

Le Mans Université

Licence Informatique *2ème année*

Rapport de projet

NINETEEN

<https://github.com/Bniais/NineTeen>

David Legrand, Hugo Lecomte, Samy Mahi, Rayyan Lajnef

14 mai 2020



Table des matières

1	Introduction	3
2	Organisation du travail	4
3	Fonctionnalités et concept du projet	5
3.1	Installation	5
3.2	Environnement	5
3.3	Description des jeux	6
3.3.1	Flappy Bird	6
3.3.2	Démineur	7
3.3.3	Snake	8
3.3.4	Tetris	9
3.3.5	Astéroïde	10
3.3.6	Space Shooter	12
4	Développement	12
4.1	Création de la salle d'arcade	12
4.2	Importation du fichier 3D	13
4.3	Déplacements et interactions dans la salle d'arcade	13
4.4	Les lumières	14
4.5	Spatialisation des sons ambiants	14
4.6	Transition tri-dimension / bi-dimension	15
4.7	Utilisation de SDL dans les jeux	15
4.7.1	Écriture de texte	15
4.7.2	Affichage d'images	15
4.7.3	Jouer des sons	16
4.8	Réseau et sécurité	16
4.8.1	Protection des variables sensibles	16
4.8.2	API dédiée	16
4.8.3	Construction d'une requête réseau	17
4.8.4	Communication avec l'API Dédiée	17
4.8.5	Authentification de la requête	17
4.8.6	Conception de la base de données	18
4.8.7	Requête SQL	18
4.9	Conception du site web	19
4.10	Méthode de compilation	19
4.11	Phase de débogage	19
4.12	Optimisation du travail de groupe	20
5	Conclusion	20

1 Introduction

Dans le cadre du module de projet, il nous a été demandé de créer un jeu à l'aide de nos connaissances en informatique ainsi que des cours dont nous disposons. En ce qui concerne le jeu, nous avons à notre disposition plusieurs exemples type. Nous avons cependant pris certaines libertés dans l'élaboration de ce projet. En effet, nous avons créé, non pas un seul jeu global, mais plusieurs petits jeux reliés par une salle d'arcade. L'ensemble des jeux, autrement dit, le programme, est disponible sous forme de logiciel. Ce dernier donne accès à une page de connexion pour les personnes déjà inscrites, et une possibilité d'inscription pour les personnes non inscrites. La page d'inscription renvoie à une page internet comprenant un formulaire d'inscription. Une fois inscrit et connecté, le joueur peut pénétrer dans la salle d'arcade et s'y déplacer librement à la première personne. Cette salle d'arcade, nommée Nineteen, est modélisée en trois dimensions (3D) et développée à partir du moteur graphique OpenGL. La partie son de cette salle est développée avec l'aide de la librairie *SDL_mixer*. Un fois dans cette salle, l'utilisateur peut jouer sur les différentes bornes d'arcade proposées et consulter les scores des autres utilisateurs mis à disposition sur un tableau des scores dans la salle.

Chaque borne propose un jeu particulier. Tous les jeux sont codés en langage C à l'aide de la librairie SDL pour la partie graphique et la librairie CURL pour la partie réseau. Parmi ces jeux, se trouve Flappy Bird, un jeu vidéo d'obstacles faisant intervenir un oiseau qui slalome entre des tuyaux qu'il doit éviter. Il est également possible, sur une autre borne, de jouer à Astéroïde, un jeu qui consiste à diriger un petit vaisseau dans le but de détruire les astéroïdes qui s'approchent de lui, et ce, à l'aide des armes à disposition sur le vaisseau. Un autre jeu que l'on trouve dans la salle est Tetris, un jeu bien connu de puzzle dans lequel des briques descendent verticalement pour s'assembler. Dans notre version du Tetris, nous avons ajouté quelques bonus qui donnent une variante au jeu. Le joueur peut également trouver un Démineur sous sa forme la plus basique, dont le but est de localiser les mines cachées dans un champ de mines virtuel. Par ailleurs, se trouve également sur l'une des bornes, le jeu Snake, connu également, dans lequel une succession de formes représente un serpent qui slalome dans une zone et qui doit manger des fruits pour voir sa forme s'allonger. Notre version de Snake comporte, comme Tetris, des variantes. Enfin, le joueur peut trouver le jeu nommé Space Shooter, dans lequel il dirige un vaisseau et doit détruire des vagues de vaisseaux ennemis à l'aide d'armes. Il est à préciser que nous reviendrons ultérieurement sur le détail des jeux pour en expliquer les variantes.

Les performances et scores de chaque jeu sont sauvegardés dans une base de données et sont ainsi visibles pour les joueurs, ce qui leur permet de se

comparer aux autres utilisateurs. Il n’y a pour ainsi dire pas de fin à ce jeu. Mais s’il fallait en définir une, l’objectif pourrait être d’obtenir le meilleur score sur chacune des bornes.

2 Organisation du travail

Le projet que nous avons mené a été qualifié d’ambitieux en raison de la densité du travail à y effectuer. En effet, il a été question de réaliser plusieurs petits projets dans un seul projet. Pour mener à bien ce travail, nous avons formé une équipe de quatre personnes et nous sommes répartis les tâches selon les compétences plus ou moins élevées de chacun. La partie qui suit décrit dans le détail le travail effectué par chaque membre.

Avant toute chose, il est nécessaire de préciser que l’élaboration de ce projet repose sur deux entités distinctes : d’un côté les six petits jeux chacun composé de deux difficultés, de l’autre, la salle d’arcade en trois dimensions.

Hugo s’est occupé intégralement de l’élaboration de trois jeux : Snake, Tetris et Space Shooter, et a participé à l’élaboration du jeu Astéroïde. Il a également participé à la conception du son dans la salle d’arcade, ainsi que la création de certaines textures de la salle, notamment sur les bornes. De plus, il a aidé à la conception du launcher (fenêtre de connexion pour l’utilisateur avant de rentrer dans la salle d’arcade).

Rayyan, quant à lui, s’est occupé du jeu Astéroïde en deux difficultés.

David a travaillé à la conception intégrale du jeu Démineur et s’est également attelé à la rédaction du présent rapport.

Enfin, Samy a réalisé le jeu Flappy Bird dans son entièreté. Il a aussi élaboré la salle d’arcade en trois dimensions ainsi que le son de cette dernière, les déplacements du joueur à la première personne et les interactions possibles entre le joueur et la salle : principalement le fait de pouvoir utiliser les bornes et donc de lancer les jeux et de pouvoir se servir du tableau des scores. De plus, il s’est occupé de la base de données SQL où sont sauvegardés les comptes utilisateurs de chaque joueur ainsi que les scores sur chaque jeu. Il a également pris en charge la totalité de l’aspect réseau, c’est-à-dire le site internet qui comprend la page d’inscription, la page où l’on peut voir les scores, et celle où l’on peut télécharger le jeu. Il a aussi créé une API (Application Programming Interface) permettant la communication simplifiée du programme avec la base de données afin de rendre le jeu accessible en ligne.

L’organisation de notre travail s’est principalement basée sur l’utilisation de GitHub. Cet outil a l’avantage de permettre une permanente mise à jour du projet au gré des perpétuelles modifications de chaque membre du groupe. De plus, la rédaction du présent rapport a pu être menée à bien grâce au

site internet Overleaf qui offre la possibilité de créer un fichier *latex* et de le convertir aussitôt en pdf. Il permet également de disposer du document en partage entre les quatre membres du groupe.

3 Fonctionnalités et concept du projet

Cette partie a pour objectif de relater les fonctionnalités de l'ensemble de notre projet de la façon la plus détaillée et cohérente possible.

3.1 Installation

Tout d'abord, il est à savoir que le jeu est disponible et accessible à toutes et à tous l'adresse suivante : `nineteen.recognizer.fr`.

Ce site contient trois pages différentes : une page d'accueil où sont affichés tous les meilleurs scores de chaque jeu pour chaque difficulté ; une page d'inscription ; et enfin une page téléchargement où est disponible le programme pour les trois principaux systèmes d'exploitation que sont Mac OS, Windows et Linux. Il est à noter que cette page met à disposition un feedback qui permet à chaque visiteur de laisser ses remarques et suggestions à propos du jeu, ces dernières pouvant potentiellement aider à l'améliorer. A partir de la page téléchargement, l'utilisateur, s'il est intéressé par le jeu, peut donc télécharger l'exécutable sur le système d'exploitation qu'il utilise. L'installation se lance alors. Lorsque celle-ci est terminée, l'utilisateur se trouve face au launcher, c'est-à-dire ce qui constitue le début réel du programme.

3.2 Environnement

Une fois inscrit à l'aide d'un pseudo, d'un mot de passe et d'une adresse mail, l'utilisateur peut se connecter et entrer dans la salle d'arcade.

La salle d'arcade offre à l'utilisateur diverses possibilités d'interactions. Pour commencer, il peut apprécier les décors de la salle en se déplaçant à son gré et ce, rappelons-le, à la première personne. La salle d'arcade contient des éléments de deux ordres : certains font simplement partie du décors et ne peuvent faire l'objet d'aucune utilisation, et d'autres, les principaux, constituent les supports des jeux et sont donc interactifs. Ainsi, l'utilisateur pourra constater la présence d'un billard, d'un comptoir, de toilettes, et de quelques canapés, ces éléments faisant office de décors. En ce qui concerne les objets interactifs, certains sont de moindre importance, tels que la porte des toilettes, la radio que l'utilisateur peut manipuler pour l'éteindre ou l'allumer, et le tableau des scores qui a simplement une visée optique et informative. Quant aux éléments principaux qui concernent directement le jeu, ce sont

les bornes. En effet, la salle d'arcade contient au total douze bornes actives réparties d'un côté ou de l'autre de l'espace. Chacun des six jeux est accessible en deux versions de difficulté différentes. Ainsi, d'un côté de la salle se trouvent six bornes proposant les six petits jeux à un niveau de difficulté moyen, et de l'autre côté, six autres bornes proposent les mêmes six petits jeux dans un niveau de difficulté plus élevé. Enfin, le joueur peut bénéficier d'un environnement sonore. En effet, pour une immersion plus grande, il entend ses bruits de pas lorsqu'il se déplace, le bruit de la porte des toilettes s'il vient à l'utiliser, mais également trois musiques d'ambiance différentes selon l'endroit de la salle où il se trouve. Le son de ces musiques d'ambiance est de type stéréo, avec une spatialisation du son. Le joueur sait donc d'où vient la musique.

Il semble nécessaire de justifier les choix que nous avons fait concernant la salle d'arcade. L'objectif est de recréer l'ambiance des salles d'arcade des années '90 et de permettre au joueur la meilleure immersion possible. C'est pour cela que nos choix se sont portés sur les éléments suivants : la salle en trois dimensions, le déplacement du joueur à la première personne, les bruits de pas et la musique en stéréo.

En ce qui concerne notre volonté d'établir un tableau des scores, elle s'explique par le désir de permettre plus de compétition entre les joueurs, de pousser les joueurs à se surpasser, et donc de créer une communauté autour du jeu.

3.3 Description des jeux

Une fois qu'il a pris connaissance de l'ensemble des éléments de la salle, l'utilisateur peut enfin se placer devant la borne de son choix et lancer le jeu correspondant.

Il est donc à présent nécessaire d'expliquer dans le détail le fonctionnement et les règles de chaque jeu. Tous les jeux que nous avons utilisés pour ce projet sont des jeux qui existent déjà. Certains d'entre eux disposent ici de variantes qui sont le produit de notre propre création. Le choix d'octroyer deux difficultés à chacun des jeux repose sur notre désir d'offrir aux joueurs l'opportunité de plus de challenge.

3.3.1 Flappy Bird

Flappy Bird est un jeu en deux dimensions dans lequel l'utilisateur dirige un oiseau. Ce jeu met à l'épreuve l'agilité du joueur qui doit faire voler l'oiseau en appuyant sur la barre espace du clavier. Précisons que l'oiseau, sans intervention de la part du joueur, c'est-à-dire sans pression de la barre espace, tombe automatiquement. Une unique pression de cette barre espace provoque simplement une petite impulsion vers le haut, si bien que finale-

ment, le jeu entier consiste à se servir continuellement de la barre espace pour maintenir l'oiseau en vole. Si la qualité mise à l'épreuve dans ce jeu est principalement l'agilité, c'est parce que le joueur doit éviter les tuyaux qui se posent en obstacle devant lui au fur et à mesure que le décor bouge dans une impression de mouvement. Chaque confrontation à un obstacle correspond au franchissement de deux tuyaux se dressant verticalement, l'un venant du bas, l'autre du haut. L'utilisateur obtient un point pour chaque obstacle franchi. La fin du jeu intervient lorsque l'oiseau touche un élément du décors, qu'il s'agisse du sol ou des tuyaux en question. Le score final correspond donc au nombre total d'obstacles franchis avec succès. L'espace entre les deux tuyaux et la vitesse à laquelle défile le décors changent en fonction de la difficulté choisie : dans le mode hardcore, la vitesse de défilement est plus élevée et l'espace entre les tuyaux est réduit.

Nous avons choisi ce jeu dans l'idée de garder le jeu mondialement connu à l'identique. Il est en effet déjà très complet dans sa version d'origine, et c'est à notre sens la simplicité du jeu qui rend le tout addictif. Il ne nous a donc pas semblé nécessaire d'ajouter des variantes.

3.3.2 Démineur

Le jeu Démineur est, parmi tous ceux que nous avons choisi, le plus connu mais également le plus simple dans la compréhension des règles. L'utilisateur se trouve face à une grille en deux dimensions représentant un champ de mines. Il doit alors, à l'aide des clics de sa souris et grâce aux indications chiffrées sur certaines cases de la grille, découvrir toute la grille sans toucher une seule mine. Le joueur perd la partie dès lors qu'il touche une bombe. Le fonctionnement est donc relativement binaire, dans le sens où seulement deux alternatives sont possibles : soit le joueur ne touche aucune bombe et découvre la grille entière, soit il touche une seule bombe et la partie s'arrête. Le score final correspond en fait au temps mis par l'utilisateur à utiliser la grille, le challenge constitue donc le fait de découvrir la grille le plus vite possible. A chaque fois qu'une partie est recommencée, il va de soi que les mines sont disposées à des endroits différents. Ce jeu comporte deux difficultés, à savoir que dans le niveau le plus difficile, il y a un nombre plus élevé de bombes.

Notre Démineur n'a subi aucune modification par rapport à la version originale car nous avons considéré qu'elle se suffisait à elle-même. Nous avons choisi ce jeu pour sa grande popularité, nous étions sûrs qu'à peu près tout le monde saurait déjà l'utiliser.

3.3.3 Snake

A l'instar du Démineur, le jeu Snake est également connu du grand public. Le joueur dirige un serpent d'une certaine taille qui grossit au fur et à mesure qu'il avale les aliments qui apparaissent sur l'écran. Il faut faire en sorte que le serpent ne se touche pas lui-même, c'est-à-dire qu'il ne touche pas sa queue, ni les bords de l'écran, ce qui mettrait fin à la partie. Chaque aliment avalé entraîne, comme nous l'avons dit, le grossissement de la taille du serpent, mais également l'augmentation du nombre de points et une légère accélération du serpent. Le score augmente pour chaque aliment mangé : plus le joueur résiste dans la durée sans toucher quoi que ce soit, plus il a la possibilité de manger des aliments et donc de voir son score augmenter. Bien évidemment, avec la durée, la difficulté s'accroît puisque le serpent est de plus en plus long et le risque de toucher sa queue ou les bords de l'écran est plus élevé. D'autant plus que plus le temps passe, plus la vitesse à laquelle il se déplace s'accélère.

Il nous faut maintenant préciser la diversité des aliments auxquels le serpent à affaire. En effet, il y en a au total 25 et ils ont chacun leurs propres caractéristiques, donnant un certain nombre de points, faisant grossir différemment, et accélérant différemment.

Certains fruits ont des attributs spéciaux, comme les glaces qui permettent de ralentir la vitesse du serpent ou la tasse de café qui force l'accélération temporairement mais donne un bon nombre de points. Il existe également 7 bonus : un type d'aliments à part qui ne fait pas grossir le serpent et ne rapporte pas de points de base, mais octroie des effets particuliers :

- La bombe : Réduit la taille du serpent.
- La plume : Ralentit le serpent.
- Le coffre : Fait apparaître 6 fruits autour du coffre.
- L'arc-en-ciel : Fait apparaître 10 fruits en arc-en-ciel.
- La potion mauve : Protège de la prochaine collision avec la queue.
- La potion verte : Raccourcit progressivement le serpent (durée infinie).
- La potion jauge : Donne 30000 points.

Il est à noter que la bombe et la potion verte peuvent tuer le serpent si ces derniers détruisent sa tête. Manger des fruits permet d'augmenter la jauge fruitée, visible à droite de l'écran. Plus cette jauge est remplie, plus les fruits et les bonus qui apparaissent peuvent être rares et donc donner plus de points ou des effets plus puissants. Laisser un fruit disparaître sans le manger fait descendre cette jauge.

Il y a dans ce jeu comme dans tous les autres deux niveaux de difficulté, avec une version normale du jeu, qui est celle qui vient d'être décrite, et une

version difficile. Dans le niveau le plus difficile, le mode de jeu change en beaucoup de points. Les fruits apparaissent en très grand nombre, et manger un fruit fait perdre des points et fait descendre la jauge. Cependant, en laissant les fruits disparaître, le joueur gagne des points et fait augmenter la jauge. Le but ici est donc d’esquiver les fruits qui apparaissent de plus en plus vite. De plus, manger un fruit accélère le serpent bien plus que dans le mode normal. Les bonus ne sont pas impactés.

Il semble ici nécessaire de justifier les variantes que nous avons choisi d’ajouter. Notre version de Snake est esthétiquement plus élaborée que le jeu dans sa version la plus classique. Nous avons proposé au joueur un nombre relativement élevé d’éléments à manger, contenant des bonus et des malus. Ceci a pour but d’offrir plus de contenu et de diversité.

3.3.4 Tetris

Ici, le joueur est de nouveau confronté à un jeu qui est bien connu, Tetris. Des pièces de différentes formes et de différentes couleurs venant du haut de l’écran défilent pour se poser en bas de l’écran et s’emboîter de façon plus ou moins harmonieuse, formant ainsi une sorte de petit mur. Les pièces sont formées de quatre blocs. Le but du jeu est de réussir à former des lignes complètes de ces blocs, c’est-à-dire qui ne comprennent aucun trou, afin de faire disparaître ces dernières. Le joueur peut tourner les pièces dans un sens ou dans l’autre, accélérer la chute ou bien faire une chute instantanée de la pièce. Le jeu s’accélère dans le temps, les pièces descendent alors plus vite, se fixent plus vite au sol, et commencent à partir plus rapidement. Une jauge à droite indique la vitesse actuelle du jeu. La partie se termine lorsque le joueur n’arrive plus à former des lignes de blocs et que l’empilement des blocs, autrement dit le mur, a atteint le haut de l’écran. Chaque ligne de blocs formée rapporte 100 points. Si plusieurs lignes sont complétées en même temps, chaque ligne supplémentaire rapporte le double de la ligne précédente. Par exemple, en complétant 3 lignes en même temps, le joueur remporte 100 points pour la première ligne, 200 pour la deuxième et 400 pour la 3ème !

Notre jeu comprend quelques variantes comparé au Tetris original. En effet, dans notre version, compléter une ligne de blocs de couleur unie multiplie les points par dix. Le jeu dispose également de plusieurs bonus qui sont ancrés dans certains blocs. Lorsque qu’un bloc comprenant un bonus est complété, le bonus se déclenche. Il existe 5 bonus et 2 malus dont les effets sont les suivants :

- Bonus carré : Remplit 10 trous dans les lignes du bas.
- Bonus flèche : Supprime les trois dernières lignes du jeu.

- Bonus étoile : Multiplie par deux les points de la ligne.
- Bonus comète : Augmente de 500 les points obtenus.
- Bonus escargot : Ralentit la vitesse de défilement des blocs.
- Malus éclair : Accélère la vitesse de défilement des blocs.
- Malus cube : La prochaine pièce sera deux fois plus grande.

Le mode hardcore du jeu est quasiment identique à celui-ci, si ce n'est que les pièces ne sont plus formées de quatre blocs mais de cinq. Ce qui, mine de rien, augmente grandement la difficulté du jeu.

Notre version du Tetris est plus fluide que la version originale dans le déplacement des pièces, ce qui rend certainement le jeu plus agréable à jouer. Le système de bonus/malus que nous avons adopté permet, comme avec Snake, une plus grande diversité. De plus, cela offre des moyens de survie plus longs ainsi que des facteurs chance nécessaires pour faire de bons scores, ce qui incite le joueur à ré-essayer plusieurs fois.

3.3.5 Astéroïde

Astéroïde est un jeu en deux dimensions où l'utilisateur dirige un vaisseau dans l'espace à l'aide des touches directionnelles du clavier. Le but du jeu est de détruire les astéroïdes qui apparaissent et se déplacent sur l'écran, et de ne pas heurter un astéroïde. Pour cela, le vaisseau dispose de plusieurs armes, dont il peut obtenir des munitions en tant que bonus, et ayant chacune leurs particularités présentées dans le tableau ci-dessous.






Nom	Dégats	Fréquence	Vitesse	Durée de vie	Spécificité	Image
Basique	=	=	=	=	Munitions infinies	
Zigzag	+	+	+	-	Se déplace aléatoirement	
Tête chercheuse	++	- -	-	++	Traque les astéroïdes	
Glace	0	=	+	=	Gèle les astéroïdes	
Laser	/	/	/	/	Laser en continu	

TABLEAU 1 : Description des armes

Le joueur dispose également d'une bombe, qu'il peut utiliser une fois pour détruire tous les astéroïdes présents sur l'écran.

Les types d'astéroïdes à détruire sont au nombre de six et rapportent un nombre de points différent selon leur type.

Type						
Points	50	100	200	300	400	500

TABLEAU 2 : Points des astéroïdes

Le type d'astéroïde dépend de la difficulté actuelle du jeu, qui augmente au fil de la partie. Les points de vie des astéroïdes augmentent alors, et ils apparaissent en plus grand nombre. Chaque astéroïde, peu importe son type, possède des points de vie aléatoires, dont l'intervalle dépend de la difficulté, et a une taille et vitesse qui dépendent de ses points de vie. De plus, les gros astéroïdes se séparent en deux plus petits astéroïdes lorsqu'ils sont tués et que la difficulté est assez élevée. Ces astéroïdes fils ont la moitié des points de vie de l'astéroïde père, la même vitesse, et octroient la moitié des points lorsqu'ils sont tués. Le jeu prend fin lorsque le vaisseau heurte un astéroïde.

La version hardcore de ce jeu est totalement différente. Il n'y a pas d'armes. Les astéroïdes ne sont donc ici plus à détruire mais à éviter. Ces astéroïdes arrivent par vague, ils apparaissent chacun d'un côté aléatoire et se déplacent en ligne droite vers le côté opposé, où ils disparaissent ensuite. Le joueur gagne des points lorsque les astéroïdes sortent de l'écran, avec le même fonctionnement d'attribution de points que dans le mode normal.

Une particularité du jeu est que le vaisseau est difficile à manier en raison de la reproduction du manque de gravité dans l'espace. De plus, les missiles, le vaisseau et les astéroïdes passent de l'autre côté du terrain quand ils arrivent d'un côté, ce qui constitue une mécanique de jeu de plus à maîtriser. Pour s'améliorer, le joueur doit donc apprendre à manier et à maîtriser le vaisseau de la façon la plus efficace possible, ce qui apporte de la difficulté à l'ensemble du jeu. Cela est d'autant plus valable dans le mode hardcore où tout est basé sur la capacité du joueur à maîtriser son vaisseau efficacement, étant donné l'absence d'armes et la nécessité d'éviter les astéroïdes.

3.3.6 Space Shooter

Dans Space Shooter, jeu en deux dimensions, l'utilisateur dirige un vaisseau qui peut se déplacer vers les côtés et vers le haut/bas et tirer des missiles. Il est attaqué par des vaisseaux ennemis qui apparaissent de chaque côté et lui tirent dessus avec leurs armes. Il y a pour l'instant trois types d'ennemis, chacun a des armes et un comportement uniques avec une part d'aléatoire dans leurs actions. Le joueur peut obtenir d'autres armes en appuyant sur tab, et il y aura plus tard un système d'obtention d'armes lorsque l'on tue assez d'ennemis.

Ce jeu n'est pas encore complet, mais les éléments y figurant sont déjà bien intégrés et ne sont pas amenés à être changés, sauf pour l'équilibrage.

Dans notre version de ce jeu, ce qui est intéressant réside dans le fait qu'il y a beaucoup moins d'aspect chance, tout est basé sur le talent et l'adresse, ce qui peut attirer les joueurs les plus exigeants.

Pour rappel, à la fin de chaque partie de chacun des jeux, le score final de l'utilisateur est envoyé vers une base de données qui met à jour le tableau des scores consultable dans la salle d'arcade. Le tableau des scores affiche le nom des utilisateurs en face du score correspondant, pour chaque jeu et pour chaque difficulté, et ce, dans l'ordre décroissant, c'est-à-dire allant de haut en bas du meilleur score au moins bon.

4 Développement

A présent que les fonctionnalités de notre programme ont été détaillées, il est temps d'en venir au coeur du développement de notre travail. Autrement dit, cette partie explique les méthodes que nous avons employées pour mettre le programme au point.

4.1 Création de la salle d'arcade

La salle d'arcade a été conçue sur Blender, logiciel de modélisation en trois dimensions. Tous les éléments de notre salle ont été conçus par nos soins exceptés le billard, le canapé et les trois radios. En effet, ces modèles ont été récupérés sur le site suivant : <https://free3d.com/> Étant donné que nous sommes partis d'un espace vide, nous avons pu créer la salle comme bon nous semblait, la seule contrainte que nous avons eu a été le nombre de bornes présentes dans la salle. Il a donc fallu créer un espace suffisamment grand pour y placer ces bornes, pour que le joueur puisse s'y déplacer et que l'on soit libre de rajouter quelques éléments décoratifs rappelant une salle d'arcade (accueil, canapé, toilettes, billard etc..).

Concernant les textures présentes dans la salle d'arcade, seules celles présentes sur les bornes de jeu sont le produit de notre propre conception. D'autres textures, telles que celles présentes sur le sol, le plafond ou les murs ont été trouvées sur d'autres sites et ont subi des modifications de notre part. Les modifications apportées ont été faites dans le but de convenir au mieux au design envisagé, et sont les suivantes : ajout d'éléments supplémentaires, changement de certaines couleurs, modification de la luminosité, répétition des textures dans l'espace. D'autres textures trouvées sur divers sites sont utilisées dans notre salle sans avoir subi de modifications. Ce groupe de textures rassemble tout ce qui se rapporte aux matières "simples" telles que le marbre, le bois, les murs en briques blanches, la moquette verte du billard ou les cadres présents sur les murs.

Pour appliquer ces textures sur les objets en trois dimensions, nous avons effectué ce que l'on appelle du *UVmapping*. Le *UVmapping* consiste en la méthode suivante : grâce à une fonction du logiciel Blender, l'objet en trois dimensions est projeté en deux dimensions (nous avons donc la vision du patron de cet objet). A partir de cette projection, il est possible d'appliquer les textures plus facilement sur ces objets. L'ensemble des textures de notre salle d'arcade a été appliqué grâce à cette méthode. Le rendu final de ces manoeuvres constitue la salle modélisée.

4.2 Importation du fichier 3D

Pour l'importation du fichier 3D, nous avons fait appel à la fonction *aiImportFile* (que l'on trouve dans la librairie *assimp*). Cette fonction prend en paramètre notre objet 3D, en l'occurrence notre salle d'arcade, et renvoie un pointeur sur une structure qui est exploitable en langage C. L'exploitation des données présentes dans cette structure se fait de manière récursive, en parcourant chaque maillage de chaque objet 3D.

Concernant OpenGL, il suffit donc simplement d'utiliser *glBegin* et de dessiner la mesh trouvée en faisant appel à *glNormal* et/ou *glVertex* et de se servir de *glEnd*. Cela a pour effet de dessiner la mesh. Le fait de réaliser cette opération en boucle et de manière récursive permet de dessiner l'intégralité de l'objet dans notre fenêtre OpenGL. Le principe est le même pour les textures, cela revient à dessiner une image en faisant appel à la fonction *glTexCoord2f*.

4.3 Déplacements et interactions dans la salle d'arcade

Les déplacements se font à l'aide de la gestion d'événements, ces derniers étant constitués par les manipulations faites sur le clavier et la souris, et ce, sur Mac et Windows (à noter que nous avons rencontré une petite complication sur Linux pour utiliser le curseur de la souris lors des mouvements

de caméra). Les évènements sont tous gérés dans une fonction mouvement-Camera qui fait appel à *SDL_GetKeyboardState*. Il nous est ainsi possible de gérer les différents évènements dans un même temps comme l'appui sur plusieurs touches à la fois. Côté 3D, et donc OpenGL, le déplacement de la caméra se gère grâce à la fonction *gluLookAt* nous permettant de placer le point de vue où nous le souhaitons. Il faut spécifier la position de la caméra sur l'axe tridimensionnel ainsi que le regard de la caméra. Le placement de la caméra revient donc à déplacer tout l'environnement pour qu'il soit centré sur la caméra et orienté sur l'axe du regard.

Les interactions avec l'environnement se font par "détection". Plusieurs chiffres magiques sont présents comme la position des bornes, la position des radios ainsi que celle de la porte des toilettes, permettant de créer une zone 2D propice à l'interaction. Si les coordonnées du joueur sont comprises entre telle et telle coordonnée, alors une interaction est envisageable. Cela nous a permis de mettre facilement en place un environnement interactif où la seule limite était notre imagination ainsi que le temps restreint dont nous disposions pour finir le projet.

4.4 Les lumières

Fonctionnalité optionnelle en terme de développement, mais qui a le don de sublimer un environnement 3D, l'ajout de lumière repose sur différents critères établis par la librairie OpenGL. Il est nécessaire de sélectionner le type de lampe voulu puis, selon la sélection, de lui appliquer différents réglages, comme l'atténuation, la position de la lampe, sa direction d'éclairage, ou encore sa couleur sous forme de trois paramètres (ambient, spéculaire, diffusion). Il est nécessaire d'activer au préalable les différentes sources de lumières ainsi que de prévenir OpenGL que nous allons utiliser les lumières à l'aide des fonctions *glEnable(GL_LIGHT '0 à '7')*, *glEnable(GL_LIGHTING)*. Cependant, il n'est pas possible de placer les réglages de la lumière n'importe où, ils doivent impérativement se trouver après le déplacement de la caméra et avant le premier appel de *glBegin* permettant de prévenir OpenGL que nous allons maintenant dessiner.

4.5 Spatialisation des sons ambiants

L'ambiance de la salle tridimensionnelle repose sur l'adaptation soignée du volume de chaque son présent dans la salle, permettant au joueur de situer sa localisation avec précision même les yeux fermés. Afin d'arriver à ce résultat, il était nécessaire de disposer d'une position précise du son ainsi que de la caméra afin de pouvoir en déduire une distance qui nous a permis de réduire ou de diminuer les sons à l'aide de la fonction *Mix_volume*. À l'aide d'un calcul entre la position du son, celle de la caméra et celle de l'angle de vision de la caméra, nous avons pu en ressortir une intensité sonore à

réglé dans l'oreille gauche et droite, permettant de créer un son d'ambiance facilement localisable à l'aide de la fonction *Mix_setPanning*. Cette fonction est donc utilisée pour les trois musiques d'ambiance de la salle et pour le son de la porte des toilettes. Le son des pas étant toujours localisé sur la position de déplacement du joueur, il n'était pas nécessaire de lui appliquer un réglage particulier. Toutes les fonctions utilisées ici pour le son sont issues de la librairie *SDLMixer*.

4.6 Transition tri-dimension / bi-dimension

Il s'agit ici d'expliquer de quelle manière le programme passe de l'environnement 3D de la salle d'arcade à l'environnement 2D d'un jeu choisi. Étant donné l'importance des jeux dans la conception du projet, et ces jeux étant développés en bi-dimension, il était impératif de créer la transition la plus naturelle possible d'un environnement à l'autre. Le contexte d'affichage d'une fenêtre est différent d'un moteur graphique à l'autre (de SDL pour la 2D à OpenGL pour la 3D), et ces deux contextes ne sont pas compatibles sur la même fenêtre. Il faut donc détruire les textures associées de la salle d'arcade et le *SDL_GLContext* et créer un *SDL_Renderer*. Le programme fait l'inverse lors du passage de la 2D à la 3D.

4.7 Utilisation de SDL dans les jeux

Notre utilisation de la librairie SDL a été à la fois basique mais complète. En nous basant sur un modèle de boucle Événements - Gestion événement - gestion jeu - affichage, nos différents jeux sont tous très similaires dans leur utilisation de SDL. La librairie SDL basique nous a permis de créer la fenêtre de jeu, d'y afficher des éléments simples, tels des rectangles, ou encore d'avoir une gestion du temps en jeu. Cependant, nous avons besoin d'autres éléments pour nos jeux, comme afficher des textures, du texte ou encore jouer de la musique.

4.7.1 Écriture de texte

L'utilisation de *SDL_TTF* nous a permis de charger des polices d'écriture, qui sont ensuite transformées en surfaces et textures, que nous avons pu manipuler pour afficher les différents textes que nous avons besoin d'écrire, tels que les scores, les indicateurs de bonus, ou encore le temps passé dans Démineur, tout en pouvant faire des effets sur leur taille, position ou transparence pour faire des animations plus fluides.

4.7.2 Affichage d'images

La librairie *SDL_image* sert à charger toutes nos textures en jpg ou png. Nous les avons organisées en feuille (sprite) de manière à pouvoir les

regrouper et les parcourir le plus efficacement et simplement possible grâce à des variables de nos programmes, comme par exemple l'id des fruits dans Snake qu'il suffit de multiplier par la largeur d'un fruit pour tomber sur la bonne texture.

4.7.3 Jouer des sons

Quant à *SDL_Mixer*, elle permet de charger nos effets sonores et de les jouer en réaction aux événements du jeu. Nous avons également dû faire attention à l'utilisation des channels et à leur volume, pour ne pas empiéter sur les sons de la salle, et avoir un son équilibré.

4.8 Réseau et sécurité

4.8.1 Protection des variables sensibles

Il nous a semblé nécessaire dans ce projet de protéger plusieurs variables sensibles contre toute personne mal intentionnée qui souhaiterait modifier le comportement de notre programme. Nous avons donc créé des fonctions qui ont pour but de protéger ces variables. Ces fonctions détectent toutes modifications extérieures au programme sur un certain nombre de variables.

Un tableau de nombres aléatoires est généré à l'initialisation d'une variable permettant de créer un homonyme hasher de notre variable. La modification ou l'utilisation de variables protégées doit intervenir après la vérification de l'égalité nouveau hashage = ancien hashage et dans le cas d'une modification, il est conseillé de régénérer de nouveaux nombres aléatoires afin d'accroître la sécurité.

4.8.2 API dédiée

La décision de rendre notre application uniquement accessible en ligne nous a confronté à la nécessité de trouver une solution technique afin de rendre cela possible. Ayant à notre disposition un serveur web, la solution retenue a été de développer un ensemble normalisé de fonction en PHP servant de façade pour notre application. "De manière plus générale, on parle d'API à partir du moment où une entité informatique cherche à agir avec ou sur un système tiers." ¹. Techniquement parlant, cela nous a apporté la possibilité de mettre en relation notre base de données avec notre application. Il était uniquement nécessaire de connaître les besoins de notre application vis-à-vis de la base de données, c'est-à-dire : quelle information devait-elle recevoir ou apporter à la base de données ? A partir de cela, nous avons pu précisément déterminer les éléments qui seront présents dans notre API et

1. Wikipedia

commencer le développement en PHP ainsi que la conception de la base de données.

4.8.3 Construction d'une requête réseau

Plusieurs services pouvant être accessibles, il est avant tout nécessaire de comprendre quel service spécifique nous voulons atteindre. Chaque fonction d'envoi de requête fait appel à la construction de la requête et se charge donc de passer les bons paramètres à la fonction afin d'avoir une requête correcte en sortie. La fonction se charge elle-même, si nécessaire, de générer une clé sécurisée qui permettra au serveur d'authentifier la requête comme provenant de l'application. Une fois la requête construite, il reste la phase d'envoi, qui est gérée par la bibliothèque CURL. Il suffit simplement de spécifier l'url à la fonction *curl_easy_setopt* ainsi que les paramètres de la requête associés à la méthode d'envoi. Dans notre cas, il s'agit uniquement des méthodes post. Cela revient tous simplement à utiliser l'API prévue à cet effet.

4.8.4 Communication avec l'API Dédiee

Une fois la requête précédemment construite et préparée pour l'envoi avec la fonction *curl_easy_setopt*, il est nécessaire de la transmettre à l'aide de la fonction *curl_easy_perform*. Il est de notre responsabilité de vérifier que la valeur de retour de la fonction nous indique bien que tout s'est déroulé avec succès. Une fois la requête envoyée et traitée par le serveur, il nous suffit simplement de lire la valeur de retour. Nous connaissons au préalable la topologie de la réponse, et puisqu'il s'agit d'une API, il nous est possible de réagir en fonction des différents cas de réponse et ainsi de mettre en place des actions spécifiques à chaque cas de retour.

4.8.5 Authentification de la requête

À l'aide de certains logiciels, tels que Wireshark pour n'en citer qu'un, il est tout à fait possible d'intercepter les requêtes afin de voir le destinataire ainsi que ses différents paramètres. Ainsi, il est ensuite très facile d'envoyer une requête au même destinataire en changeant la valeur de certains paramètres de la requête initiale, comme le score par exemple.

Il était donc nécessaire de se prémunir contre ce type d'attaque en mettant en place un système permettant au serveur web d'authentifier la provenance de la requête. Par conséquent, nous avons pensé un système reposant sur une synchronisation des horloges application/serveur web permettant de générer une clé avec une durée de vie de 1 seconde. Certaines modifications sur les horloges ainsi que le hashage de cette valeur permettent d'augmenter la sécurité de cette méthode.

4.8.6 Conception de la base de données

La conception d'une base de données nécessite de se poser les bonnes questions avant de se lancer pour ne pas se perdre. Il a donc été établi un certain nombre d'éléments qui devaient être présents mais également une liste d'éléments qui pouvaient n'être qu'optionnellement présents. Notre base de donnée est mutualisée. Sa limite étant de 30 connexions simultanées (peut-être plus ou moins selon l'utilisation de l'hébergement mutualisé), l'allègement du nombre de requêtes ne pouvait être qu'un avantage indéniable en terme de coût financier, évitant la location d'une base de données avec une allocation d'un certain nombre de ressources.

4.8.7 Requête SQL

Certaines requêtes demandent la jointure de plusieurs tables et la combinaison de plusieurs fonctions. Afin de simplifier leur compréhension, il faut établir un lot d'explications.

La requête ci-dessous permet de retourner une liste d'utilisateurs ainsi que la somme de leurs scores, tout cela en prenant en compte les différents multiplicateurs associés à chaque jeu afin d'harmoniser le classement général tout en les classant du meilleur au moins bon, puis en les triant par ordre alphabétique.

```
SELECT
    username ,
    FLOOR( SUM( score * multiplicator ) ) AS total
FROM
    nineteen_scores
    NATURAL JOIN
    nineteen_multiplicators
    NATURAL JOIN
    nineteen_players
GROUP BY
    userId
ORDER BY
    total DESC, username
LIMIT
    $limite
OFFSET
    $offset
```

Une variante existe afin de se focaliser sur un jeu et non sur le classement général.

4.9 Conception du site web

Le caractère "en ligne" de notre jeu nous oblige indirectement à ajouter le développement d'un site web à la conception du jeu actuel. Il est tout à fait évident pour l'utilisateur, concernant un jeu en ligne, de pouvoir s'inscrire mais également de télécharger le jeu. Le développement des pages web *download.php* et *inscription.php* étant nécessaire, il nous a semblé intéressant de pouvoir ajouter une page *index.php* permettant de visualiser les classements sur les différents jeux. Seuls quelques éléments *css*, comme les *input* présents sur la page d'inscription avec le design des *input google*, ont été pris depuis le web.

4.10 Méthode de compilation

Plusieurs étapes de développement se sont succédées, et les méthodes de compilation ont évolué au fil de ces étapes. Les jeux étaient d'abord testés indépendamment de la salle, et avaient donc leur propre makefile et leur propre exécutable. Il a donc été privilégié dans ce cas la compilation avec un *main* propre à ce programme et un makefile temporaire le temps du développement. D'un autre côté, il a fallu un programme debugger et donc intégrable dans un makefile ainsi que dans la salle sans apporter d'erreur générale au programme principal.

Le concept de makefile étant propre à un environnement Unix, ce paragraphe exclut la compilation sur Windows. Le makefile est un fichier qui est interprété par la commande *make* afin d'automatiser des tâches comme la compilation d'un projet dans notre cas. Il a l'avantage d'éviter dans un gros projet la lourde tâche de devoir compiler chaque fichier dépendant du *main* un par un, le gain de temps étant par conséquent considérable. Cela n'empêche pas la réalisation de tâches extérieures, comme l'installation des bibliothèques nécessaires à la compilation du projet.

La compilation sur Windows a été réalisée depuis *l'IDE Codeblocks*. Une étape de préparation est nécessaire afin de lier les bibliothèques et d'indiquer à l'IDE les chemins où les trouver. Une fois l'IDE parfaitement configuré, il suffit simplement de lancer la commande *build* afin de créer un exécutable pour Windows. Il faut toutefois placer des fichiers *.dll* dans le dossier d'exécution pour pouvoir lancer l'exécutable.

4.11 Phase de débogage

Le travail de débogage pouvant être très long et rébarbatif, il a principalement été réalisé au fur et à mesure du développement. Nous avons surtout et avant tout tenté d'anticiper, par la gestion d'erreurs ainsi que l'affichage précis des erreurs générées, afin de passer le moins de temps possible sur cette phase de recherche d'erreurs. L'utilisation des logiciels de débogage, comme

lldb ou *gdb*, n'a été réalisée que très rarement en début de projet. Celui-ci devenant lourd, il semblait plus simple de réaliser le suivi de variables ou d'éléments particuliers à l'aide de *fprintf*. Quelques utilisations de la librairie *asserts* peuvent être présentes dans le programme, le plus souvent précédant une allocation.

Le déploiement du programme de bêta test, permettant un débogage à grande échelle et avec un faible coût en terme de temps, nous a permis de faire le plus gros du travail avec l'aide de testeurs compréhensifs et capables de s'investir en prenant le temps de répéter les étapes qui ont pu le faire arriver à une anomalie lors de l'exécution du programme. Cela nous a permis un gain de temps dans ce domaine. La participation au bêta test de la part de certains camarades du groupe de TP mais également de la part de personnes externes au domaine de l'informatique a permis de présenter un projet final de cette ampleur avec une optimisation cross-platform.

4.12 Optimisation du travail de groupe

Étant donné l'étendue de notre projet, la communication entre chaque membre du groupe s'est révélée être d'une importance capitale. Ainsi, pour mener à bien l'ensemble du projet, nous avons mis en place différents moyens de communication spécifiques à chaque élément relatif à l'avancée de notre travail. Afin de se partager le code à plusieurs, pour que chacun puisse le modifier et que la mise à jour soit transmise à nous tous, nous avons utilisé GitHub. Par ailleurs, pour communiquer par écrit, nous avons utilisé Discord, créant plusieurs channels propres à chaque sujet précis du projet. Par exemple, nous avons les channels suivants, nommés ainsi : "avancement", "bugs", "rapport" ou encore "gameplay". Cette organisation nous a permis de savoir où chacun en était, ce qu'il prévoyait de faire, quels étaient les problèmes qu'il rencontrait et plus globalement de connaître l'avancement général du projet sans passer par le code.

5 Conclusion

Pour conclure, il est à présent nécessaire d'établir un bilan d'ensemble de notre projet.

Le corps principal de notre programme est finalement fonctionnel. En effet, les fonctionnalités principales du programme ont été réalisées comme nous l'avions prévu : la salle d'arcade et les jeux présents sont utilisables, et la base de données comprenant les comptes utilisateurs et les scores associés est opérationnelle. Autrement dit, la somme des éléments du programme final fonctionne.

Néanmoins, en plus des six jeux finalement présents dans la salle, nous avons l'intention d'inclure davantage de jeux, tels que Pacman ou encore Piano Tiles (jeu de rythme musical). Cela s'explique par le fait que les jeux finaux ont nécessité une somme importante de travail en raison des différentes variantes à élaborer. De plus, nous avons initialement prévu de créer la salle d'arcade en deux dimensions, ce qui nous aurait laissé plus de temps pour développer d'autres jeux. Nous avons cependant effectué le choix de créer une salle plus élaborée en terme d'esthétique et de qualité immersive, la réalisant finalement en trois dimensions. Ainsi, à défaut d'un nombre important de jeux, nous avons préféré privilégier la qualité de la salle mais également la qualité de seulement certains jeux.

Étant donné le concept du projet, ce dernier est modifiable et améliorable autant qu'on le souhaite, c'est-à-dire que l'on pourra toujours apporter des améliorations aux jeux déjà présents ou tout simplement, ajouter de nouveaux jeux sur les bornes. La salle 3D est quant à elle agrandissable, ce qui pourra permettre de nouvelles bornes, et donc encore de nouveaux jeux. Nous pourrions également améliorer le graphisme complet de la salle d'arcade. Par ailleurs, comme mentionné antérieurement, nous avons sur le site de présentation du jeu, mis à disposition un feedback pour les utilisateurs, feedback qui nous sera très utile et pourra servir de support pour les futures améliorations du jeu. C'est donc un projet qui peut être sans fin, et nous avons d'ailleurs l'intention de continuer à développer ce projet par la suite.

En ce qui concerne notre ressenti final sur le travail de groupe, ce projet nous a apporté un certain nombre de choses. Premièrement, il nous a appris à partager un code à plusieurs et à l'adapter pour qu'il soit compris de tout le monde. Ensuite, nous avons appris à créer des animations et textures travaillées, que ce soit à l'aide du logiciel 3D ou en SDL. Nous avons également appris les bases de Github. Enfin, ce projet a aidé à travailler sur notre capacité d'adaptation. En effet, il a fallu s'adapter à la programmation crossplatform et à la mise en place d'environnements de développement sous différents OS.

Par conséquent, nous sortons de ce projet avec un ressenti positif et des compétences supplémentaires. Ce fut intéressant, enrichissant, en somme, une bonne expérience.