ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

Rapport Projet PAP

Courbes de Bézier et polices de caractères

XU Kevin
LI Ziheng

13 janvier 2019

Sommaire

Préambule

L'objectif de ce projet est réalisé des polices de caractères en utilisant des courbes de Bézier.

— un actif sans risque : S_t^0 à l'instant t
— un actif risqué (une action) : S_t une variable aléatoire

On va utiliser une fonction $f : \mathbb{R}_+ \to \mathbb{R}_+$ tout au long du problème. Cette fonction renvoie le montant d'argent gagné pour un certain montant de l'actif risqué en paramètre.

1 Les classes (Diagramme UML)

1.1 Diagramme UML

1.2 Question 1

On a $q_N = \mathbb{Q}(T_1^{(N)} = 1 + h_N)$ donc $1 - q_N = \mathbb{Q}(T_1^{(N)} = 1 + b_N)$ car $T_1^{(N)}$ ne

2 Image

2.1 Réalisation

2.2 Solution

3 Point

3.1 Solution

4 Courbes de Bézier

4.1 Algorithme de de Casteljau

`points_` contient les points de contrôle de la courbe de Bézier et $step\_ = 0.00001$ correspond à la valeur du paramètre lors du calcul du barycentre entre deux points.

Algorithm 1 `getCasteljauPoint`

Require: $c \in \mathbb{N}$, $index \in \mathbb{N}, t \in \mathbb{R}^*$, *points* A vector which contains the control points of the Bezier Curve
Ensure: Point

```
1: function getCasteljauPoint(c, index, t, points)
2:     if c = 0 then
3:         return points[index]
4:     end if
5:     Set a Point in P1 to getCasteljauPoint(c-1, index, t, points)
6:     Set a Point in P2 to getCasteljauPoint(c-1, index+1, t, points)
7:     Set a Point in P with x = (1 − t) × (x of P1) + t × (x of P2) and y = (1 − t) ×
       (y of P1) + t × (y of P2)
8:     return P
9: end function
```

Algorithm 2 `getCurvePoints`

Require: *points* The control points of the Bezier Curve, *step* The parameter of the barycenter
Ensure: A vector *Res* of Points which picture the Bezier Curve

```
1: function getCurvePoints(points, step)
2:     Set an empty vector Res of Points
3:     Set size to the size of the vector points_
4:     for t = 0 to 1 with a step of step do
5:         Add to the vector Res the Point : getCasteljauPoint(size-1, 0, t)
6:     end for
7:     return Res
8: end function
```

4.2 Solution

5 Police 1

5.1 Réalisation

5.2 Solution

6 Police 2

6.1 Solution

img est l'image sur laquelle on a déjà dessiné les contours d'une lettre.

Algorithm 3 `colorInBlack`

Require: $x \in \mathbb{N}$, $y \in \mathbb{N}$
Ensure: Color the inside of a letter in black

```
1: function colorInBlack
2:     if The coordinates (x, y) are out of bounds then
3:         return
4:     end if
5:     if The pixel of coordinates (x, y) is white then
6:         Color the pixel of coordinates (x, y) in black on the Image img
7:         colorInBlack(x+1, y)
8:         colorInBlack(x-1, y)
9:         colorInBlack(x, y+1)
10:        colorInBlack(x, y-1)
11:    end if
12: end function
```

7 Police 3

7.1 Solution

Algorithm 4 `addRedContour`

Require:
Ensure: Add a contour of two pixels around the letter

1: **function** addRedContour
2: **for** The coordinates (x, y) are out of bounds **do**
3: **if** The pixel of coordinates $(x-1, y)$ is white or red **then**
4: Color the pixel of coordinates $(x-1, y)$ in red on the Image img
5: Color the pixel of coordinates $(x-2, y)$ in red on the Image img
6: **end if**
7: **if** The pixel of coordinates $(x+1, y)$ is white or red **then**
8: Color the pixel of coordinates $(x+1, y)$ in red on the Image img
9: Color the pixel of coordinates $(x+2, y)$ in red on the Image img
10: **end if**
11: **if** The pixel of coordinates $(x, y-1)$ is white or red **then**
12: Color the pixel of coordinates $(x, y-1)$ in red on the Image img
13: Color the pixel of coordinates $(x, y-2)$ in red on the Image img
14: **end if**
15: **if** The pixel of coordinates $(x, y+1)$ is white or red **then**
16: Color the pixel of coordinates $(x, y+1)$ in red on the Image img
17: Color the pixel of coordinates $(x, y+2)$ in red on the Image img
18: **end if**
19: **end for**
20: **end function**