Изготвили:

Цветина Георгиева, СИ, 2 курс, 61696

Симона Янакиева, СИ, 2 курс, 61731

Тодор Желев, СИ, 2 курс, 61738

Графичен редактор STS

Курсова работа по Приложения на математиката за моделиране на реални процеси

Съдържание

[1 Постановка на задачата 2](#_Toc421030013)

[1.1 Различни начини за задаване на криви 2](#_Toc421030014)

[1.2 Рисуване на стандартни равнинни фигури 2](#_Toc421030015)

[1.3 Задаване на ефекти върху изображение 2](#_Toc421030016)

[1.3.1 Зареждане на изображение 2](#_Toc421030017)

[1.3.2 Селектиране на част от изображение 2](#_Toc421030018)

[1.3.3 Копиране и местене 2](#_Toc421030019)

[1.3.4 Ротация 2](#_Toc421030020)

[1.3.5 Транслация 2](#_Toc421030021)

[2 Математически основи на реализираните алгоритми 2](#_Toc421030022)

[2.1 Различни начини за задаване на криви 2](#_Toc421030023)

[2.1.1 Криви на Безие 2](#_Toc421030024)

[2.1.2 Сплайн криви 6](#_Toc421030025)

[2.2 Рисуване на стандартни равнинни фигури 6](#_Toc421030026)

[2.2.1 Окръжност 6](#_Toc421030027)

[2.2.2 Елипса 7](#_Toc421030028)

[2.2.3 Триъгълник 7](#_Toc421030029)

[2.2.4 Правоъгълник 8](#_Toc421030030)

[2.2.5 Квадрат 8](#_Toc421030031)

[2.3 Задаване на ефекти върху изображение 9](#_Toc421030032)

[2.3.1 Ротация 9](#_Toc421030033)

[2.3.2 Транслация 9](#_Toc421030034)

[3 Описание на крайния продукт 9](#_Toc421030035)

[3.1 Form 9](#_Toc421030036)

[3.1.1 Form 9](#_Toc421030037)

[3.1.2 ToolBox 9](#_Toc421030038)

[3.1.3 ColorBox 10](#_Toc421030039)

[3.1.4 Brush Size 10](#_Toc421030040)

[3.1.5 File 10](#_Toc421030041)

[3.2 Renderer 10](#_Toc421030042)

[3.3 Drawables 10](#_Toc421030043)

[3.3.1 Point 10](#_Toc421030044)

[3.3.2 Line 10](#_Toc421030045)

[3.3.3 Circle 10](#_Toc421030046)

[3.3.4 B-Spline Function 10](#_Toc421030047)

[3.3.5 Bezier Function 10](#_Toc421030048)

[3.3.6 Rectangle 11](#_Toc421030049)

[3.3.7 Triangle 11](#_Toc421030050)

[3.3.8 Square 11](#_Toc421030051)

[3.3.9 Ellipse 11](#_Toc421030052)

[3.4 Tools 11](#_Toc421030053)

[3.4.1 Eraser 11](#_Toc421030054)

[3.4.2 Selector 11](#_Toc421030055)

[3.4.3 Rotate 11](#_Toc421030056)

# Постановка на задачата

\*\* тук обясняваме какво очаква потребителя от функционалностите

## Различни начини за задаване на криви

По няколко дадени контролни точки да се построи крива

## Рисуване на стандартни равнинни фигури

По дадени 2 точки да се чертае кръг, триъгълник, квадрат, правоъгълник, елипса в зависимост от желанието на потребителя.

## Задаване на ефекти върху изображение

### Зареждане на изображение

### Селектиране на част от изображение

### Копиране и местене

### Ротация

### Транслация

# Математически основи на реализираните алгоритми

\*\* тук слагаме формули, теореми, чертежи и тн

## Различни начини за задаване на криви

### Криви на Безие

Крива на Безие е параметрична крива, често използвана в компютърната графика.

#### История

Математическата база за кривите на Безие – полиномът на Бернщайн – е познат още от 1912, но приложението му в графиката е разбрано една половин век по-късно. Кривите на Безие били широко публицирани в 1962 от френския инженер Пиер Безие, който ги използвал при дизайна да автомобилни шасита за Рено. Изследването на тези криви обаче първи разработил математика Пол дьо Кастелажо през 1959, използвайки алгоритъма на Дьо Кастелажо, числено стабилен метод за оценка на криви на Безие, в Ситроен, друга френска автомобилна компания.

#### Приложение

##### Компютърна графика

Кривите на Безие са широко използване в компютърната графика за моделиране на гладки криви. Тъй като кривата изцяло се намира в полуравнината на контролните си точки, точките могат графично да се изобразят и да се използват да manipulate кривата интуитивно.

Най-простия метод за scan converting (rasterizing) една крива на Безие е да се оцени в много близко намиращи се точки и да scan convert the approximating sequence of line segments. Това обаче не гарантира че изходното изобразяване ще изглежда достатъчно гладко, защото точките могат да са прекалено отдалечени. И обратното – може да се генерират прекалено много точки в участъзи, където кривата е близка до линейна. Често използван метод е рекурсивно подразделяне, в което контролните точки на кривата се проверяват да се види дали кривата наподобява права (използва се определена точност). Ако не, кривата се разделя на два сегмента за 0 ≤ t ≤ 0.5 и 0.5 ≤ t ≤ 1 и същата процедура се прилага рекурсивно за всяка част.

##### Анимации

В приложения за анимация като Adobe Flash и Synfig, криви на Безие се използват за описание например на движение. Потребителят очертава искания път с криви на Безие и приложението създава необходимите кадри, така че обектът да се движи по пътя.

##### Шрифтове

\*\* TODO превод

TrueType fonts use composite Bézier curves composed of quadratic Bézier curves. Modern imaging systems like PostScript, Asymptote, Metafont, and SVG use composite Béziers composed of cubic Bézier curves for drawing curved shapes. OpenType fonts can use either kind, depending on the flavor of the font.

The internal rendering of all Bézier curves in font or vector graphics renderers will split them recursively up to the point where the curve is flat enough to be drawn as a series of linear or circular segments. The exact splitting algorithm is implementation dependent, only the flatness criteria must be respected to reach the necessary precision and to avoid non-monotonic local changes of curvature. The "smooth curve" feature of charts in Microsoft Excel also uses this algorithm.

#### Обща дефиниция

##### Дефиниция

За задаване на крива на Безие от степен n са необходими:

* n + 1 на брой контролни точки **P**0, **P**1, ..., **P***n*, където *n* ∈ N. При последователното си свързване тези точки образуват контролния полигон на кривата. Отсечките **P**i **P**i+1 (*i* = 0, 1, ..., *n* − 1), свързващи две последователни контролни точки, определят т. нар. контролни рамене.
* Полиномите на Сергей Бернщайн (основни функции на Безие), означени с **B**n,i(u) . Това са полиноми на една реална променлива u, определени чрез:
* Тогава кривата на Безие **C**(u) от *n*-та степен,определена от контролните точки **P**0, **P**1, ..., **P***n*, има вида



* Параметърът *u* на кривата се изменя в затворения интервал [0, 1].

##### Рекурсивна дефиниция

\*\* TODO еднакви означения

Рекурсивната дефиниция за крива на Безие от ред n я изразява като линейна интерполация на двойка съответни точки в две криви на Безие от ред n-1.

Нека с означим кривата на Безие определена от кое да е подмножество от точки **P**0, **P**1, ..., **P***n.*



Тогава

C:\Users\Tsvetina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\99a7fa8da2d1abbcb3fb8a5723eb75bb.png



#### Терминология

\*\* TODO еднакви означения

Крива на Безие се задава по следния начин:

#### C:\Users\Tsvetina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\9581dc2354d458d372d62d2959c61d47.png

Където



Са полиномите на Бернщайн от ред n.

Точките **P***i* се наричат контролни точки за кривата на Безие. Отсечките **P**i **P**i+1 (*i* = 0, 1, ..., *n* − 1), свързващи две последователни контролни точки, определят т. нар. контролни рамене. Полигонът формиран при последователното свързване на контролните рамене се нарича полигон на Безие или контролен полигон. Изпъкналата обвивка на полигона на Безие съдържа кривата на Безие.

#### Свойства

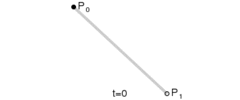
\*\* TODO превод

* The curve begins at **P**0 and ends at **P***n*; this is the so-called *endpoint interpolation* property.
* The curve is a straight line if and only if all the control points are collinear.
* The start (end) of the curve is tangent to the first (last) section of the Bézier polygon.
* A curve can be split at any point into two subcurves, or into arbitrarily many subcurves, each of which is also a Bézier curve.
* Some curves that seem simple, such as the circle, cannot be described exactly by a Bézier or piecewise Bézier curve; though a four-piece cubic Bézier curve can approximate a circle (see composite Bézier curve), with a maximum radial error of less than one part in a thousand, when each inner control point (or offline point) is the distance \textstyle\frac{4\left(\sqrt {2}-1\right)}{3}horizontally or vertically from an outer control point on a unit circle. More generally, an *n*-piece cubic Bézier curve can approximate a circle, when each inner control point is the distance \textstyle\frac{4}{3}\tan(t/4) from an outer control point on a unit circle, where *t* is 360/*n* degrees, and *n* > 2.
* Every quadratic Bézier curve is also a cubic Bézier curve, and more generally, every degree *n* Bézier curve is also a degree *m* curve for any *m* > *n*. In detail, a degree *n* curve with control points **P**0, …, **P***n* is equivalent (including the parametrization) to the degree *n* + 1 curve with control points **P'**0, …, **P'***n*+ 1, where \mathbf P'_k=\tfrac{k}{n+1}\mathbf P_{k-1}+\left(1-\tfrac{k}{n+1}\right)\mathbf P_k.
* Bézier curves have the variation diminishing property. What this means in intuitive terms is that a Bézier curves does not "undulate" more than the polygon of its control points, and may actually "undulate" less than that.
* There is no local control in degree *n* Bézier curves—meaning that any change to a control point requires recalculation of and thus affects the aspect of the entire curve—, "although the further that one is from the control point that was changed, the smaller is the change in the curve."

#### Построяване на криви на Безие

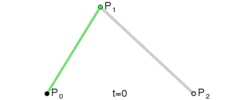
\*\* TODO

##### Линейни криви

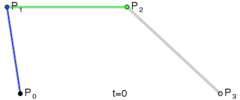


##### Квадратични криви



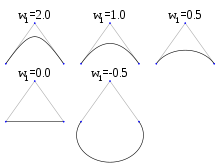


##### Криви от по-висок ред



#### Рационални криви на Безие

\*\* TODO



### Сплайн криви

В математиката под сплайн-функция или по-кратко само сплайн се разбира по части полиномиална функция. Най-прост пример за сплайн това е по части линейната функция.

#### История

За основоположници на теорията на Б-сплайните се считат Николай Лобачевски и Исак Шьонберг. Думата "сплайн" е термин от корабостроенето. Представлява тънка еластична дървена лента, използвана за изчертаване на дъги от криви. Буквата "Б" идва от "базисни" (basis splines).

#### Дефиниция

За задаване на Б-сплайн крива C(u) от степен p са необходими:

## Рисуване на стандартни равнинни фигури

### Окръжност

В приложението имаме създаден нов клас за рисуване на окръжност – class Circle. Класът е public, като той има право да се достъпва от всяко място в приложението. Създава се списък от точки.

#### Рисуване на окръжност



Взимаме координатите на центъра и радиуса на окръжността чрез мишката. Всяка точка има свои полярни координати,съставени от радиуса на окръжността и стойност на ъгъл, вариращ от 0 до 360 градуса. Самото чертаене става чрез свързване на първата точка до затваряне на окръжността чрез линия.

### Елипса

В приложението имаме създаден нов клас за рисуване на елипса – class Ellipse. Класът е public, като той има право да се достъпва от всяко място в приложението. Създава се списък от точки.

#### Рисуване на Елипса



За изчертване на елипса се използва уравнението на елипса

Където е центърът на елипсата, а е главната полуос, b е малката полуос. Първо се изчертава горната част на елипсата, след това долната, като получените точки се свързват последователно с линии

При mouse down избираме координатите на първата точка Р1, a последната точка P2 се намира при mouse up на мишката. Същестуват два радиуса, наречени малка и главна полуос. Главната полуос е с координати, които се намират като се вземе по абсолютна стойност половината от разликата между Х-координатите на Р1 и Р2. Малката полуос е с координати, които се намират като се вземе по абсолютна стойност половината от разликата между У-координатите на Р1 и Р2. Центърът се намира с координати от половината от сборовете на Х и У координатите на началната и крайната точка.

### Триъгълник

В приложението имаме създаден нов клас за рисуване на триъгълник – class Triangle. Класът е public, като той има право да се достъпва от всяко място в приложението. Създава се списък от точки.

#### Рисуване на триъгълник



За изчертаването на триъгълник ни трябва начална точка P1 с координати (a,b) и крайна точка P2 с координати (c,d). Те са координати на невидим правоъгълник, чрез който избираме мястото, където да се начертае фигурата. На картинката се вижда чрез координатите на точка Р3 са Х-координата на точка Р1 и У-координата на точка Р2. Между Р3 и Р2 се рисува линия от точки. Върхът на равнобедрения триъглъник е намерен с чрез половината от сбора Х-координатите на Р1 и Р2. У-координата е взета от началната точка Р1. Свързват се трите точки Р3, Р2 и Р4 чрез последователни линии.

### Правоъгълник

В приложението имаме създаден нов клас за рисуване на правоъгълник – class Rectangle. Класът е public, като той има право да се достъпва от всяко място в приложението. Създава се списък от точки.

#### Рисуване на правоъгълник



За изчертаването на правоъгълник са нужни координати на първата и последната точка. Р3 се получава от Х-координатата на Р1 и У-координата на Р2. Р4 се получава от Х-координатата на Р2 и У-координата на Р1. Рисуването става чрез свързване на точките чрез линии.

### Квадрат

В приложението имаме създаден нов клас за рисуване на квадрат– class Square. Класът е public, като той има право да се достъпва от всяко място в приложението. Създава се списък от точки.

#### Рисуване на квадрат



За изчертаването на квадрат са нужни координати на първата точка Р1. Последната точка има своята особеност. За да не се получи отново правоъгълник при функционалността mouseUр, която избира последната точка, се изчислява абсолютната стойност от разликата на крайната и началната точка, измерена по У-координата, наречена в нашата програма diff. По този начин крайната точка има вида – Х-координата на Р1 + diff и своята У-координата. Р3 има стойност Х-координата на Р1 и У-координата със стойност У-координата на Р1+ diff. Р4 има стойност X-координата със стойност X-координата на Р1+ diff и У-координата на Р1. Рисуването става чрез свързване на точките чрез линии.

## Задаване на ефекти върху изображение

В проекта е реализирано завъртане на област от изображение.

### Ротация

Взимаме селектирана област от точки с даден център (𝑐𝑥,𝑐𝑦).

Тъй като стандартната матрица за завъртане завърта само около началото на координатната система, а не около произволен център трябва да приложим и транслиране.

Точките от селектираната област се транслират до началото на координатната система, завъртат се на 90 градуса по часовниковата стрелка и след това се транслират обратно до началния център (𝑐𝑥,𝑐𝑦)

За да получим координатите на новата точка правим следното умножение, като използваме матрица, която комбинира ротация и транслация

Така x координата на новата точка ще бъде

а y координатата ще е

# Описание на крайния продукт

\*\* тук описваме как реализираме приложението – събития, класове, методи...

\*\* TODO

## Form

### Form UI

– форма, в която са всички бутони и място, където извършваме всички действия. Тя има своите ограничения за рисуване - размер 900х700. В програмния код се достига чрез името STS.

#### ToolBox

- от кутията за инструментите създаваме основните бутони, които са ни необходими за създаване на приложението. Всеки бутон има свои функции, които може да променяме.

#### ColorBox

- тук се намират цветовете, които може да използваме за нашите фигури, чрез които може да ги рисуваме. Имаме право да избираме на различни цветове, както и да създаваме нови, като може да ги запазваме и в бъдеще да ги използваме. Има създадено текстово име ColorBox с фон и различен font. В програмния код се достига чрез името ColorButton.

#### Brush Size

– избираме различната дебелина на линия, с която желаем да чертаем. Има три различни видове линии, създадени чрез инструмент на ToolBox-a, наречено ContextMenuStrip. Изброени са три вида - малка, средна, голяма. Достъпват се чрез натискане с ляв или десен бутон на мишката върху бутона за линии. В програмния код се достига чрез името brush.

#### File

– падащо меню, създадено чрез инструмент от ToolBoх, наречено MenuStrip. В него има създадени функции New, Open, Save, Exit. При избрана команда New формата се изчиства и отновно може да се рисува. При избрана команда Open се отваря запазен документ, който сме запазили при команда Save. При команда Exit излизаме от цялото приложение.

### Events

В приложението се използват 3 основни събития – MouseDown, MouseMove и MouseUp.

#### MouseDown

При натискане на левия бутон на мишката, командата се изпълнява и се предава на избрания Drawable обект (от InputOptions).

#### MouseMove

Осъществява се при движение на мишката, след като е натиснат и задържан бутон на мишката.

#### MouseUp

Осъщестява се след като натиснат бутон на мишката е пуснат. При пускане на левия бутон на мишката се подава известие към Renderer, който изчертава подадената информация от CurrentFigure.

### Graphics е системен обект, вграден в WFA, който се използва за чертаене върху полето.

### Renderer-a е описан в точка 3.2

### InputOptions държи информация за цвета, дебелината на линията и обекта, който трябва да се начертае.

## Renderer

## Drawables

### Point

- рисуваме точка или рисуваме линии чрез продължително натискане на левия бутон на мишката (MouseDown) чрез движение.

### Line

– чертаем линия. Използва се алгоритъма на Брезенхам. Чертаенето се осъщестява като се намерят координатите на първата точка чрез mouse down и се прекратява чертаенето при намиране на координатите на втората точка чрез MouseUp.

### Circle

– чертаем кръг. Чертаенето се осъщестява като се намерят координатите на първата точка чрез mouse down, последната точка се намира при MouseUp на мишката, като това е радиусът на кръга. Радусът се изчислява като корен-квадратен от сбора на двете разлики между координатите на точките, повдигнати на степен.

### B-Spline Function

– Б-сплайн криви. Всяка крива не е длъжна да минава през първата и последната си контролна точка, нито да се допира до първото и последното контролно рамо на контролния си полигон.

### Bezier Function

– крива на Безие. Всяка крива минава през първата и последната си контролна точка, създадени от MouseDown и MouseUp.

### Rectangle

– чертаем правоъгълник. Чертаенето се осъщестява като се намерят координатите на първата точка P1 чрез МouseDown, a последната точка P2 се намира при MouseUp на мишката.

///Третата точка Р3 взима Х- координата на Р1 и У-координатата на Р2. Четвъртата точка Р4 взима Х- координата на Р2 и У-координатата на Р1.

### Triangle

– чертаем триъгълник. Чертаенето се осъщестява като се намерят координатите на първата първата и последната точка.

///точка P1 чрез mouse down, a последната точка P2 се намира при mouse up на мишката. Третата точка Р3 взима Х- координата на Р1 и У-координатата на Р2. Четвъртата точка Р4 взима половината от сбора на Х- координата на Р2 и Р1 и У-координатата на Р1.

### Square

– чертаем квадрат.

### Ellipse

– чертаем елипса. При mouse down избираме координатите на първата точка Р1, a последната точка P2 се намира при mouse up на мишката. Същестуват два радиуса, наречени малка и главна полуос. Главната полуос е с координати, които се намират като се вземе по абсолютна стойност половината от разликата между Х-координатите на Р1 и Р2. Малката полуос е с координати, които се намират като се вземе по абсолютна стойност половината от разликата между У-координатите на Р1 и Р2. Центърът се намира с координати от половината от сборовете на Х и У координатите на началната и крайната точка.

## Tools

### Eraser

– това е гума. При mouse move и mouse down на потребител се запазват точките, през които е минал, като запазените точки се свързват последователно с линии. При mouse up цветът на тези точки и точките, генерирани от линиите, става бял, което симулира изтриване.

### Selector

– селектиране на дадена част, избрана чрез мишката.При mouse down и mouse up се запазват две точки (P1,P2), които дефинират правоъгълна област. Тази правоъгълна област от точки се запазва и се използва в другите инструменти.

### Rotate

–завъртане(ротация) на селектирана част. Селектираната част се върти 90 градуса по обратно на часовниковата стрелка. Функцията се осъщестява върху настискането на самия бутон във WFA.