Anne-Elisabeth Lelièvre 16 084 130

Olivier Beauséjour 16 099 883

Devoir en Processus concurrents et parallélisme

Projet de session

Rapport de travail

Dans le cadre du cours IFT630

Présenté à :

Gilles Brunet

Université de Sherbrooke

10 août 2018

**Table des matières**

[1. Présentation de l’application 2](#_Toc520724982)

[2. Démarche et recherches 2](#_Toc520724983)

[3. Description du mode de déploiement 2](#_Toc520724984)

[4. Références 3](#_Toc520724985)

# Présentation de l’application

Nous avons décidé de faire un projet pratique basé sur un traitement parallèle pour la recherche d’images par le contenu. Le but principal de notre projet est de faire un analyseur d’image de pixel. Le cœur du programme est que nous avons des images de références d’une table de cartes dans le dossier ressources et notre analyseur doit déterminer la nature de nos 2 cartes dans notre main. Notre analyseur détermine seulement que le chiffre de la carte.

# Démarche et recherches

Pour nos recherches, nous avons un intérêt commun pour les jeux de cartes. De plus, après avoir complété le cours d’intelligence artificielle, nous avons acquis des connaissances sur la méthode des k plus proches voisins. Cette approche est une méthode d’apprentissage supervisé. Pour estimer la sortie associée à une nouvelle entrée x, la méthode des k plus proches voisins consiste à prendre en compte les k images échantillons d’apprentissage dont l’entrée est la plus proche de la nouvelle entrée x. Ainsi, nous considérons la distance entre chaque pixel de l’image, ce qui nous donne le score pour l’image. Les images ayant le score minimal seraient plus proche des images de références ce qui nous permettrait de déterminer quels sont les 2 cartes dans notre main. Un peu comme le devoir 2 avec le *BruteForceAttaque*, nous avons fait une version *SingleThread* et une version *MultiThread*. Nous avons omis la version GPU étant donné la contrainte de temps.

# Description du mode de déploiement

Au sujet du découpage de notre code, nous avons tout d’abord fait une classe *Analyseur.cs* qui a la logique de l’analyseur de nos images. C’est dans cette classe que nous paralysons le calcul de distance de pixel. La classe *Carte.cs* contient tous les caractéristiques propres à une carte (sa valeur et son type). La classe *Constantes.cs* contient les coordonnées en pixel exacte de la main de nos images de références. Ces coordonnées nous servent à bien analyser pour une même dimension chaque pixel. La classe *Position.cs* contient les coordonnées d’un pixel dans une image. Les classes *Program.cs* et *frmMain.cs* contiennent l’ouverture de l’application permettant le choix d’exécuter le programme soit en séquentiel ou en multiple fils. Nos images de références tels que la table de jeux de cartes et des images de références des nombres sont répertoriées dans le dossier *Ressources*.

Au sujet de la multi fils, nous avons utilisé un *task.run()* pour que chaque processus exécute le calcul des distance de pixel. Par contre, nous avons rencontré un problème lors d’une modification d’une de nos variables lors des calculs des lignes. Il a donc fallu corriger ce problème en ajoutant un lock pour garantir qu’un thread n’entre pas dans cette section critique du code. Alors si un autre thread tenterait d’entrer dans un code verrouillé, il attend, en restant bloqué jusqu’à ce que l’objet soit libéré.

# Références

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/lock-statement>

<https://www.ggpoker.com/poker-card-games/special-features/>

<https://msdn.microsoft.com/en-us/library/hh195051(v=vs.110).aspx>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/unsafe-code-pointers/pointer-types>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries>

<https://msdn.microsoft.com/fr-ca/library/dd492132.aspx>

<https://msdn.microsoft.com/en-us/library/ms182351%28VS.80%29.aspx>

<https://msdn.microsoft.com/en-us/library/system.drawing.bitmap(v=vs.110).aspx>

<https://msdn.microsoft.com/en-us/library/5ey6h79d(v=vs.110).aspx>