния полигон и контролните точки на кривата/кривите. Функцията се извиква винаги, когато има промяна в броя на контролните точки или промяна в начина на показване (различен цвят, скриване на детайли, промяна на размера на прозореца и т.н.).

# Използвана литература

Лекции към курса за бакалаври ,,Компютърно геометрично моделиране”

# Използван софтуер

Microsoft Visual Studio - https://visualstudio.microsoft.com
The OpenGL Utility Toolkit - https://www.opengl.org/resources/libraries/glut