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 $ratio\_ = 0.00001$ correspond à la valeur du paramètre lors du calcul du barycentre entre deux points.

Algorithm 1 `getCasteljauPoint`

Require: $c \in \mathbb{N}$, $index \in \mathbb{N}, t \in \mathbb{R}^*$
Ensure: Point
1: **function** GETCASTELJAUPOINT(c, $index$, t)
2: **if** c = 0 **then**
3: **return** points_[index]
4: **end if**
5: Set a Point in P1 to `getCasteljauPoint(c-1, index, t)`
6: Set a Point in P2 to `getCasteljauPoint(c-1, index+1, t)`
7: Set a Point in P with $x = (1 - t) \times (x\ of\ P1) + t \times (x\ of\ P2)$ and $y = (1 - t) \times (y\ of\ P1) + t \times (y\ of\ P2)$
8: **return** P
9: **end function**

Algorithm 2 `getCurvePoints`

Require:
Ensure: A vector *Res* of Points
function GETCURVEPOINTS
Set an empty vector *Res* of Points
Set S to the size of the vector points_
for $t = 0$ to 1 with a step of *ratio*_ **do**
Add to the vector *Res* the Point : `getCasteljauPoint(size-1, 0, t)`
end for
return *Res*
end function

4.2 Solution

5 Police 1

5.1 Réalisation

5.2 Solution

6 Police 2

6.1 Solution

7 Police 3

7.1 Solution