DUT informatique de gestion

Année Universitaire 2010-2011

Rapport de projet de fin d'année

Le jeu des pousses

Etudiants
Martin Nathanaël
Meresse Lucie

Tuteur de projet
M. Champ Julien

Table des matières

Remerciements

Tout d'abord, nous tenons à remercier notre tuteur de projet Julien Champ qui a toujours été là pour répondre à nos questions et nous éclairer.

Nous remercions Mathieu Virbel qui a pris le temps de répondre à nos nombreuses questions concernant le framework Kivy, et pour la relecture de ce rapport.

Nous remercions les développeurs et membres actifs de la communauté Kivy (Christopher Denter) qui ont su répondre à certains de nos questionnements rapidement.

Résumé

Le « Sprouts game », aussi appelé le jeu des pousses, est un jeu qui, habituellement, se joue à deux joueurs sur une feuille de papier et utilisant des propriétés mathématiques. Avec ce projet, nous avons rendu ce jeu jouable sur ordinateur (aussi bien à la souris, qu'au doigt). En effet, nous avons utilisé le langage Python[1] et le framework[2] Kivy[3] pour rendre ce jeu disponible en multitouch[4]. Aussi, l'intégralité des sources sont sous la licence GPL[5] et disponibles sur le site github.com[6]

En début de partie, le joueur peut choisir le nombre de nœuds qui seront positionnés aléatoirement à l'écran. Chacun des joueurs, relie deux points existants à tour de rôle. Ce tracé crée automatiquement un troisième nœud centré sur le tracé. Certaines contraintes sont à respecter comme par exemple le fait que les lignes ne peuvent pas se croiser (aussi bien au moment du tracé que dans le nœud lui-même), les nœuds ne peuvent pas se superposer et un nœud ne peut posséder, au maximum, que 3 degrés.

Il faut savoir qu'au minimum, il y a 2 *n coups jouables et au maximum (3*n)-1 coups possibles. De plus, selon le mode de jeu choisit (misère ou normal), le gagnant est différent. Dans le mode normal, le joueur qui joue le dernier coup est déclaré gagnant alors que dans le mode misère le dernier joueur est déclaré perdant.

D'un point de vue personnel, ce projet nous a permis d'utiliser un langage non enseigné à l'IUT et d'apprendre à utiliser de nouveaux outils. De plus, il nous a fallu faire preuve d'autonomie et d'esprit critique avant de mener à bien notre projet tuteuré.

1 Python: http://python.org/
2 Bibliothèque de classes et fonctions, prévus pour être dérivées et étendues, répondant au besoin du développeur.
3 Kivy : http://kivy.org/
4 Multitouch : interactions avec plusieurs doigts
5 GPL : http://www.gnu.org/licences/gpl.html
6 Service web d'hébergement et de gestion de code source, basé sur le programme Git. Github propose des comptes professionnels payant, ainsi que des comptes gratuits pour les projets open source

Abstract

The Sprouts game is a two-player game that uses mathematical properties. It is usually played on paper. Now, through this project, it can be played on a computer, with either mouse or fingers! In fact, we have used the Python language and the Kivy framework to be able to program the Sprouts game in multitouch mode. All sources are licensed through GPL and available on github.com.

At the beginning of the game, the players can choose how many nodes will be randomly positioned on the screen. Each player must link two existing points in turn: this action will automatically create a third node centered halfway between these 2 nodes. Some rules must be observed such as lines that cannot cross one another, each node can have at most 3 lines starting and nodes cannot overlap. With these rules, there is a minimum of 2*n possible moves and a maximum of 3* n-1 possible moves. According to the game mode chosen (“misère” or normal mode), the winner is different. In normal play, the player who makes the last move wins while in « misère » mode, the player who makes the last move loses.

From a personnal point of view, this project give us the possibility to learn a new language from our own, and learn new coding tools.

Introduction

Dans le cadre de notre DUT informatique de gestion à l'Institut Universitaire de Technologie de Lille, nous avons été amené à réaliser un projet de fin d'année en binôme. Notre projet, le Sprouts-Game, est un jeu simple qui permet à deux joueurs de s'affronter sur la même machine ou en réseau et qui utilise de nombreuses propriétés mathématiques.

Lors du choix des sujets, nous avons vu à travers le Sprouts Game la possibilité de réaliser une interface proche des attentes des joueurs d'aujourd'hui. Certaines problématiques étaient alors à résoudre :

➔ Comment réaliser une interface de jeu agréable ?

➔ Comment susciter l'intérêt des joueurs ?

➔ Comment être au plus proche du cahier des charges tout en respectant les délais ?

Pour répondre à la première question, nous avons réfléchi longuement au choix du langage et nous avons opté pour Python. Nous avons essayé de déterminer les attentes des joueurs actuels en prenant en compte les avancées technologiques comme les smart phones[7] ou encore les tablettes tactiles et nous en sommes arrivés à l'utilisation du Framework Kivy qui intègre nativement le multitouch.

Dans une première partie, nous procèderons à une présentation du jeu, des contraintes qui ont dû être respectés ainsi que nos choix techniques. Nous verrons dans une seconde partie le développement en lui-même, avec l'explication de nos choix, l'évolution de notre travail au cours des semaines et les améliorations envisageables pour l'avenir. Enfin, nous établirons un bilan technique et humain propre à chacun des membres du binôme.

7 Téléphone intelligent

1. Présentation du jeu

1.1. Histoire et règles du jeu

Le jeu des pousses est un jeu qui se joue habituellement à deux joueurs sur une simple feuille de papier. Inventé à Cambridge en 1967 par Michael Paterson et John Horton Conway, il utilise de nombreuses propriétés mathématiques comme la théorie des graphes.

C'est en juillet 1967 que le jeu des pousses à connu une certaine notoriété du point de vue de la communauté scientifique grâce à une surexposition dans des revues mathématiques, notamment celle de Martin Gardner.

Comme dit précédemment, le jeu des pousses se joue à deux joueurs. L'état initial du jeu consiste à définir un nombre fini de nœuds qui seront placés aléatoirement sur le plateau de jeu en début de partie. C'est à tour de rôle que les joueurs relierons deux sommets via une ligne. Cette action aura pour conséquence de diminuer de un le nombre de liberté des sommets et de créer un nouveau nœud au centre du chemin qui aura un nombre de liberté réduit et fixé à 1.

Bien entendu, certaines contraintes sont à respecter. Aucun croisement de ligne n'est autorisé dans le Sprouts Game. Un sommet peut être soit point de départ, soit point d'arrivée de la ligne ou alors les deux à la fois puisqu'un tracé de ligne sur lui-même est autorisé. Cependant, dès qu'une ligne relie deux sommets, ces derniers voient leur degré de liberté diminuer de un. Ainsi, un sommet ne peut avoir que 3 lignes connectées sur lui-même.

De plus, il est intéressant de savoir qu'il y a au minimum 2*n coups qui peuvent être joué et au maximum 3*n-1 coups avec n le nombre de nœuds défini en début de partie par les joueurs.

1.2. Organisation et répartition des tâches

Dès la première semaine de projet, nous avons consacré énormément de temps à l'étude et à l'analyse du sujet. Il en est ressorti que plusieurs solutions à un seul et même problème donné étaient envisageables. On peut citer à titre d'exemple, la partie réseau qui pouvait se faire grâce au protocole UDP ou au protocole TCP. Concernant la représentation des nœuds au cours du jeu, diverses propositions ont été étudié aussi. Nous expliquerons nos choix dans la suite du rapport. Il a donc fallu faire le tri et peser les plus et les moins des idées proposées. Afin d'avancer au plus vite et au mieux, nous avons décidé de nous répartir les tâches à accomplir. Pour gérer cette méthode de travail, nous avons utilisé le site de dépôt github, que nous présenterons dans la sous partie suivante, qui s'est révélé être un outil indispensable.

Nous avons aussi établi un cahier des charges qui a été notre fil rouge durant notre développement. En effet, avant de commencer le jeu, le joueur se retrouvera face à un menu lui présentant 4 possibilités : jouer, configurer le jeu (mettre la musique ou pas, mode de jeu, etc.), accéder aux scores et quitter le jeu. Le joueur peut choisir le nombre de point de départ via un slider[8]. Ces nœuds seront placés de façon aléatoire sur l'écran à la création de la partie. Ces derniers ne peuvent évidemment pas se superposer.

Le joueur à la possibilité de tracer lui-même les lignes reliant les sommets au doigt ou à la souris, il a aussi la possibilité de choisir le mode jeu pour le reste de la partie (normal ou misère) à partir de l'écran « settings » du menu d'accueil. Après le tracé de la ligne et sa validation, un nouveau nœud est crée automatiquement au milieu de la ligne précédemment tracée.

Après avoir appris qu'il existait un nombre minimal de coups, nous avons décidé de ne déclencher que certaines vérifications à partir de 2*n coups (avec n le nombre de nœud de départ) afin d'optimiser le temps de calcul. Les sommets ne peuvent avoir qu'un degré allant de 0 à 3, 0 étant un sommet isolé et 3 un sommet saturé. Les lignes tracées par le joueur ne doivent pas se croiser, aussi bien lors du tracé que dans les nœuds. Elles possèdent une longueur minimale et elles ont obligation de relier deux sommets pour être validée. Il faut aussi savoir qu'un même sommet peut être relié par la même ligne. Une ligne ne peut pas passer au travers d'un sommet si elle ne le relie pas.

Et pour finir, il faut programmer une intelligence Artificielle (I.A) capable de jouer une partie avec un joueur réel.

1.3. Outils principaux utilisés

a. Gestionnaire de version

Nous avons réalisé le Sprouts-Game sous un environnement Linux : Ubuntu. Afin de faciliter l'échange du code source au cours du développement, avec des conseils extérieurs, nous avons utiliser Git, un gestionnaire de code source décentralisé, au travers du site web Github. Nous avons appris le workflow de Git, avec ces clone, commit, push, pull. Github publiait en temps réel nos commits, permettant de voir et réaliser l'avancée du projet, ce qui est un réel avantage quand le travail est partagé entre les deux membres de l'équipe.

Pour un utilisateur lambda qui souhaite récupérer l'intégralité des sources de notre projet, il lui suffit de « cloner » notre projet via la commande :

```
git clone git://github.com/Dexedrine/Sprouts-Game.git
```

On se retrouve avec un répertoire Sprouts-Game, contenant la dernière version du code source de notre projet.

8 Une barre de défilement

b. Éditeur de texte

Nous avons décidé de ne pas utiliser d'IDE[9] comme Eclipse mais plutôt les éditeurs gedit et gvim. Ce dernier est la version graphique de « vi » qui nous avait été présenté assez succinctement lors des enseignements de programmation système en première année de DUT

c. UML et Chronographes

Pour la réalisation des schémas UML de notre projet, nous avons utilisé un outil open source nommé Gaphor. Celui ci permet d'importer nos fichiers Python, d'en analyser les sources et d'afficheer le schéma UML des ces dernières. Néanmoins, la mise en page a été refaite à la « main ».

Concernant l'enchainement des écrans, nous avons utilisé graphviz (en particulier dot), qui permet la création de graphe connecté à partir d'un fichier texte et d'une syntaxe particulièrement simple à écrire.

Les chronographes réseau ont été généré avec un outil open source nommé mscgen. Il ressemble de très près à graphviz, mais génère des chronographes.

d. Langages

Concernant le choix du langage Python et du framework Kivy, nous en parlerons dans la seconde partie de ce rapport.

1.4. Conclusion partielle

Grâce à notre expérience personnelle, acquise lors du projet de fin du semestre 2 et via des projets personnels, nous avons appris à ne pas nous jeter tête baissée dans un projet. L'analyse du jeu a été un point crucial dès le début et il a fallu faire preuve d'un bon esprit critique, de synthèse et surtout de réalisme. Nous avons séparé le projet en plusieurs sous problèmes afin de pouvoir affecter à chacun des membres de l'équipe une tâche particulière. Cette méthode de travail nous a été bénéfique et nous a permis d'avancer assez rapidement sur le jeu des pousses.

9 Integrated Developpement Editor : application facilitant la saisie de code pour des « gros » projets

2. Réalisation du projet

2.1. Analyse et objectifs

L'objectif principal de ce projet était de rendre ce jeu attractif, simple et agréable à prendre en main.

Durant la réflexion, nous nous sommes posés des objectifs tels que :

- Réaliser le projet en multitouch : nous avons étudier et écouter les attentes des joueurs et il en est ressorti qu'aujourd'hui, c'était le multitouch qui arrivait en tête de liste.
- Publier le jeu sur l'Android market : nous pensons qu'un jeu qui se joue avec les doigts, et simple d'utilisation, peut avoir sa place sur l'Android Market.
- Apprendre un nouveau langage : Python.
- En bien sûr, respecter les objectifs des consignes : différencier les sommets par des couleurs, indiquer le nombre de degré au centre des noeuds, limiter le temps de jeu de chaque joueur et faire la partie réseau serveur / client.

Pour la réalisation du jeu en multitouch, nous avons décidé d'utilisé un Framework open source nommé Kivy. Celui ci nous impose la programmation de notre jeu dans le langage Python.

2.2. Choix techniques

a. Langage Python

L'opportunité d'apprendre un langage en autodidacte tel que Python était un véritable défi, et nous y avons vu un moyen de perfectionner nos connaissances en programmation objet, mais aussi de nous démarquer par rapport aux autres binômes qui ont sûrement choisi Java.

Python est un langage à typage dynamique : la déclaration des variables et de leur types est faite à la première assignation. A la manière de Java, Python est un langage compilé, dont son bytecode est exécuté dans une machine virtuelle. Néanmoins, nous apprécions le fait que la compilation se fasse à la volée. Cela lui procure la facilité d'usage d'un langage interprété, c'est à dire que nous n'avons pas à attendre la fin de la compilation. Nous avons été un peu dérouté par un côté explicite que nous n'avions pas en Java : dans une classe en java, nous pouvons omettre le « this. » lorsque nous faisons référence à une variable appartenant à la classe. Avec Python, nous devons explicitement mettre « self. ».

b. Framework Kivy

Au cours du semestre 3, nous avons été sensibilisé à l'utilisation de framework et nous avons pu mettre en pratique cet enseignement à travers l'utilisation de Kivy.

Kivy est un framework open source qui permet de développer des interfaces dites NUI[10]. On y voit par NUI une nouvelle approche de l'interface utilisateur, notamment une interface qui semble naturelle, et utilisable avec les doigts. Ce framework intègre donc nativement la gestion du multitouch, ce qui lui offre de nombreuses possibilités de conception. Les applications faites avec Kivy peuvent être exécutées sur Windows, Linux, MacOSX et Android.

Kivy est la suite de PyMT, un ancien framework multitouch. Au moment du projet, Kivy n'était pas encore sorti officiellement, néanmoins, nous en avons profiter pour comprendre son utilisation, et contribuer à sa stabilité : nous avons appris à reporter des bugs et améliorations, notamment sur sa documentation.

c. Réseau

Concernant la partie réseau, notre choix s'est vite porté sur le protocole TCP[11] à l'instar du protocole UDP[12]. En effet, le protocole TCP est un protocole assez fiable et fonctionnant en mode connecté. L'existence de certains mécanismes clefs afin d'assurer la bonne réception des lignes tracées et des points crées nous ont rapidement convaincu. Ces deux protocoles ont été vu lors de l'enseignement réseau réalisé au semestre 2 et 4.

2.3. Choix interfaces

Afin d'atteindre l'un de nos objectifs qui était de rendre ce jeu agréable, simple et intuitif, nous avons réfléchi au problème de la présentation du jeu. Ainsi, nous avons convenu de la création d'un menu. Ce dernier propose au joueur 4 possibilités vu auparavant [13]. Après avoir cliquer sur le bouton « play », le joueur se retrouve face à 3 choix possibles : jouer sur sa machine, jouer via le réseau local en étant le serveur ou jouer via le réseau local mais en étant le client de la partie. Si le joueur décide de jouer seul sur sa machine, il a alors la possibilité de choisir le nombre de nœud de départ qui seront placés aléatoirement. Si le joueur décide d'être le serveur, il a aussi la possibilité de choisir le nombre de nœud crée puis apparaît un écran d'attente de connexion de la part du client. Si le joueur décide d'être le client, alors il doit renseigner l'adresse IP du serveur (par défaut 127.0.0.1). Il est alors connecté et redirigé automatiquement vers l'écran de jeu.[14]

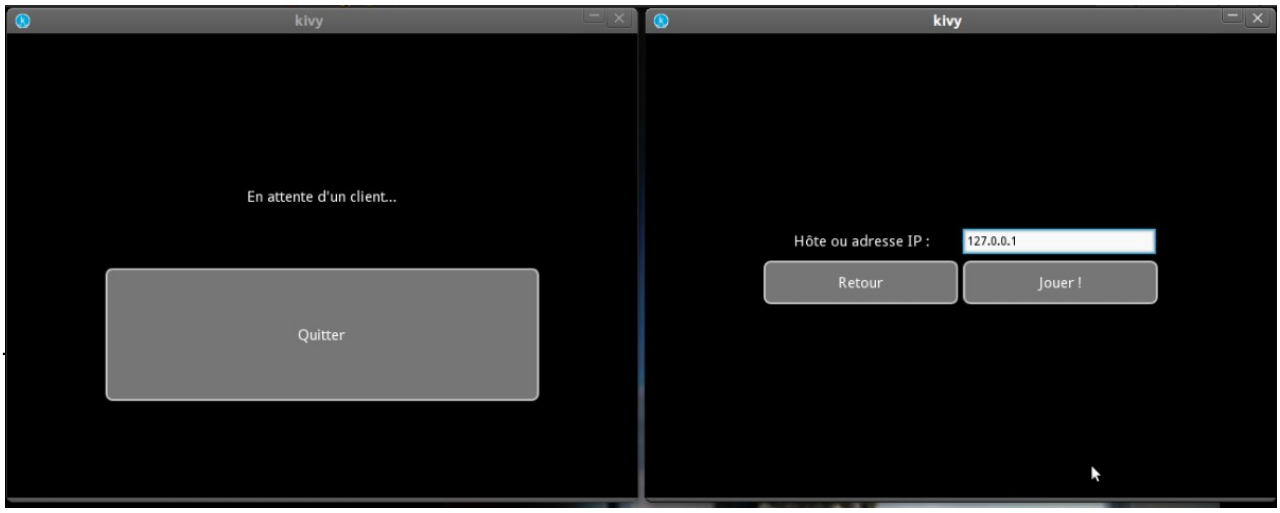


Illustration 1: A gauche le serveur, à droite le client

Aussi, la représentation des points est différentes en fonction du nombre de degré du noeud : violet quand le degré est 0, rouge quand le degré est 3.

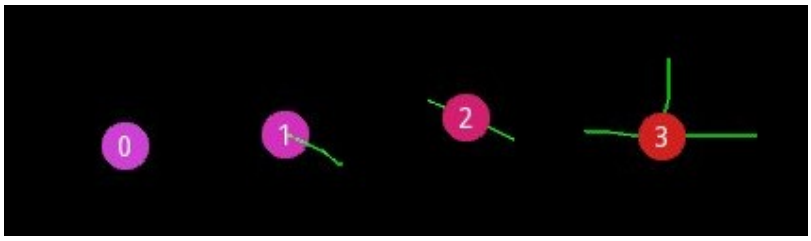


Illustration 2: Evolution de la couleur des points selon leur degré

2.4. Algorithme de collisions entre lignes

a. Méthode brute

L'algorithme utilisé pour la collision entre les lignes s'est fait en deux étapes.

La première version de l'algorithme était représenté comme ceci :

```
# Algorithme de test entre chaque lignes
Soit l1 la ligne à vérifier
Pour chaque l2 dans les lignes affichées:
    Si l1 == l2 :
          On continue la boucle
    Test de l'intersection entre l2 et l1
    → Appel de is_intersect(l1, l2)

# Fonction de test d'intersection entre deux lignes
Fonction is_intersect(l1, l2)
Pour i dans la liste de points de l1 :
    Récupère 2 points en partant de i dans l1 (A, B)
    Pour j dans la liste de points de l2 :
          Récupère 2 points en partant de j dans l2 (C, D)
          Test intersection entre les 4 points (2 segments)
          → Appel de intersect(A, B, C, D)

# Fonction de test d'intersection entre deux segments
def ccw(A,B,C):
  return (C[1]-A[1])*(B[0]-A[0]) > (B[1]-A[1])*(C[0]-A[0])

def intersect(A,B,C,D):
```

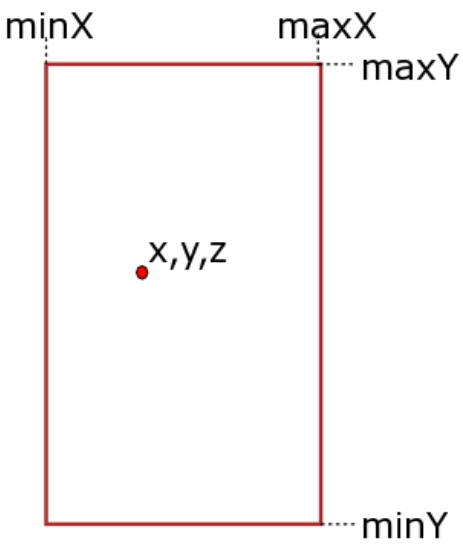
```
return ccw(A,C,D) != ccw(B,C,D) and ccw(A,B,C) != ccw(A,B,D)
```

La méthode d'intersection entre 4 points / 2 segments provient de Bryce Boe, dans son article nommé *Line Segment Intersection Algorithm* en 2006. Il explique une méthode d'intersection basée sur le calcul de l'orientation des deux segments.

A la vue de cet algorithme, nous avons compris que plus il y avait de ligne, plus le temps de vérification serait long.

b. Optimisation à l'aide de Bounding Box

Les bounding box sont des objets qui sont très importants dans nos algorithmes, plus particulièrement dans l'algorithme qui détecte les croisements de ligne.



Avant d'aller plus loin il est nécessaire d'expliquer ce qu'est une bounding box :

Lorsque l'on vérifie une ligne, on calcul en premier sa bounding box correspondante, en prenant le minimum et maximum de chaque point.

Ainsi la ligne tracée sera encadrée par un rectangle virtuel qui englobe sa forme entièrement.

Le calcul de la bounding box se fait avec un algorithme comme celui ci :

```
xmin = ymin = xmax = ymax = 0
Pour chaque x et y dans les points constituants la ligne :
    xmin = min(xmin, x)
    xmax = max(xmax, x)
    ymin = min(ymin, y)
    ymax = max(ymax, y)
```

Pour en revenir à notre algorithme sur le croisement de ligne, nous pouvons inclure un test de collision avec les bounding box de chaque ligne. Si celui ci réussi, c'est seulement à ce moment que nous effectuerons la collision détaillée de chaque segment :

```
# Algorithme de test entre chaque lignes
Soit l1 la ligne à vérifier
Pour chaque l2 dans les lignes affichées:
    Si l1 == l2 :
```

```
        On continue la boucle
    Test entre la bounding box de l1 et l2
    Si le test échoue :
        On continue la boucle
    Test de l'intersection entre l2 et l1
    → Appel de is_intersect(l1, l2)
```

2.5. Création d'un protocole réseau

Nous avons d'abord développé la partie locale avant de développer la partie réseau.

Même si nous avons tenté de faire attention à la séparation des classes et méthodes pour une future utilisation réseau, cela n'a pas été suffisant. Notre code initial était trop lié avec le jeu local, et pas assez abstrait. Par exemple, la validation de la ligne nécessitait des informations que l'on calculait lors du tracé. Ces informations étaient pourtant nécessaire à la validation de la ligne dans son ensemble. Nous avons donc effectué quelques refactorisations pour permettre l'utilisation de la fonction de validation dans un environnement réseau.

Nous avons établi un protocole basé sur les principes suivants :

- Le serveur agit comme une autorité de validation : c'est lui qui est en charge de vérifier les informations envoyés par le client.
- Le client agit comme un esclave, il ne fait qu'exécuter ce que le serveur lui envoye (message, ligne, noeud).

Nous avons séparer le protocole en 4 phases[15] :

- Initialisation du jeu
- Boucle de jeu
- Messages
- Vérification de timeout

Nous avons établi un protocole « texte », dans le format suivant :

```
<COMMANDE> <options>\n
```

Par exemple, voici la liste des commandes disponibles dans la boucle de jeu, coté serveur vers client :

`LIGNE <x;y ;..;xn;yn>\n`	Ajout d'une ligne dans le client

15 Voir annexe 3

`NOEUD <x;y>\n`	Ajout d'un noeud dans le client
`MSG <message>\n`	Affichage d'un message
`FAIL <message>\n`	Affichage d'un message d'erreur, et demande au client de rejouer

2.6. Problèmes rencontrés et solutions

a. Mode « debug »

Au cours de notre projet nous avons créé un mode « debug » qui nous a permis de faire évoluer notre projet en détectant plus facilement les erreurs, ainsi on peut y trouver plusieurs ajouts par rapport à la version normale :

- L'apparition en transparence des bounding box
- Les points créés lorsque l'on trace une ligne
- Le nombre de sommet disponible qui sera affiché sur le sommet lui-même. (gardé dans la version finale)

Ce mode est nécessaire pour bien comprendre comment fonctionne notre programme, on voit plus facilement les traces d'exécution , le tout en graphique.

Pour lancer ce mode, la commande python suivant doit être écrite en console depuis le répertoire contenant le projet :

```
$ python Sprouts.py --debug
```

b. Gestion du temps

Il faut aussi savoir, que tout nos objectifs n'ont pas été atteint, principalement par manque de temps. En effet, l'IA n'a pas été réalisé. L'algorithme de fin de partie non plus.

Cependant nous y avons longuement réfléchi et nous pensons que la théorie des graphes et plus particulièrement les chaines eulériennes entre en compte. Nous avons aussi penser à utiliser une technique appelé l'algorithme du peintre autour des noeuds restants.

La partie « configuration » du jeu n'a pas abouti même si la bande son du jeu a été réalisé. Et enfin, la gestion des scores n'a pas été programmé. Concernant ce dernier point, il faut savoir que nous avons une petite idée sur la question puisque nos

recherches nous ont dirigé vers la création d'une base de données web, il y a aussi nécessité de création de compte joueur avec pseudo et mot de passe pour authentification.

2.7. Conclusion partielle

Au cours du développement nous nous sommes heurtés à des problèmes que nous n'avions pas prévu lors de notre analyse de la première semaine. Nous avons donc dû nous adapter rapidement. Nos recherches ont portés leurs fruits puisque nous avons découvert le principe des bounding box qui s'est révélé être un atout particulièrement intéressant pour la réactivité et la rapidité de traitement de notre programme. Nous avons aussi développé un mode « debug » uniquement utilisé pour la compréhension et la détection d'erreur couplé à des affichages en console.

C'est donc à l'aide de développement d'outils « personnels » et à l'application de principe que nous avons réussi à avancer au mieux dans le développement du jeu des pousses tout en restant proche des objectifs que nous nous étions fixés.

3. Bilans technique et humain

3.1. Bilan technique

Concernant Nathanaël

Ce projet tuteuré m'a permis d'améliorer mon esprit d'analyse, en effet j'ai du bien analyser le problème avant de me lancer dans la partie codage, on a dû réaliser un diagramme UML et de mon côté j'ai dû apprendre le langage python qui était tout nouveau pour moi. De plus j'ai étudié le Framework Kivy afin de voir quelle solution il pouvait nous apporter afin de programmer l'application. Les règles du jeu, en apparence simple, ont été difficiles à coder parfois, par exemple pour le croisement de ligne et les autres vérifications ainsi que les optimisations à apporter, comme par exemple les bounding box, dont le principe a été expliqué précédemment.

Concernant Lucie

En choisissant de réaliser ce projet en Python, j'ai appris une nouvelle syntaxe qui, comme dit précédemment, est parfois plus explicite que celle de Java ou d'autres langages. Prenons par exemple l'instance de la classe représentée par « self » en python et par « this » en java. En Java, le « this » est implicite et l'IDE Eclipse l'ajoute automatiquement. En passant à l'utilisation de Python, j'ai rencontré, au début, un grand nombre de problèmes lors du lancement de l'application dû à l'oubli de ce fameux « self ».

Cette réalisation m'a permis de confirmer mes compétences en programmation objet. De plus, en programmant la partie réseau, j'ai pu mettre en application une partie des enseignements des semestres 2 et 4 avec par exemple l'utilisation des threads, les chronographes, le choix du protocole, etc.

J'ai aussi pu mettre en pratique l'apprentissage de l'utilisation d'un framework enseigné au semestre précédent. J'ai dû apprendre à prendre en main une nouvelle documentation, à comprendre l'anglais et il m'a aussi fallu vaincre ma peur de m'exprimer en anglais sur un chat afin de pouvoir signaler les bugs de Kivy, d'ouvrir des rapports d'erreurs sur le site github mais aussi de pouvoir poser des questions et demander des précisions sur certaines méthodes.

3.2. Bilan personnel

Concernant Nathanaël

Durant ce projet, nous avons dû nous documenter sur le framework Kivy qui est en développement. Ainsi nous devions nous tenir informer des changements constamment. De plus, il est impératif de communiquer avec l'équipe de développeur de ce framework afin d'être force de proposition pour l'évolution de Kivy.

Concernant Lucie

A la fin du semestre 3, nous avons réaliser un « mini »projet en binôme. Ce mode de travail est quelque chose que j'apprécie et qui me convient. J'ai d'ailleurs renouvelé l'expérience avec le même binôme pour un projet personnel. Cependant, avec le Sprouts-Game j'ai rencontré certaines difficultés.

Étant de nature stressée, j'ai dû apprendre à gérer mon propre stress mais j'ai aussi dû apprendre à gérer et à comprendre les angoisses et les peurs de mon binôme ce qui a été un exercice assez difficile. Cette situation a souvent été à l'origine de conflits et d'incompréhension de la part de chacun des membres du binôme.

De plus, même si ce sujet semblait simple de prime abord, il est apparu que certaines contraintes pouvaient être réglées via différentes solutions. Il a donc fallu s'accorder sur ces idées.

Globalement, ce projet s'est révélé extrêmement intéressant d'un point de vue technique mais aussi humain. Il a fallu affronter notre stress et nos angoisses face au temps qui est passé très vite. Nos envies et choix initiaux n'ont pas tous pu être respecté, il m'a été très difficile d'évaluer le temps nécessaire au développement de certaines partie du jeu. Cela m'a appris plus précisément à comprendre que les tâches en apparence simple peuvent être celles qui prennent le plus de temps.

Aussi, je suis maintenant capable de demander de l'aide et de poser des questions sur un chat en anglais et d'ouvrir des rapports d'erreurs afin d'améliorer les outils que nous avons utilisé au cours de ce projet.

3.3. Améliorations

Même si le temps consacré pour ce projet est terminé, il n'est pas question pour nous d'en rester là. En effet, notre objectif principal est de rendre ce jeu disponible sur l'Android Market. L'équipe de Kivy s'est réjouit à la prise de connaissance de notre projet : ce jeu sera alors le premier jeu utilisant le framework Kivy publié sur l'Android Market. Si tout se passe bien, il serait alors intégrer dans les sources de cet outil en tant qu'exemple d'utilisation.

Afin de rendre réalisable cette opportunité, nous allons :

1. Commenter au mieux et en anglais l'intégralité des fonctions de notre programme
2. Terminer et peaufiner ce projet en développant l'algorithme de fin de partie
3. Inclure la gestion des scores (secondaire)
4. Inclure la configuration du jeu par le joueur avant de commencer la partie
5. L'intégration d'un bouton « quitter » lors de la partie
6. L'intégration d'un chat pour la communication entre les joueurs
7. L'affichage d'une barre d'état indiquant au joueur le temps qu'il lui reste pour jouer son coup
8. Faire que le joueur puisse voir en direct la création de la ligne de son adversaire (ceci nécessite une amélioration du protocole)
9. Tester et peut-être modifier l'apparence du Sprouts-Game pour le rendre jouable sur smart phone, tablettes tactiles et ordinateurs dans les meilleures conditions possibles pour les futurs joueurs.

3.4. Conclusion partielle

Pour conclure cette partie, nous pouvons aujourd'hui affirmer que ce projet nous a été bénéfique, aussi bien d'un point de vue technique qu'humain. Il nous a permis de nous sensibiliser à un framework en développement et de pouvoir nous investir avec l'équipe de développeurs. Étant un projet open source assez récent, il est nécessaire pour eux de corriger toutes les erreurs et bugs que les utilisateurs lambda, tel que nous, rencontrons.

4. Conclusion générale

A travers ce projet de fin d'études, nous avons essayé de montrer qu'il est inutile voir dangereux de se lancer sans avoir réfléchi préalablement à la modélisation du sujet donné. Nos expériences personnelles et les enseignements de l'IUT nous ont permis de ne pas tomber dans ce piège. Nous avons donc consacré une partie du temps pour définir le cahier des charges et pour trouver les solutions envisageables au problème donné. Il faut avancer pas à pas et ne pas se précipiter sur des parties de code qui ne peuvent être fait avant l'heure. C'est donc directement l'application d'un diagramme des tâches, vu à la fois en mathématiques et en gestion, que nous avons appliqué afin de répondre à la troisième question de ce projet tuteuré.

Cependant, malgré toutes les mesures prises pour éviter les mauvaises surprises, nous nous sommes heurtés à des problèmes de communication entre membres de l'équipe, à des problèmes techniques qui ont réussi à être solutionnés après révision de nos idées de départ et application de l'idée : «coder sur papier avant de se lancer ». Enfin nous avons dû gérer des problèmes personnels comme la gestion du stress.

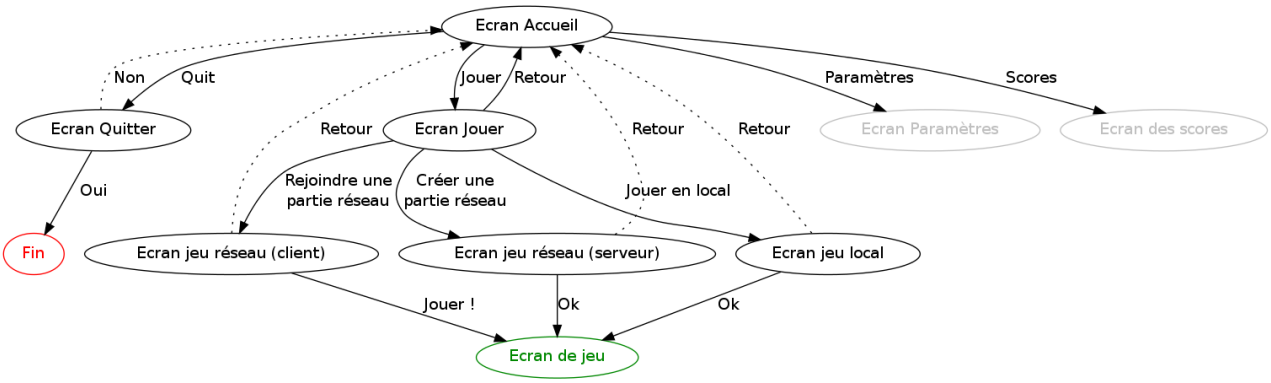
Le travail en équipe s'est révélé être tantôt bénéfique tantôt néfaste. Mais dans l'ensemble, le point de vue de deux étudiants est nécessaire pour la réalisation d'un tel projet. Ces points de vue permettent de trouver diverses solutions et d'être plus proche des attentes des futurs joueurs. Faisant partie de la tranche d'âge visée par ce jeu, il a donc été facile de faire la liste des désirs et des envies des joueurs. C'est ainsi que nous avons pu répondre assez facilement aux deux autres problématiques du projet.

Dans la globalité, le jeu des pousses s'est révélé être un projet très prenant, aussi bien en terme de temps qu'en terme de « plaisir à programmer ». On a pu y découvrir de nouveaux concepts et mettre en application une grande partie des enseignements que l'on a reçu au cours de notre scolarité à l'IUT. D'autre part, nous avons appris à nous servir d'un « jeune » framework et à être capable de suivre son évolution quasi quotidienne.

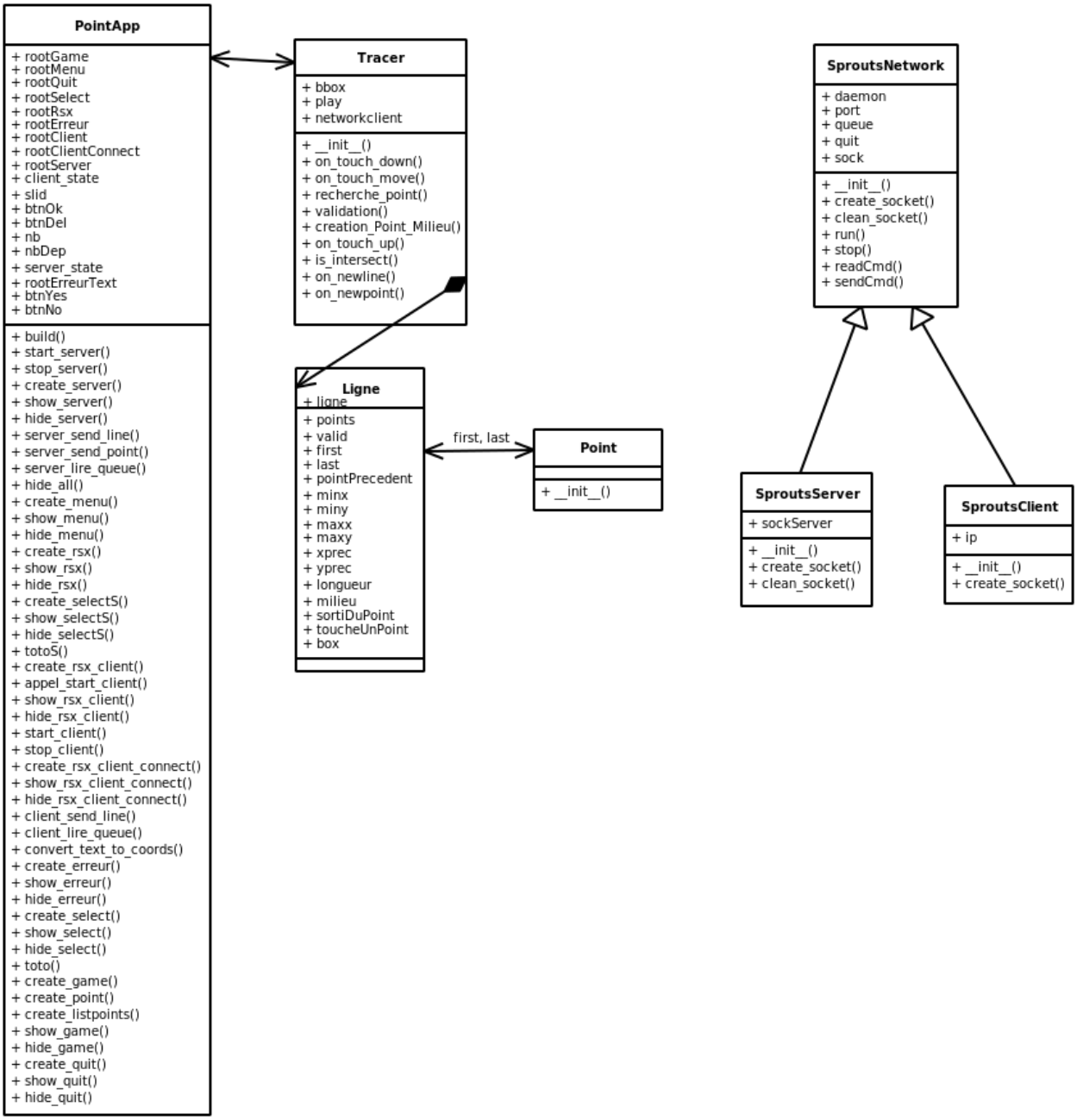
Même si certains objectifs n'ont pas été atteints à ce jour, ce projet ne s'arrêtera pas là. Des objectifs ont été fixés, ils doivent donc être remplis. De plus, une certaine motivation nous anime du fait de la probable intégration du jeu au framework Kivy, ainsi qu'une certaine fierté si nous réussissons à rendre ce jeu disponible sur l'Android Market dans les semaines à venir.

Annexes

Annexe 1 : Enchainement des écrans



Annexe 2 : Diagramme UML



Annexe 3 : Chronographes

Illustration 3: Initialisation du jeu

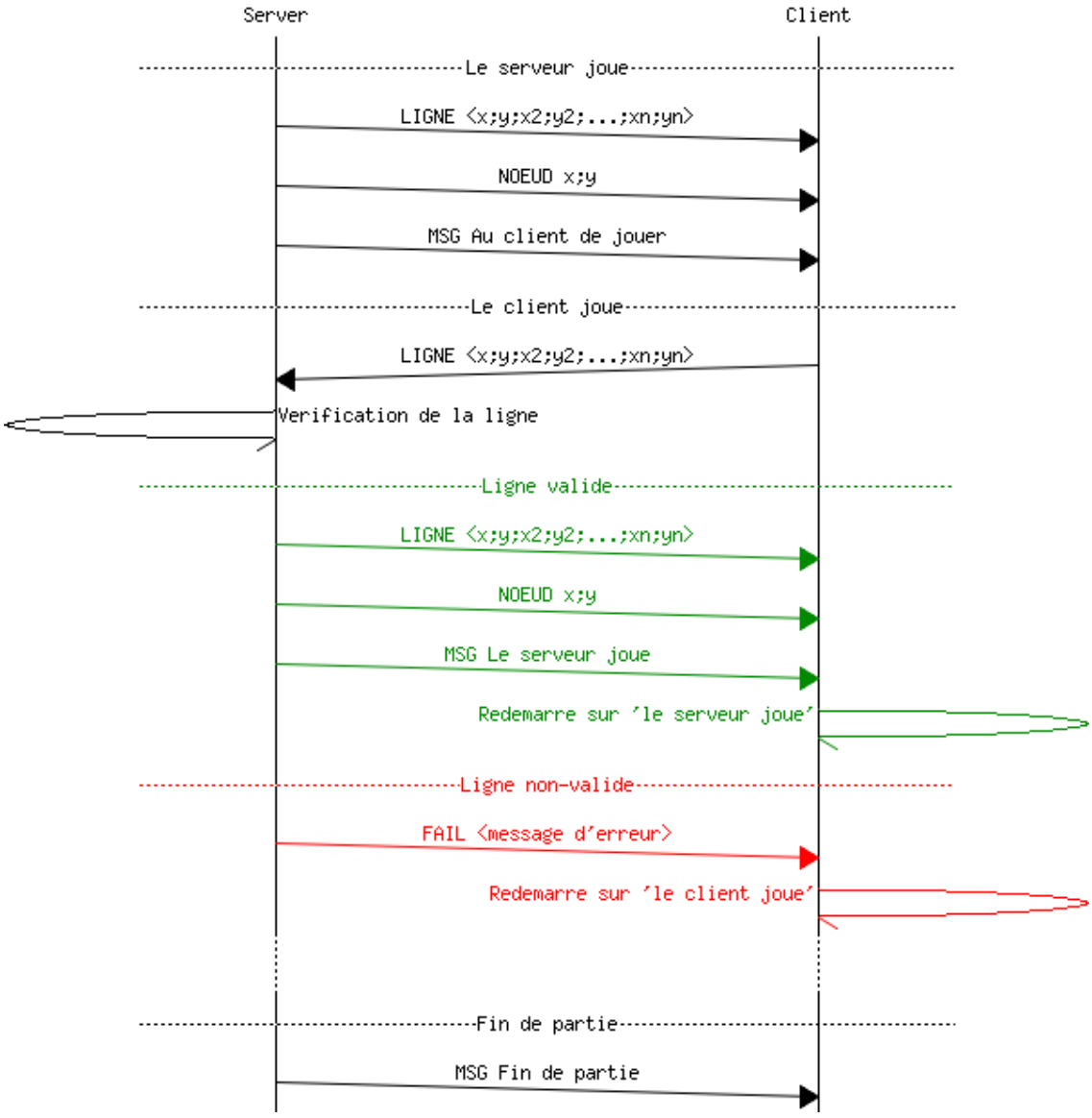


Illustration 4: Boucle de jeu

Illustration 5: Envoi d'un message

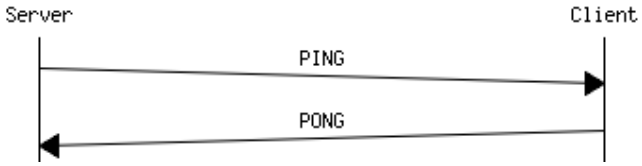


Illustration 6: Vérification du timeout

Glossaire

Android Market : Boutique d'application en ligne pour le système d'exploitation Android (tm)

Framework : Bibliothèque de classes et fonctions, prévus pour être dérivées et étendues, répondant au besoin du développeur.

Multitouch : Interaction avec le matériel informatique (tablette, smartphone) incluant plus d'un point de contact (les doigts)

TCP/UDP : Type de protocole réseau : TCP est connecté et fiable, UDP est non connecté et plus rapide que TCP

Ide : Logiciel permettant d'accélérer le développement d'un logiciel en automatisant les tâches récurrentes, et en aidant le développeur avec divers outils intégrés, des squelettes de méthodes, etc

Github : Site web social qui permet le dépôt de code source ainsi que sa gestion basé sur l'outil GIT (notamment utilisé pour gérer les versions du code source du noyau Linux)

Smartphone : Téléphone mobile intelligent, incluant beaucoup d'outils (agenda, web, email, etc.)

Workflow : Processus d'information / flux de travail

Commit : Commande GIT permettant la sauvegarde du travail en cours dans le dépôt de code source, et en lui attribuant un numéro unique

NUI : Acronyme de « Natural User Interfaces »

Bytecode : Code intermédiaire qui requière une machine virtuelle pour son exécution

GPL : Licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU. Les termes de la GPL n'autorisent toute personne à recevoir une copie d'un travail sous GPL. Chaque personne qui adhère aux termes et aux conditions de la GPL a la permission de modifier le travail, de l'étudier et de redistribuer le travail ou un travail dérivé. Cette personne peut toucher de l'argent pour ce service ou bien ne rien toucher.

Références

- Algorithme du peintre : http://fr.wikipedia.org/wiki/Algorithme_du_peintre
- Git : http://git-scm.com
- Github : http://github.com
- Kivy : http://kivy.org
- Line Segment Intersection Algorithm, Bryce Boe : http://www.bryceboe.com/2006/10/23/line-segment-intersection-algorithm/
- Nuigroup : http://nuigc.com
- PyMT : http://pymt.eu
- Python : http://python.org/
- Ubuntu : http://ubuntu.org
- Wikipédia : http://fr.wikipedia.org