UNIVERSITE MOHAMED V de RABAT

ECOLE NATIONLE SUPERIEURE D'INFORMATIQUE ET D'ANALYSE DES SYSTEMES

Rapport de Projet de Fin de 1^{ere}Année

Filière : Ingénierie Intélligence Artificielle

Snubby Land : World's Hardest Game

Soutenu par :
Abdellatif Ait Hammadi
Ayoub Assis

Encadré par :
Mr. Mohamed Lazaar

Examinatrice :
Pr. Houda Benbrahim

Année Universitaire 2019-2020

Table des matières

Table des figures

Introduction Genérale

Dans notre société, le divertissement est devenu une nécessité quotidienne, ce dernier est satisfait maintenant par les jeux videos.
La plupart des jeux videos aujourd'hui sont des jeux complexes qui exigent avoir une machine assez puissante pour bien marcher. Ainsi la majorité des utilisateurs cherchent des jeux qui sont assez stimulant mais qui marchent sur des machines modestes.
Il est donc généralement nécessaire d'optimiser l'équation pour offrir un resultat faisable, dont le sens des exigences, qui maximise la satisfaction de l'utilisateur.
Ce projet consiste à concevoir et à réaliser une variante d'un jeu video 2D existant sur le web intitulé **Snubby Land : The World's Hardest Game** ; en fait ce n'est pas le plus difficile jeu, mais malgré il est un jeu qui provoque l'esprit du joueur et perturbe son état psychique.
Ce rapport synthetise le travail que nous avons effectué. Il est constitué des parties suivantes :

- Le premier chapitre est consacré à présenter le concept du jeu (Game Concept), l'étude de la version existante, et les objectives de ce projet.
- Le deuxième chapitre "Specification des besoins", présente les differentes besoins et options à intégerer dans cette version, tout en gardant les options de la version etudiée.
- La conception des options est presentée dans le troisième chapitre.
- La partie de réalisation résume le produit final avec des captures d'écrans, et les outils utilisés pour aboutir a ce produit.
- Finalement, la conclusion donne des perspectives et quelques points de developpement potentiels.

Chapitre 1

Présentation du projet

Introduction

Ce chapitre est consacrée pour donner une description générale du jeu, notamment les éléments constituant le jeu, comment jouer et comment gagner une partie du jeu, après citer les propriétés du jeu existant dont on s'est inspirer, enfin donner les grandes objectifs de ce projet. Pour une meilleure description du jeu, nous avons utilisés des figures prise du jeu existant.

1.1 Description du jeu

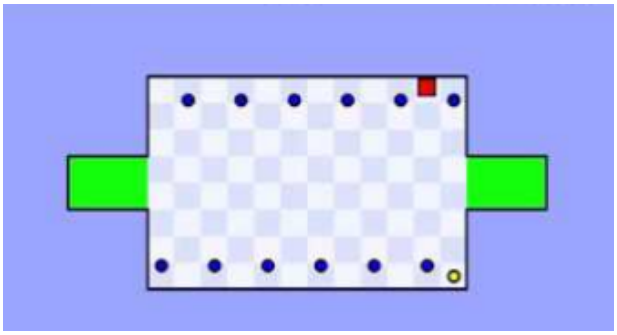

FIGURE 1.1 – Les elements d'un stage de SnubbyLand

Ce jeu est constitué de plusieurs stages. Dans un stage, il y a un objet, qui sera appelé dans toute la suite Snubby, qui doit collecter toute les récompenses, des cercles jaunes. Le mouvement de Snubby est limité par des frontieres et des obstacles mobiles qu'il doit éviter tout contact avec pour ne pas échouer. Au cours du jeu, Snubby peut sauvegarder son progrés de mouvement en se deplacant vers une zone verte dans l'espace du mouvement. Snubby a un nombre d'éssais prédefini pour chaque stage ; quand Snubby touche un obstacle mobile ce nombre décroit et il revient à la dernière zone verte touchée pendant le stage, et si ce nombre est nulle Snubby doit recommencer le stage à zéro. Si Snubby a collecté toute les récompenses, donc il gagne.

1.2 Etude de l'existant

Le jeu existant (SnubbyLand : World's Hardest Game) développé par Snubby Land en 2008, suit presque le même principe décrit au-dessus, il offre un nombre qui est limité des stages (30 stages) en un seule mode de jeu, notamment jeu individuel. Aussi, il ne laisse pas la liberté aux utilisateurs/joueurs d'explorer les stages du jeu, et de les rejouer après les terminer.

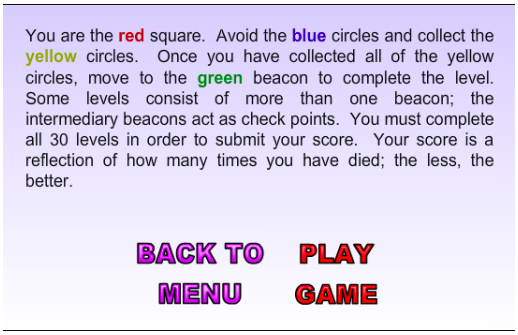

FIGURE 1.2 – Description du principe du jeu.

FIGURE 1.3 – Menu principal du jeu.

1.3 Objectifs du projet

En relation avec le module du projet de fin d'année, ce projet vise a nous permettre de finaliser nos connaissances en ce qui concerne le development et l'application des concepts des structures de données, et aussi améliorer nos compétences de travail en groupe et d'organization.

L'objectif de ce projet est d'améliorer la version existante en offrant plus de liberté au joueur en :

— ajoutant d'autres modes de jeu, surtout jouer avec ces amis.
— laissant la liberté aux joueurs de créer leurs propres stages.
— laissant la liberté aux joueurs de rejouer les stages.

Conclusion

Ce jeu d'arcade est constitué de plusieurs stages dont chacun est constitué d'un carre (Snubby) qui doit se déplacer dans une zone limitée, qui ne doit pas toucher des obstacles mobiles, et qui doit collecter des récompenses et après ce diriger vers une zone securisée pour valider le stage. L'objectif principal est d'offrir plus d'options que celles presentées dans l'existant.
Dans le chapitre suivant, nous définissons en détail les élements et les options qui seront offert par notre version.

Chapitre 2

Spécification des besoins

Introduction

Cet étape est indispensable pour décrire les besoins qui seront satisfaites par ce jeu avec clarté. Ainsi pour offrir une bonne expérience à l'utilisateur (UX), il est essentiel de bien definir sans ambiguité les options qui seront offertes par cette version et sa valeur ajoutée, mais aussi les limites de cette version.

2.1 Les besoins fonctionnels

Ce jeu doit fournir une expérience vivide au joueur, tout ce qui est déjà integré dans l'existant ainsi que des nouvelles fonctionnalités :

- **Les fonctions basiques :**
  - Le joueur doit être capable de jouer à n'importe quel stage indéfiniment.
  - Le joueur doit être capable de déplacer snubby dans les huits directions(haut,bas,droite,gauche,bas droite,bas gauche, haut droite, haut gauche) dans la zone permise.
  - Le joueur doit être capable de collecter les recompenses dès qu'il les touche.
  - Le jeu doit offrir un certain nombre de stages crés par le developpeur.
  - Le joueur peut créer son stage, l'enregistrer, le jouer et le partager en ligne.
- **Les modes de jeu :**
  - Le jeu doit offrir le choix de jouer individuellement un stage.
  - Le jeu doit offrir le choix de jouer en ligne avec un autre joueur, évidemment le même stage pour les deux.
  - Le jeu doit offrir le choix de jouer avec un autre joueur sur la même machine.
  - Le jeu doit offrir le choix de jouer contre la machine.

2.2 Les besoins non fonctionnels

- **Ergonomie et souplesse :**
  L'interface graphique doit être conviviale, significative.
- **Rapidité :**
  Le temps de chargement d'un stage donne doit être le plus rapide possible.
- **Efficacité :**
  L'application doit repondre convenablement vis-à-vis toute commande de l'utilisateur et ne doit pas se planter.

- **Portabilité :**
  Le jeu doit etre utilisable sur toute machine et n'importe quelle systeme d'exploitation.
- **Maintenabilité et scalabilité :**
  L'architecture du jeu doit permettre l'optimisation des fonctionnalites et l'ajout d'autres.

2.3 Les limites

Elles concernent des erreurs qui ne peuvent pas être résolus ou des exigences :

- les fichiers contenant les stages seront enregistrés localement sur la machine de l'utilisateur, ainsi si l'un est corrompus ou n'exister pas le jeu ne va pas fonctionner correctement.
- Evidemment pour accéder au mode de jeu en ligne, la machine doit être connectés à un réseau internet.
- L'interface utilisateur n'est adaptative : La fenêtre du jeu est de taille fixe, c'est à dire que les dimensions des composantes de l'interface graphique du jeu sont les même pour toute les machines, ainsi dans certains machines la fenêtre peut couvrir toute l'écran, dans certaines non. Ca dépend de la résolution de l'écran.
- Le mode en ligne ne peut pas être jouer qu'avec les stages prédefinis.

2.4 Diagrammes des cas d'utilisation

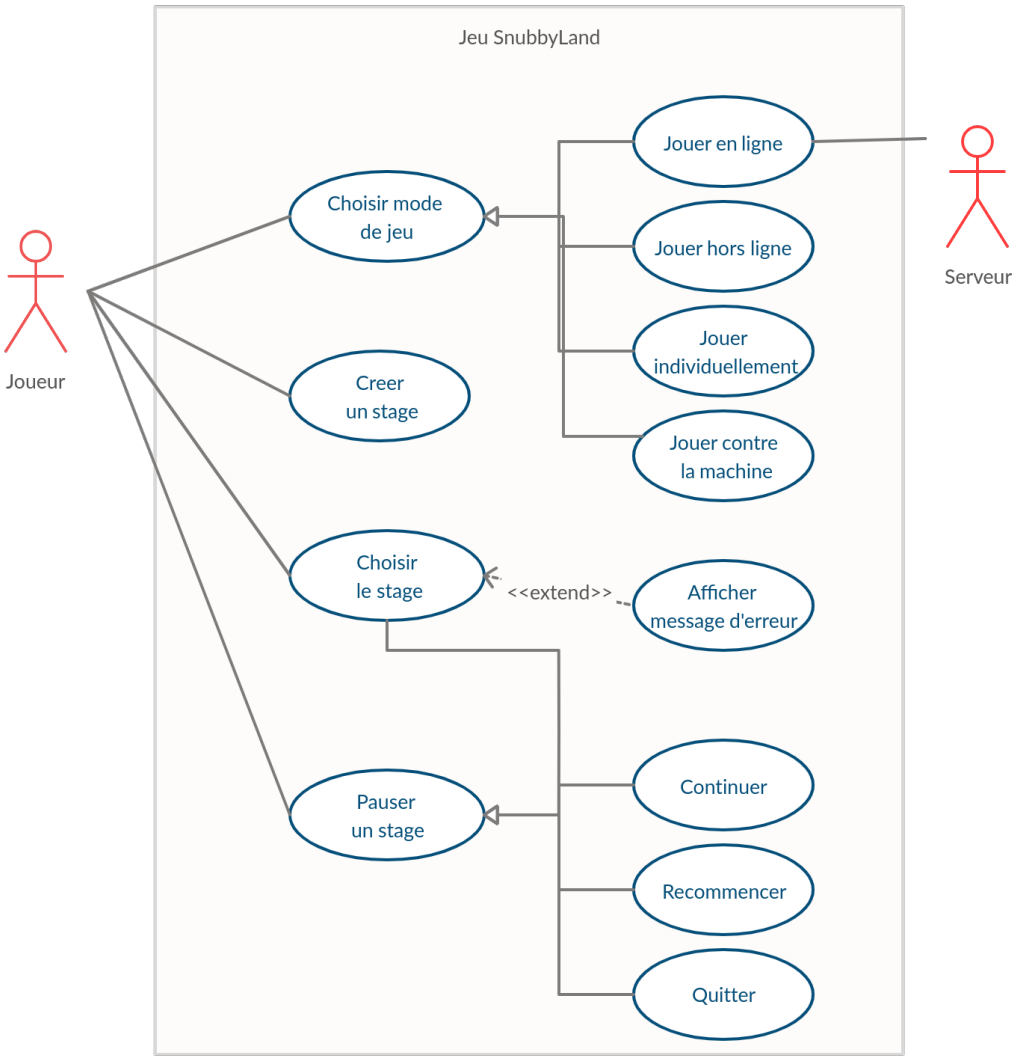

FIGURE 2.1 – Diagramme des cas d'utilisation

Conclusion

Ce chapitre nous a permis de couvrir les différents besoins fonctionnels et non fonctionnels auxquels doit satisfaire notre jeu, et les points de faiblesse. Nous avons aussi détaillé ces besoins à travers un diagramme de cas d'utilisation. Dans la partie suivante, nous allons détailler les éléments et les

procédes utilisés pour satisafaire les besoins decrits ci-dessous.

Chapitre 3

Conception

Introduction

Dans ce chapitre, nous allons entamer une partie importante du développement du jeu qui lie la spécification avec la réalisation. Nous allons présenter dans un premier temps la conception générale de notre jeu, puis la conception du mode en ligne et enfin celle du mode automatique. Nous allons utiliser des diagrammes UML4.3 pour bien décrire ces modes.

3.1 Conception générale

3.1.1 Conception générale

Notre jeu consiste en un module web, utilise pour le mode en ligne uniquement, et un autre d'ordinateur qui sont en interaction à travers des services web.
Le module d'ordinateur est composee de trois parties :

- **La logique du jeu :**
  C'est l'ensemble des fonctions utilisées pour gérer l'état du jeu, et des fonctions pour assurer l'interation entre les differents éléments constituant un stage ainsi que le respect des règles pendant le jeu.
- **L'interface graphique :**
  Pour visualiser les changements d'état du jeu pour le joueur.
- **Partie d'événements :**
  Celle-ci s'occupe de aquérir les commandes entrées par l'utilisateur puis les transmettre vers la partie de la logique.
- **Base de donnée :**
  Il s'agit de l'ensemble des fichiers qui contiennent les stages du jeu, ils

seront stockés au niveau de la machine du joueur.

3.1.2 Diagrammes d'activité

Le diagramme de la figure 3.1 illustre le déroulement des actions entre le lancement du jeu et le choix d'un stage.

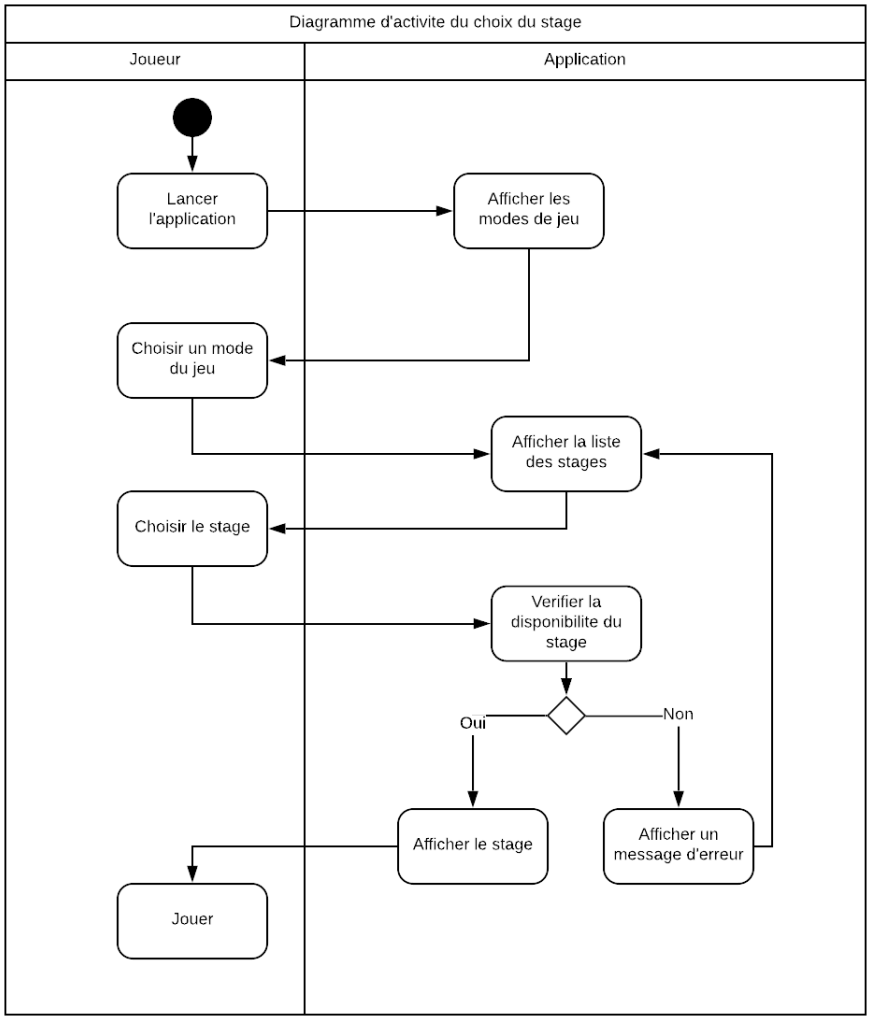

FIGURE 3.1 – Diagramme d'activité du choix du stage

Après avoir choisir un mode de jeu parmi les quatres proposés, le joueur est invité à choisir un stage a partie d'une liste affichée. Dans le cas ideal,c'est-à-dire le fichier du stage existe et non corrompu le stage sera chargé, et le jeu

commencera selon la logique qui sera presentée dans le diagramme suivant. En cas de problème, un message d'erreur expliquant le probleme sera affichee au joueur pour prendre une nouvelle décision adéquate.

Le diagramme de la figure 3.2 détaille la logique utilisée, pendant le jeu d'un stage, pour assurer le bon deroulement de la session du jeu et le respect du règle du jeu.

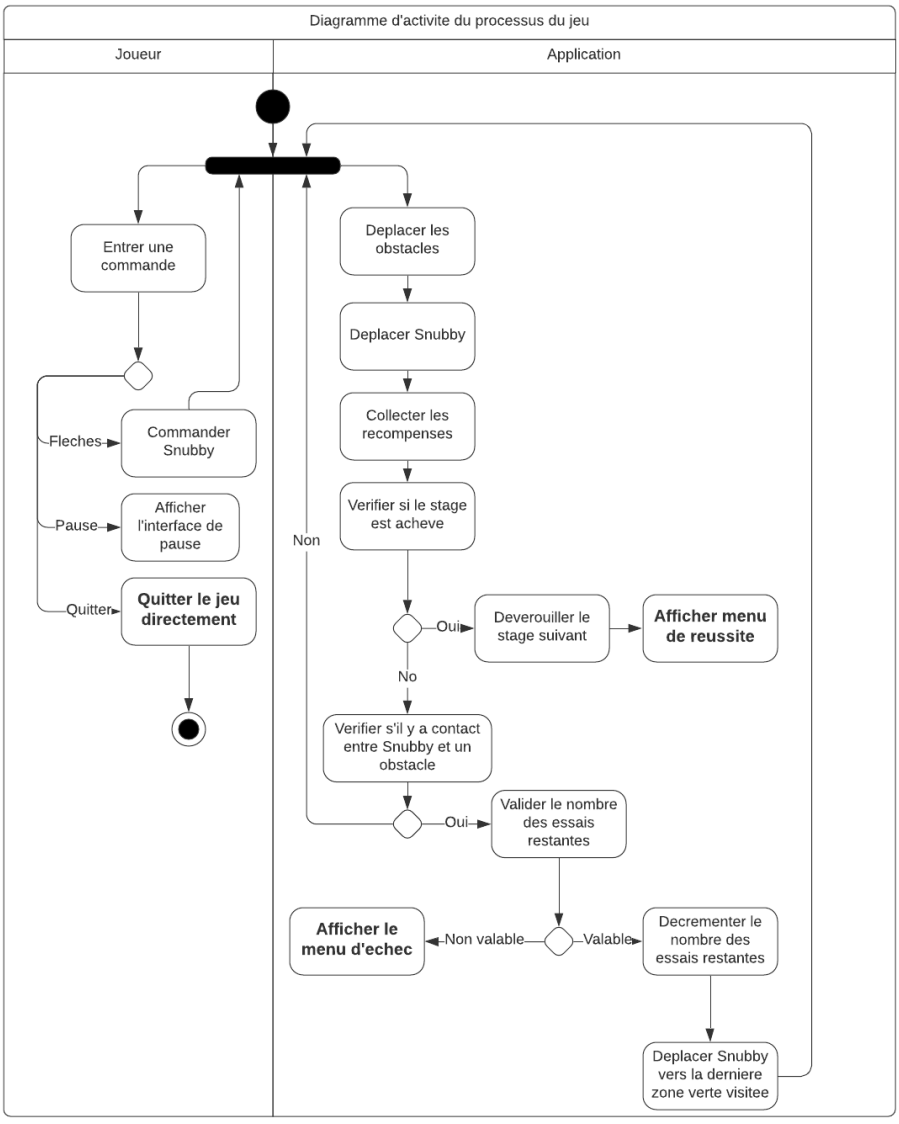

FIGURE 3.2 – Diagramme d'activité du jeu en un stage donné

Le jeu est divisée en deux parties qui s'exécutent simultanement, une permettante de capturer les commandes du joueur et agit adéquatement ; soit quitter la fenêtre du jeu, faire une pause du jeu, ou commander snubby pour se déplacer.
La deuxieme partie est executée indefinement suivant un ordre d'instructions bien determinée pour éviter les problèmes de concurrence entre les threads ;

- les obstacles se déplacent vers leurs nouvelles positions,
- Snubby se déplace selon la commande recu de la part de l'utilisateur,
- collecter les recompenses s'il y a contact avec ces derniers, puis vérifier les conditions d'achèvement d'un stage et agir en conséquence,
- vérifier s'il y a contact entre snubby et un obstacle et agir ainsi.

3.1.3 Diagramme des classes

La figure 6 représente le diagramme de classes des entités utilisées dans notre jeu :

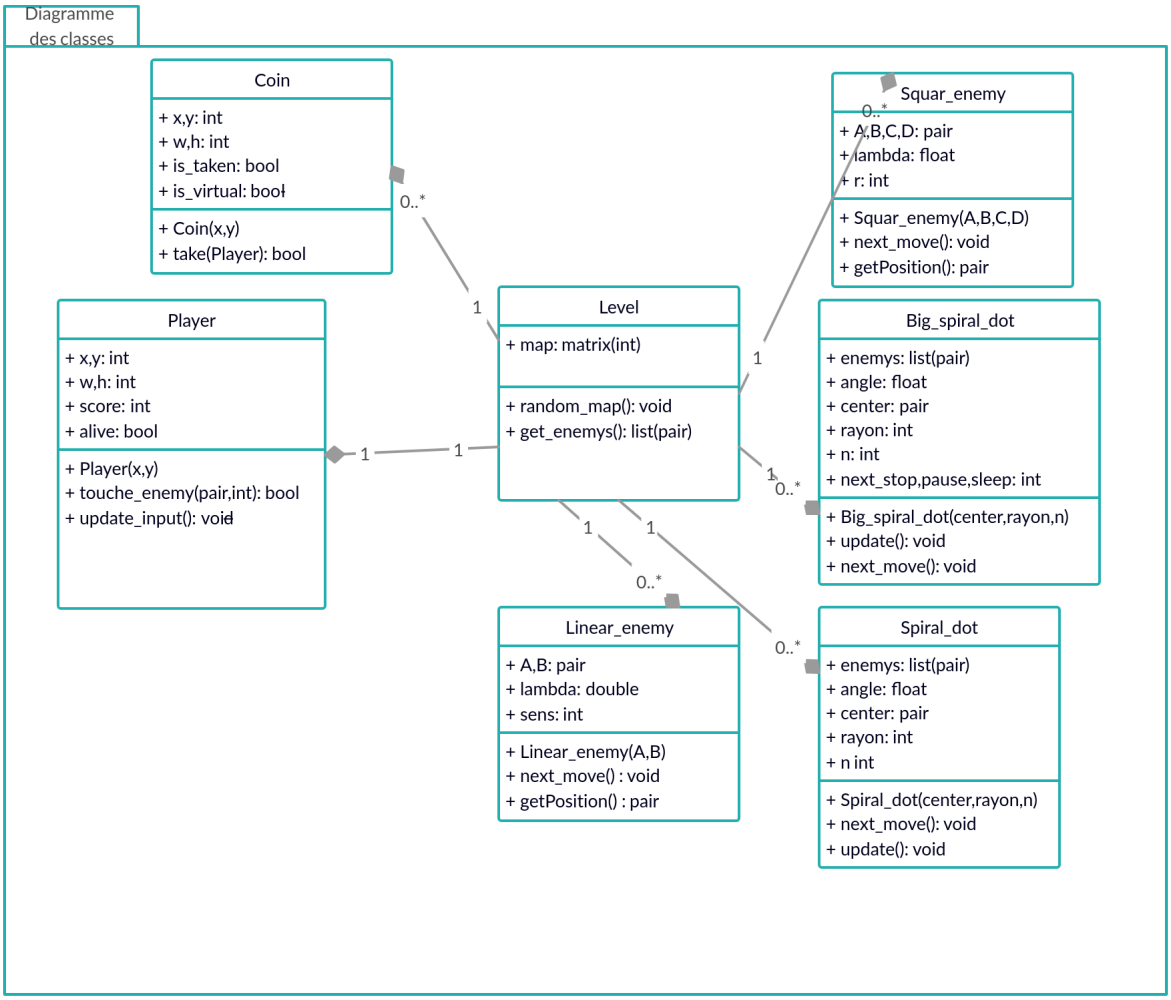

FIGURE 3.3 – Diagramme des classes

Chaque élément du jeu est modélisé par une classe avec des attributs et des opérations, et le tout est regroupé/emballé (Wrapper) dans la classe **Level**

- **Zone :**
  L'espace de mouvement contient trois zones ; permises, bloques, de reprise. Il est representé par une matrice de deux dimensions d'entiers ; chaque zone est codee par un entier unique significatif. Chaque élément de la matrice correspond à un carré de taille fixe dans l'espace de mouvement.
- **Player :**
  Cette classe représente le joueur, il est un élément de la classe Level.

— **Coin :**
Cette classe représente une recompense, il est un élément de la classe Level.
— **Linear enemy :**
Cette entite représente un obstacle qui se déplace entre deux points suivant une ligne d'un manière linéaire de va et vient.
— **Square enemy :**
Cette entité représente un obstacle qui se déplace suivant le perimétre d'un carré ABCD.
— **Spiral dot :**
Cette entité représente des obstacles en rotation uniforme suivant un cercle autour d'un même centre mais différentes rayons, i.e sous la forme d'un plus (+).
— **Big Spiral dot :**
Cette entité est une superposition de deux Spiral dot mais qui tourne en rotation non uniforme.

3.2 Conception du mode en ligne

Le module web installé sur un serveur est l'intermediaire permettant de coordonner entre deux joueurs choisissant le mode en ligne et le même stage. Ceci est modéliser par une file FIFO tel que uniquement deux joueurs peuvent jouer en ligne à la fois. Les étapes pour accéder à ce mode sont dans l'ordre suivant :

1. **Se connecter** :
   Le joueur se connecte avec le serveur après avoir choisir un stage, ce dernier lui associe un identifiant unique tant qu'il est connecté.
2. **Trouver un joueur** :
   Le premier joueur attend qu'un autre joueur qu'il se connecte. Une fois c'est fait, le serveur connecte les deux joueurs (S'ils ont le même niveau demandé) en échangeqnt leurs identifiants entre eux, après le stage est lancé pour les deux.
3. **Au cours du jeu** :
   Chaque joueur lorsqu'il effectue un mouvement, il enregistre sa nouvelle position dans la base de donnée du serveur et il recupére celle de son adversaire, et vice-versa.
4. **Se deconnecter** :
   Si l'un des deux joueurs se déconnecte, sa position dans la base de

donnée est changée par un code de déconnexion. Et dans ce cas, l'autre joueur est consideré gagnant.

3.3 Conception du mode automatique

Pour automatiser le processus du jeu, nous avons eu recours au principe de simulation du phénomene physique : la force de l'intéraction électrique entre deux particules chargées électriquement connue sous le nom de la force de Coulomb 4.3.

Conception

Nous avons associer à Snubby une charge, aux obstacles mobiles une charge chacune et de même signe que celle de Snubby(la même valeur) pour avoir un phénomene de repulsion, et à chaque récompense une charge de signe opposée pour avoir une attraction avec Snubby.
Nous avons uniquement considérer les obstacles dont Snubby appartient à leur zone de danger décrite dans la figure suivante :

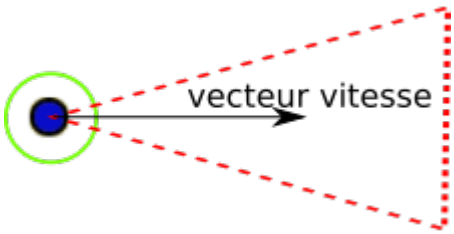

FIGURE 3.4 – Zone de danger d'un obstacle

Cette zone est constituée de deux parties :
— Un cercle dont le centre est celui de l'obstacle est de rayon petit.
— Un partie d'un autre cercle de même centre mais de rayon plus grand, elle est caracterisée par un angle autour du vecteur de vitesse de l'obstacle. Nous considérons le produit scalaire entre la vitesse de l'obstacle et le vecteur entre l'obstacle et Snubby pour détecter si ce dernier y appartient.

Le vecteur résultant de la somme des forces va diriger le sens du mouvement de Snubby.
Pour remédier le problème du phénomene de blockage décrit dans la figure 3.5 ci-dessous, qui résulte quand le vecteur résultant guide Snubby vers la frontière, nous avons eu recours à un algorithme de recherche de chemin qui trouvera un chemin court entre la position actuelle de Snubby et la position

de la plus proche récompense en passant dans la zone permise, cet algorithme sera utilisée d'une manière périodique ; c'est-à-dire apres un certaine periode de temps (par exemple tout les 10 seconds), et Snubby sera guide en ajoutant des recompenses *virtuelles*[1].

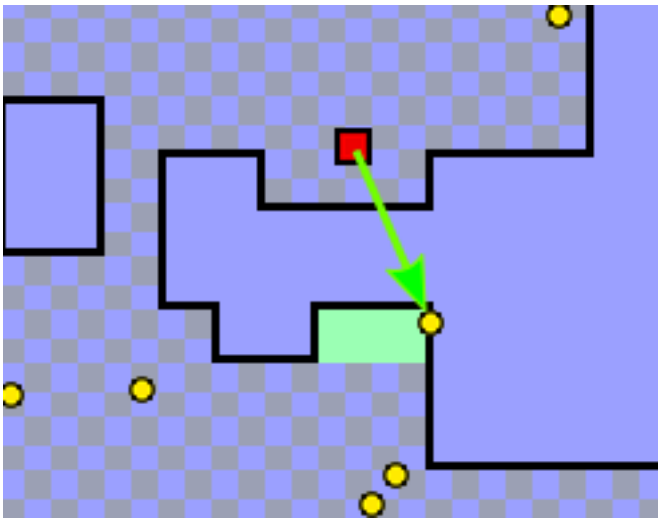

FIGURE 3.5 – Snubby bloque par la frontière pour arriver à la récompense

Algorithme

1. Trouver les obstacles dont Snubby appartient à leur zone de danger.
2. Calculer le vecteur résulant,
3. Déplacer Snubby par un pas suivant ce vecteur,
4. Tout les Δt, ajouter des récompenses *virtuelles* à l'aide de Djikstra.

Conclusion

Dans ce chapitre, nous avons modélisé le fonctionnement du jeu afin d'avoir une vue globale du système. Nous avons aussi détaillé les différents modules et algorithmes du jeu ce qui nous a permis d'organiser le travail et d'avoir une idée claire sur le travail à réaliser. Ce travail est décrit plus précisément dans le chapitre qui suit.

1. Ne seront pas compter pendant le jeu et ne seront pas visualiser pour l'utilisateur.

Chapitre 4

Réalisation

Introduction

Après l'étape de conception de l'application, nous allons, dans ce chapitre, décrire la phase de réalisation. Nous allons présenter, en premier lieu, l'environnement du travail utilisé pour le développement du jeu, ensuite, nous allons donner un aperçu sur le travail accompli à travers des captures d'écran.

4.1 Environnement de travail

Dans ce paragraphe nous décrivons les différents outils et logiciels utilisés pour la mise en œuvre du jeu.

4.1.1 Environnement matériel

Les machines utilisées pour réaliser ce projet :

Proprietes	Lenovo	Samsung
Systeme d'exploitation	Ubuntu 18.04 LTS 64	Ubuntu 19.04 x86 64
Processeur	Intel Core i7-6500U 2.50GHz x 4	Intel celeron 847 (2) 1.100GHZ
RAM	7.7GB	3642MB
Disque dur	214.8 GB	500GB
Resolution de l'ecran	1366x768	1280x728

4.1.2 Environnement logiciel

Afin de réaliser notre application,nous avons eu recours au programme Inkscape pour le désign des images utilisées dans l'interface graphique , et a

Sublibme Text comme editeur de texte pour coder.

4.2 Les technologies utilisées

4.2.1 Les langages de programmation

— **C++** :

FIGURE 4.1 – Langage de programmation

On a utilisé la version 14 de c++, le choix de ce langage était principalement pour bénéficier des structures du données élémentaires prédéfinies dans la bibliotheque STL, aussi pour pour bénéficier du paradigme objet de ce langage.

— **Python 3** :

FIGURE 4.2 – Langage de programmation

Ce langage est utilisé dans la partie Serveur.

4.2.2 Bibliothèques utilisees

Pour faciliter notre travail, nous avons considérer l'utilisation les bibliothèques suivantes :

— **SDL2** :

FIGURE 4.3 – Bibliothèque de multimedia

biblithèque qui assure tout ce qui en relation avec le multimédia, notamment l'affichage de l'interface du jeu sur l'écran, capter les événements/commandes du l'utilisateur.

— **Boost Serialization** :

FIGURE 4.4 – Bibliothèque de gestion des opérations Entrées-Sorties (IO).

Celle-ci s'occupe du stockage, et le chargement des stages à partir du disque dur.

4.3 Interface Homme/Machine

Menu d'accueil

C'est le premier menu rencontre par le joueur, il donne plusieurs choix de mode entre outre : mode individuel,mode en ligne, créer son level...

FIGURE 4.5 – Menu d'accueil

Le choix de n'import quelle mode de jeu le mène vers une interface pour choisir un stage, mais pour le choix de créer un stage, il sera dirigé vers une autre interface contenant un tutoriel comment faire.

Choix d'un stage

Après avoir choisir un mode de jeu, le joueur choisira parmi les stages un, ou partager un stage avec quelqu'un par l'intermédiaire du serveur ou bien recevoir un stage.

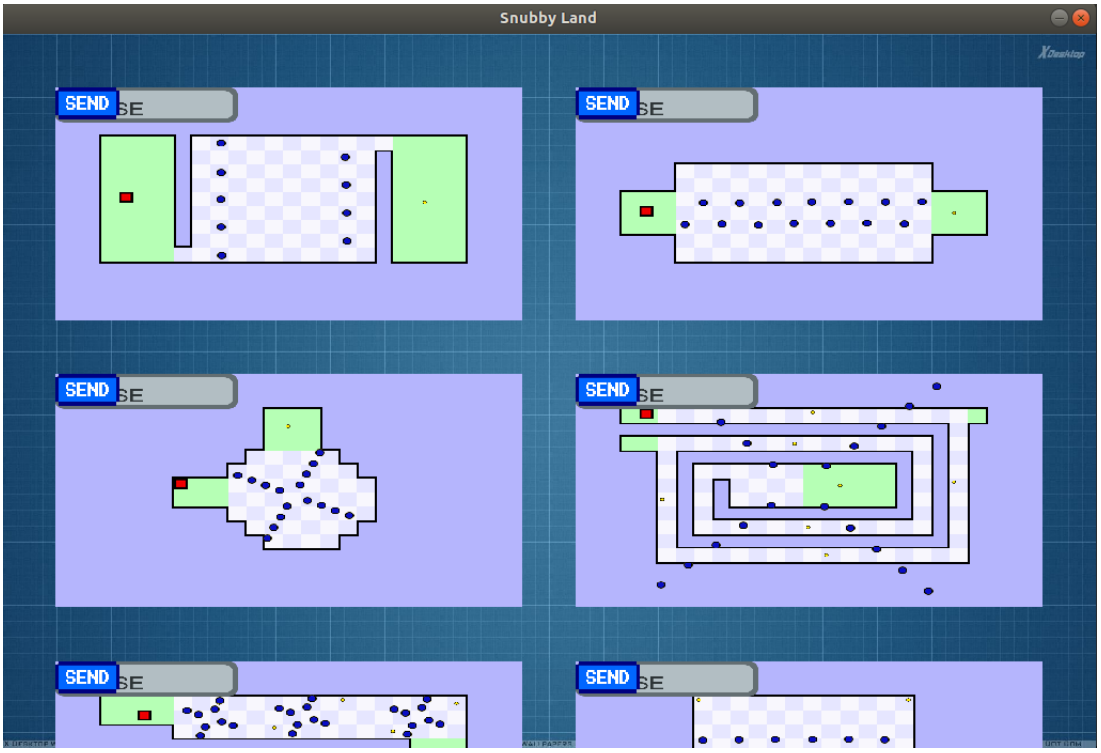

FIGURE 4.6 – Choix d'un stage

Creation de son propre stage

Si le joueur a choisit de construire son niveau, il sera guidé à travers un tutoriel ; la partie gauche de la figure designe le progrés de l'utilisateur dans sa tâche, la partie en-bas présente les buttons à utiliser pour personnaliser la tâche actuelle.

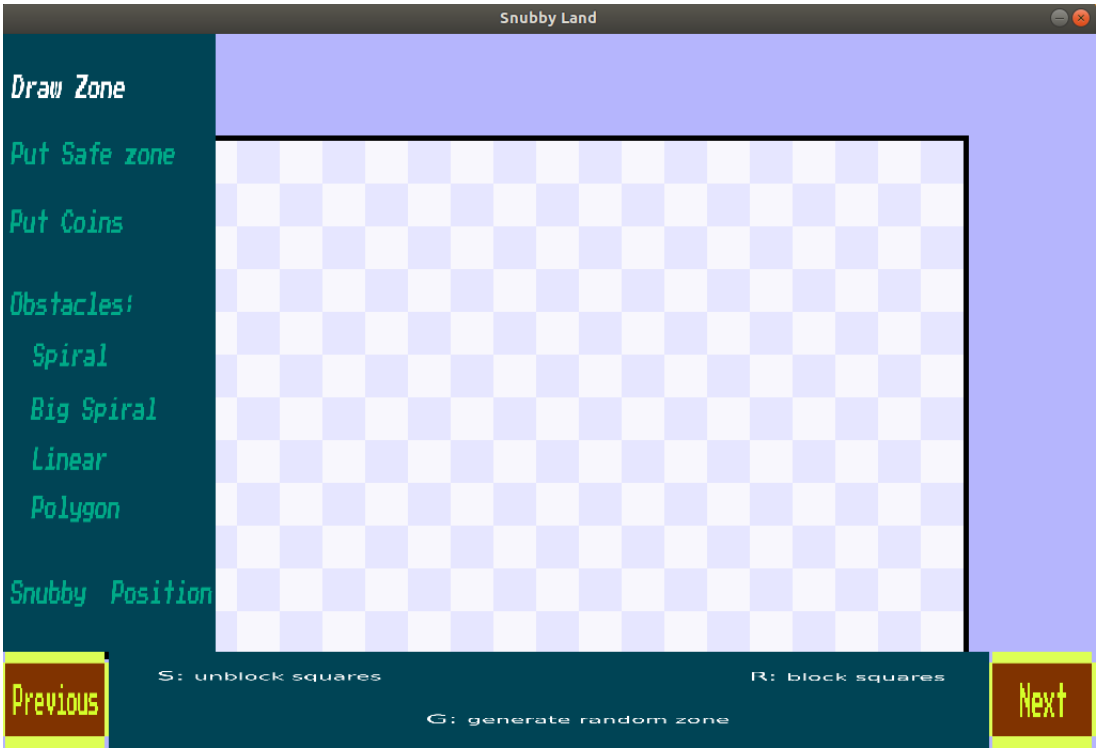

FIGURE 4.7 – Restreindre la zone du mouvement de Snubby

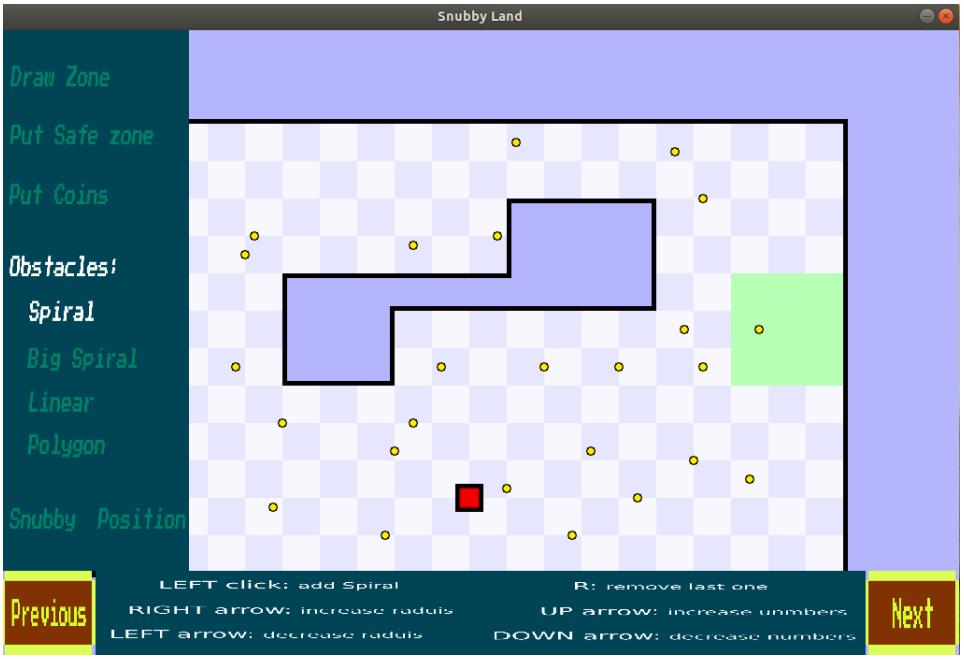

FIGURE 4.8 – Personnalization des obstacles spirales

A la fin de tout les étapes, le stage sera enregistré et lancer directement pour être jouer.

Interface lors du jeu

Après choisir un stage valable, ce dernier sera chargé du disque dur et lancé. Il offre la possibilité de pauser le stage.

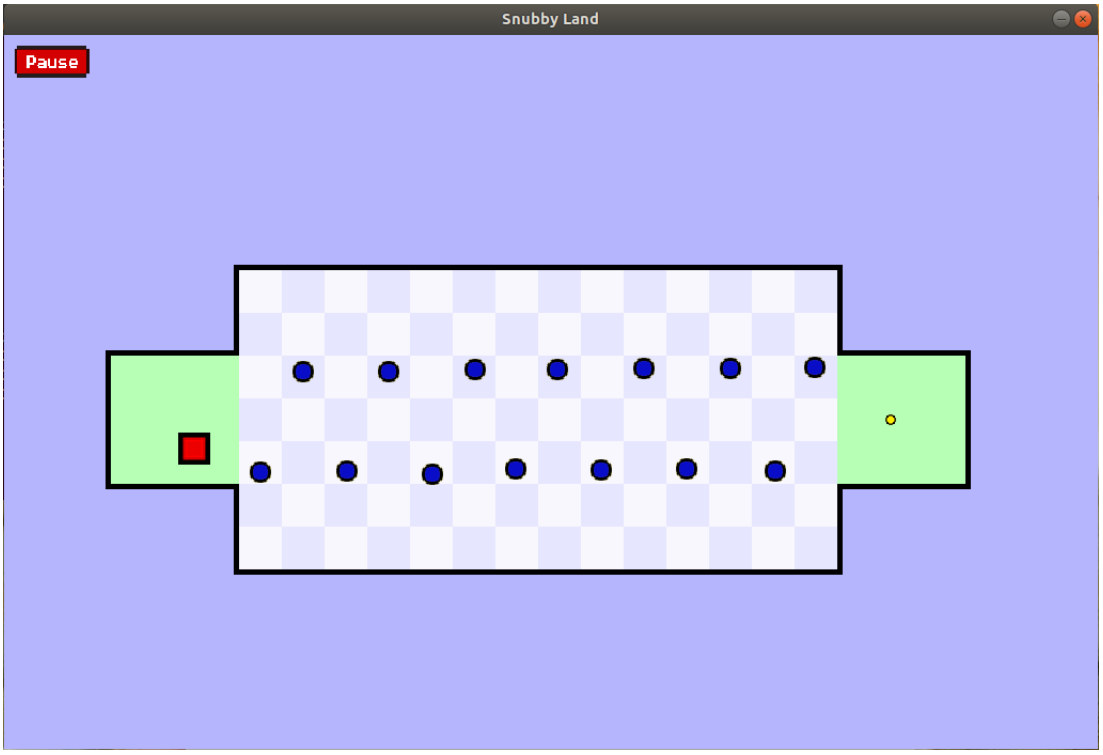

FIGURE 4.9 – Au cours du jeu

Menu de réussite d'un niveau

Si le stage est complété avec succes, ce menu ci-dessous sera affiche directement.

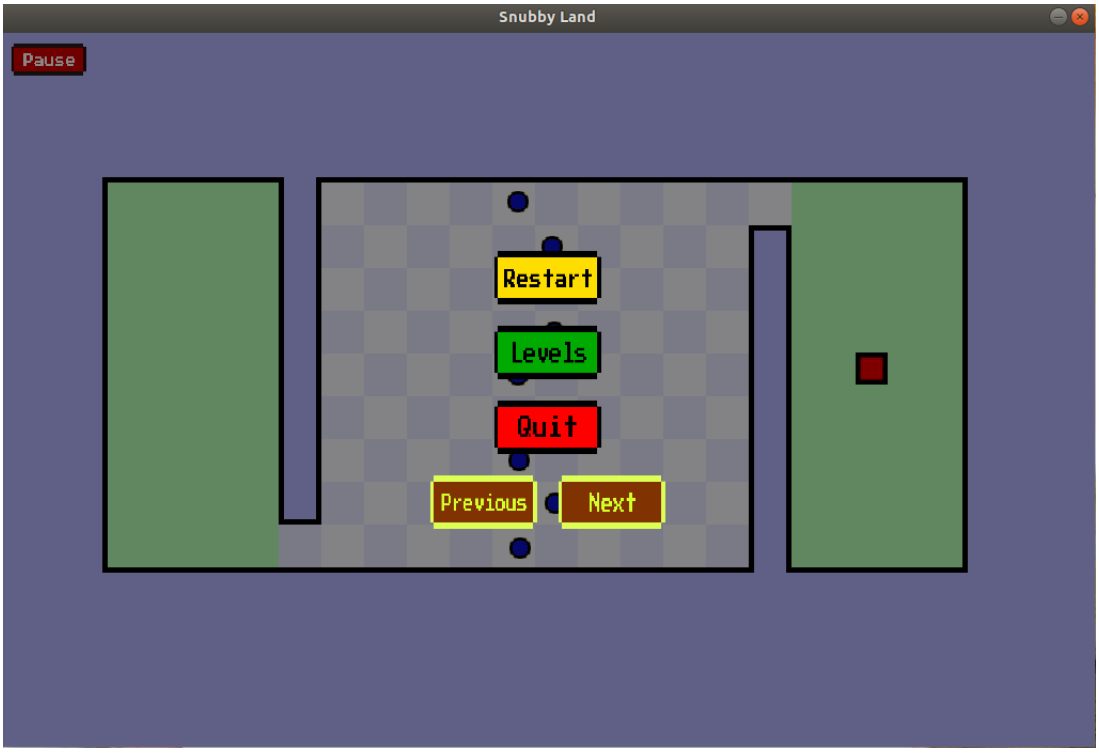

FIGURE 4.10 – Menu d'achévement de stage.

Conclusion

Dans ce chapitre, nous avons presenté quelques interfaces graphiques du jeu, ainsi que les bibliothèques utilisées pour faciliter le travail.

Conclusion générale

L'élaboration de notre travail était dans le but de concevoir un jeu video dédié aux ordinateurs. Ce jeu offre plus d'options et de liberte au joueur pour choisir parmi différents modes de jeu, parmi plusieurs stages, meme de créer ses propres niveaux.

Depuis le présent document, nous détaillons les différentes étapes suivies pour une réalisation réussie du travail demandé.

Dans un premier lieu, nous avons commencé par décrire les éléments du jeu, son principe, ses règles et la valeur ajoutée de cette version-là.

La deuxième étape dans la réalisation de notre projet était de faire une étude detaillée à propos des options que nous pouvons offrir et les besoins, fonctionnels et non fonctionnels, à satisfaire pour atteindre les grandes objectives.

Partant de la spécification, nous avons commencé la conception de l'application à travers les différents diagrammes UML à savoir le diagramme de classes et les diagrammes d'activites, puis la description de l'algorithme d'automatisation du jeu, et le module en ligne.

Par la fin, nous avons décrit, depuis le chapitre réalisation, les bibliothèques et logiciels utilisés, puis présenté le résultat final en quelques captures d'écran de certaines interfaces.

Ce projet nous a également donné l'occasion de découvrir des nouveaux concepts dans le monde de dévelopment d'application tels que : séparation des responsabilités (Separations of concerns), l'expérience de l'utilisateur (User Experience UX), en plus de consolider nos expériences de travail en groupe.

Cette version du jeu offre justement un peu de chaque option, donc elle peut être aisément améliorée. En fait, il y a possiblité d'améliorer plus le mode automatique, généraliser le mode en ligne à plusieurs joueurs jouant simultanement, et aussi dans le cadre de la génération procédurale des cartes pour créer une infinité de stages générés aléatoirement.

Bibliographie

— https ://wiki.libsdl.org/APIByCategory , la documentation officielle de SDL.
— https ://www.boost.org/doc/libs/1_72_0/libs/serialization/doc/ , la documentation officielle de Boost Serialization.
— https ://www.learncpp.com/ .
— https ://stackoverflow.com/ .
— https ://www.khanacademy.org/science/ap-physics-1/ap-electric-charge-electric-force-and-voltage ,Electric charge and electric force.

Annexes

Le formalisme UML

Presentation du UML

UML est un langage graphique de modélisation des données et des traitements. C'est une formalisation très réussie de la modélisation objet utilisée en génie logiciel. Il est l'accomplissement de la fusion des précédents langages de modélisation objet Booch, OMT Et OOSE.

Description du UML

Le formalisme UML est composé de 13 types de diagrammes (9 en UML 1.3).
UML n'étant pas une méthode, leur utilisation est laissée à l'appréciation de chacun, même si le diagramme des cas d'utilisation est généralement considéré comme l'élément central d'UML. De même, on peut se contenter de modéliser seulement partiellement un système, par exemple certaines parties critiques.

UML se décompose en plusieurs sous-ensembles :

— **Les vues** :
Les vues sont les observables du système. Elles décrivent le système d'un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique, etc. En combinant toutes ces vues il est possible de définir (ou retrouver) le système complet.

— **Les diagrammes** :
Les diagrammes sont des éléments graphiques. Ceux-ci décrivent le contenu des vues, qui sont des notions abstraites. Les diagrammes peuvent faire partie de plusieurs vues.

— **Les modèles d'élément** :
Les modèles d'élément sont les briques des diagrammes UML, ces modèles sont utilisés dans plusieurs types de diagramme. Exemples

d'élément : cas d'utilisation, classe, association, ...

Les principaux diagrammes d'UML sont :

— **Les diagrammes de cas d'utilisation** :
utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Les deux composants principaux des diagrammes de cas d'utilisation sont les acteurs et les cas d'utilisation.

— **Les diagrammes de séquences** :
permettent de décrire les interactions entre différentes entités et/ou acteurs : par exemple des objets dans un modèle d'un logiciel, des sous-systèmes dans un modèle d'un système complet. Le temps est représenté comme s'écoulant du haut vers le bas le long des "lignes de vie" (lifeline) des entités, alors que les flèches représentent les messages qui transitent d'une entité vers l'autre.

— **Le diagramme de classes** :
c'est un schéma utilisé en génie logiciel pour représenter les classes et les interfaces d'un système ainsi que les différentes relations entre celles-ci.

Au niveau des annexes, on peut inclure des documents qui peuvent fournir plus d'informations pour une meilleure compréhension de ce qui a été évoqué dans le rapport. Ces documents ne sont pas indispensables mais apportent néanmoins plus d'éclaircissement. Chaque document référencé dans les annexes doit être cité au moins une fois dans le rapport.

Force de Coulomb

C'est phénomene physique qui résulte de l'intéraction entre deux particules ponctuelles chargées électriquement.

Principe de Coulomb

La force d'intéraction mutuelle appliquées par la charge électrique q_1 sur q_2 est :

$$F_{12} = -\frac{1}{4\pi\epsilon_0}\frac{q_1 q_2}{r^3}\vec{r}$$

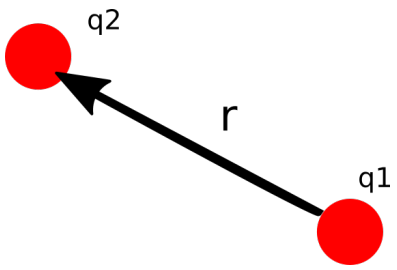

FIGURE 4.11 – Force de Coulomb