Rapport projet C.M.I.

Cérès Rémi
Groupe 1A1

29 février 2016

Table des matières

Introduction

Dans le cadre du projet CMI, j'ai réalisé l'analyse de trois jeux : le jeu de la vie, puissance 4 et le jeu de plateforme.

J'ai étudié les différents principes, les modélisations, les algorithmes, les spécifications et le temps de réalisation nécessaire à leurs créations. J'ai pu ainsi réaliser une comparaison de ces trois jeux.

Jeu 1

Jeu de la vie

Le jeu de la vie a été imaginé par John Horton Conway durant les années 70. Il appartient à la famille des automates cellulaires. [1]
Un automate cellulaire est un objet mathématique qui évolue selon des règles simples. Il est constitué d'une grille potentiellement infinie dont chaque case (nommée cellule) possède un état. L'état d'une cellule au temps $t + 1$ se calcule par rapport à l'état au temps t des cellules voisines. [2]

Le jeu de la vie est Turing-complet, c'est-à-dire qu'il possèdent une puissance de calcul égal ou supérieur à celle des machines de Turing. Ce jeu peut donc permettre d'exécuté n'importe quel algorithme. [3] [4]

1.1 Principe

Le jeu de la vie se présente sur une grille infinie en deux dimensions. Chaque case de cette grille représente une cellule qui peut être active ou inactive. Les voisines d'une case sont les huit cases qui lui sont adjacentes.

Le jeu est constitué d'une succession de temps. Un temps est l'ensemble des états d'une cellule à un instant donné. Le premier temps est fourni et le passage d'un temps au temps suivant s'effectue selon des réglés qui sont les suivantes :

— Une cellule active à l'instant t , se désactive à l'instant $t+1$ si elle ne possède pas exactement deux ou trois voisines vivantes ; 1.1 1.2
— Une cellule désactivée à l'instant t s'active à l'instant $t+1$ si elle possède plus de trois cellules voisines. 1.3 [5] [1]

1.2 Modélisation

Les cellules ne peuvent prendre que deux états, actives ou inactives. Un booléen ne pouvant prendre que deux valeurs, c'est donc la manière la plus simple de les représenter : par exemple avec la valeur `true` aux cellules actives et `false` aux cellules inactives.

Afin de modéliser la grille nous pouvons utiliser un tableau rectangulaire à deux dimensions de booléens. Cela nous permettra en associant chaque case à une cellule d'accéder facilement à son état, ainsi qu'à celui de ses voisines. [2] [6]

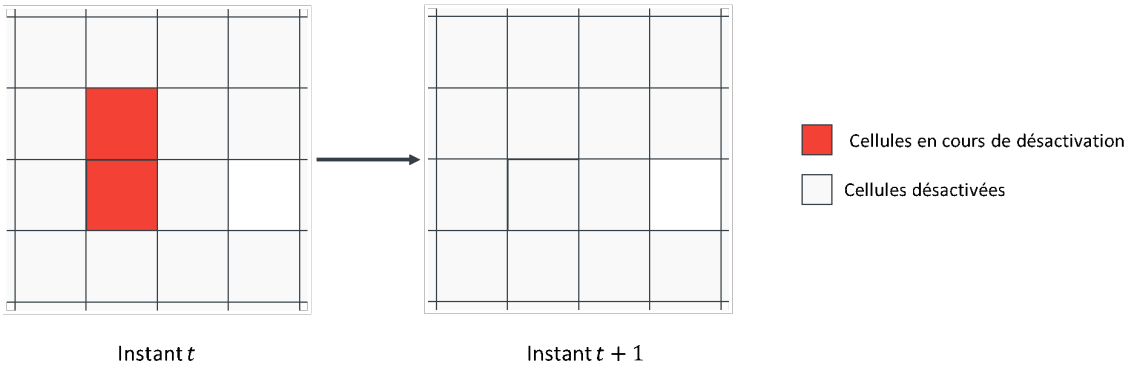

FIGURE 1.1 – cellules désactivées par manque de contact

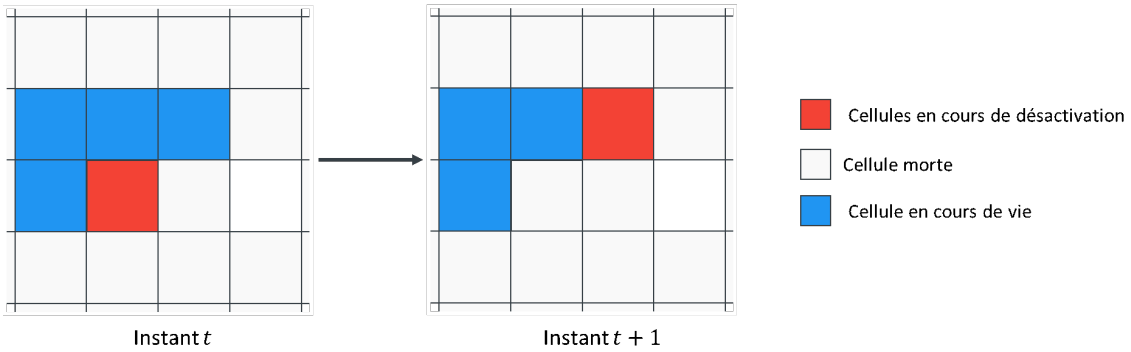

FIGURE 1.2 – cellules désactivées par un surplus de contact

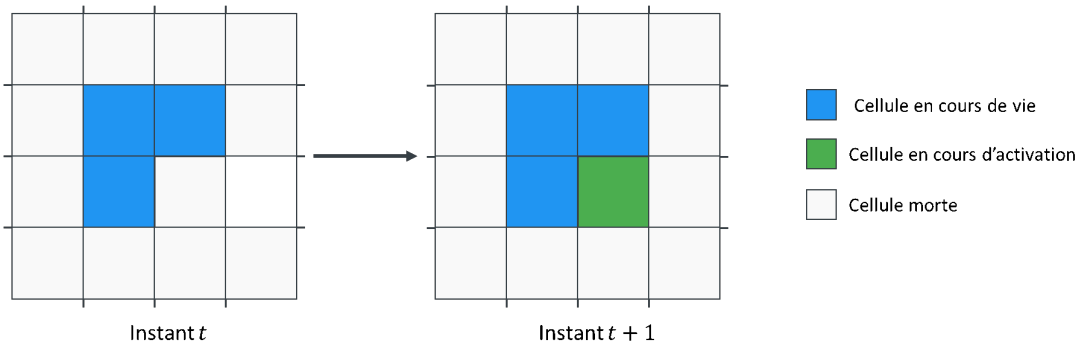

FIGURE 1.3 – cellules désactivées par contact de trois cellules

1.3 Algorithmes

1. **Création d'une grille :** Création de un tableau rectangulaire à deux dimensions de booléens et initialisation de tous ces éléments à `false` ;
2. **Calcul du nombre de cellules voisines vivantes :** En fonction d'une cellule et d'une grille a un instant donné, L'algorithme compte le nombre de cellules voisines vivantes ;
3. **Calcul de l'instant suivant :** En fonction de l'état de l'ensemble des cellules à un instant donné, l'algorithme calcule sur une nouvelle grille de même taille le temps suivant.
   — Si la cellule testé est active et si elle ne possède pas exactement 2 ou 3 voisines vivantes (déterminer par l'algorithme 2), alors elle sera inactive au temps suivant ;
   — Si la cellule testé est inactive et si elle possède plus de trois cellules voisines, alors elle sera active au temps suivant ;
   — Sinon, la cellule tester ne change pas d'états au temps suivant.
4. **Affichage :** L'algorithme parcourt toutes les cellules et représente leurs états sur l'écran.

1.4 spécifications

1.4.1 Version simple

Dans la version simple, le jeu s'affiche dans un terminal. L'utilisateur n'a pas la possibilité d'interagir avec le programme, il n'est donc qu'un simple spectateur.
L'utilisateur ne peut pas contrôler la taille du tableau, ni la position des cellules à l'état initial ou encore la vitesse de défilement des instants. De plus, toutes les cellules à l'extérieur du tableau sont considérées comme inactives.

1.4.2 Difficultés

La modélisation d'un plan infini n'est pas possible, notre plan ne sera donc pas infini. Comme l'on ne connaît pas l'état des cellules en dehors de la grille, la première difficulté va donc être la gestion des bordures.
La solution pour pallier à ce problème, serait de créer un espace torique, c'est-à-dire de considérer que la colonne de gauche et de droite sont voisines, ainsi que les lignes du haut et du bas. [6]

1.4.3 Extensions possibles

De nombreuses extensions sont possibles. L'utilisateur pourrait par exemple, avoir la possibilité de choisir la taille de la grille.
Il pourrait aussi contrôler l'état des cellules en cliquant dessus, ainsi que la vitesse de défilement des instants.
Différentes formes de configurations pourraient être proposées sur l'interface afin de permettre à l'utilisateur d'y accéder rapidement.
La possibilité de charger et de sauvegarder un état à un instant donné serait un plus. Cela permettrait à l'utilisateur d'être acteur du jeu.[5]

L'affichage du jeu pourrait s'effectuer dans une fenêtre et ainsi permettre de rendre le jeu plus visuel et agréable à utiliser.
Il serait également intéressant d'afficher des informations sur le déroulement du jeu tel que le nombre de cellules vivantes avec le tracée d'une courbe en fonction des instants.
Le jeu pourrait également proposer une interface adaptée aux appareils tactiles.

L'optimisation des algorithmes permettrait d'accélérer les calculs et augmenterait la vitesse du jeu. enfin faute de pouvoir créé une grille infinie, le jeu pourrait se dérouler dans un espace torique.

1.4.4 Langage

Le langage de programmation que j'ai choisi afin de développer le jeu de la vie est le C++. En effet, ce langage informatique appris durant le premier semesre de mon cursus possède tous les outils pour réaliser ce projet.

1.5 Organisation

Les différentes étapes de l'organisation du projet pour une durée de développement de 40 heures sont :

1. **Programmation de la base du jeu :** réalisation des algorithme de base et de la version simple du jeu dans le langage choisi. (8 heures) ;
2. **Programmation des extensions :** programmation des différentes extensions. (12 heures) ;
3. **Réalisation de l'interface :** réalisation de l'interface. (12 heures) ;
4. **Tests :** vérification du bon fonctionnement du logiciel. (33 heures) ;
5. **Rédaction du rapport :** (40 heures) 1.4

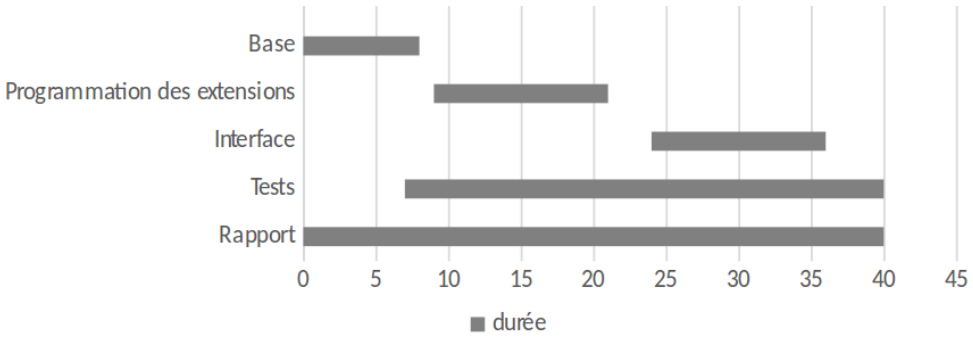

FIGURE 1.4 – Dévelopement du projet jeu de la vie sur 40 heures

Jeu 2

Le Puissance 4

Le jeu «Puissance 4» a été commercialisé pour la première fois par la société « Milton Bradley Company » en 1974.[7] Il fait partie de la famille des jeux de stratégies de combinatoires abstraits.

Un Jeu de stratégie combinatoire abstrait oppose généralement deux personnes qui jouent à tour de rôle. Tous les éléments du jeu sont connus par avance et le hasard n'intervient pas dans le déroulement de la partie. [8]

2.1 Principe

Le Jeu « Puissance 4 » se présente sur une grille comptant six rangées et sept colonnes. Les deux joueurs disposent chacun de vingt-et-un pions d'une même couleur, en général jaune et rouge. A tour de rôle, les joueurs placent un pion en haut d'une des 7 colonnes. Le pion sous l'action de la gravité coulisse alors dans sa colone jusqu'à la position inoccupé la plus basse possible.

La stratégie du jeu est de réussir à aligner quatre pions d'une même couleur, que ce soit horizontalement, verticalement, ou en diagonale et d'empêcher son adversaire de réaliser une ligne.
Le premier joueur qui aligne quatre pions de sa couleur est le gagnant, la partie est terminée. [9]
Si, toutes les cases de la grille sont remplies et qu'aucun des deux joueurs n'a réussi à aligner quatre pions, la partie est terminée et déclarée nulle. [7]

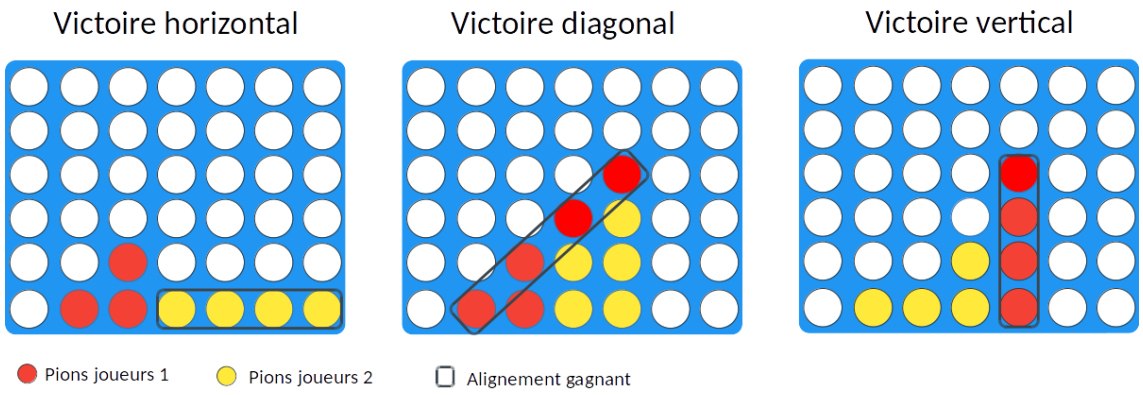

FIGURE 2.1 – conditions de victoires

2.2 Modélisation

Afin de modéliser la grille, nous pouvons utiliser un tableau rectangulaire de valeurs à deux dimensions.

Les cases ne peuvent prendre que trois états, : Le pion du joueur 1, le pion du joueur 2 ou vide. Nous pouvons par exemple représenter les cases vide par la valeurs 0, les cases contenant un pion du joueurs 1 par la valeur 1, et les cases contenant les pions du joueurs 2 par la valeurs 2.

2.3 Algorithmes

1. **Création d'une grille :** Création d'un tableau rectangulaire à deux dimensions de valeurs et initialisation de tous ces éléments à vide (correspondant à la valeurs 0).
2. **Vérification : la colonne est-elle pleine ? :** En fonction d'une colonne Y donnée, l'algorithme teste chaque case de la colonne du bas vers le haut. Si une case est vide (valeur 0) alors la colonne n'est pas pleine, sinon elle est pleine.
3. **Vérification : La colonne donnée par l'utilisateur est-elle correcte ? :** En fonction d'un colonne Y donnée, si la colonne appartient à la grille (y compris entre 0 et 6) et si elle n'est pas pleine (voir algorithme 2) , alors la colonne donnée est correcte, sinon une nouvelle colonne doit être demandée à l'utilisateur.
4. **Vérification : Placement du pion ? :** En fonction d'une colonne Y donnée et d'un joueur, l'algorithme teste chaque case du bas vers le haut. Le pion dont la valeur correspond au joueur est placé dans la première case testée vide (valeur 0).[10]
5. **Vérification : victoire :** En fonction d'une case de coordonnée (X,Y) et d'un joueur, l'algorithme vérifie si la ligne X, la colonne Y, et les deux diagonales contiennent 4 cases successives non vides de même valeurs. Si c'est le cas, le joueur gagne et la partie est terminée.
6. **Vérification : match nul :** Si toutes les colonnes sont pleines (voir algorithme 2), alors le match est nul et la partie est finie. (Cet algorithme ne peut être exécuté qu'après l'algorithme 5).
7. **Vérification : Affichage :** L'algorithme parcourt toutes les cases et représente leurs états sur l'écran.

2.4 Spécifications

2.4.1 Version simple

Le langage de programmation que j'ai choisi afin de développer le jeu « Puissance 4 » est le C++. En effet le C++ possède tous les outils permettant de réaliser ce projet, de plus il est le principal langage informatique enseigné dans notre formation.

Dans la version simple, le jeu s'affiche dans un terminal. Le ou les joueurs doivent être présents pour réaliser la partie, qui ne peut se jouer qu'en une seule manche.

2.4.2 Extensions possibles

De nombreuses extensions sont possibles. L'utilisateur pourrait par exemple, avoir le choix du nombre de partie dans une manche. La possibilité de charger et de sauvegarder une partie en cours serait un plus.

Le joueur pourrait avoir la liberté de choisir son adversaire : intelligence artificielle ou adversaire réel et de jouer en réseau. Le choix entre deux niveaux de difficultés contre l'intelligence artificielle (simple et complexe) permettrait à l'utilisateur de jouer en fonction de son niveau.
La possibilité d'enregistrer le score par utilisateur et en fonction du niveau permettrait aux joueurs d'apparaitre dans un classement

L'affichage du jeu pourrait s'effectuer dans une fenêtre et ainsi permettre de rendre le jeu plus visuel et agréable à utiliser. Le score de chaque joueur pourrait être afficher à chaque instant.
La possibilité de choisir la forme et la couleur de ses pions rendrait le jeu plus ludique.
Le jeu pourrait aussi proposer une interface adaptée aux appareils tactiles.
Enfin l'optimisation des algorithmes permettrait d'accélérer les calculs et ainsi d'augmenter la vitesse d'exécution du jeu.

2.5 Organisation

Voici les différentes étapes du développement du projet pour une durée de 40 heures.

1. **Programmation de la version simple du jeu :** réalisation de la version simple du jeu dans le langage choisi. (10 heures) ;
2. **Programmation des extensions :** programmation des différentes extensions. (12 heures) ;
3. **Réalisation de l'interface :** réalisation de l'interface. (8 heures) ;
4. **Tests :** vérification du bon fonctionnement du logiciel. (33 heures) ;
5. **Rédaction du rapport :** (40 heures) 2.2

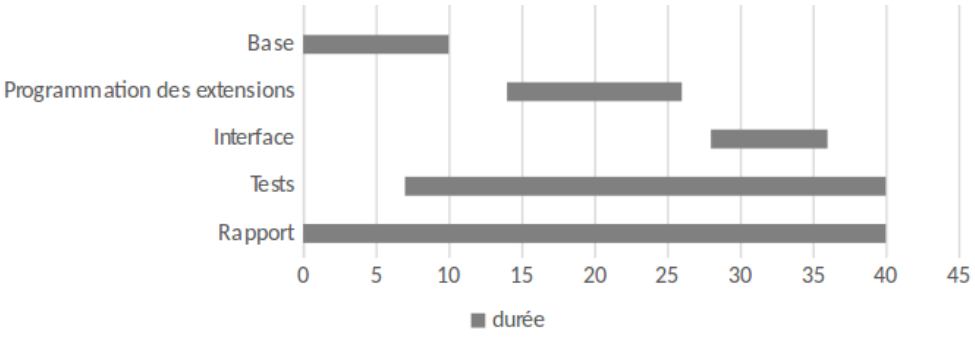

FIGURE 2.2 – Dévelopement du projet Puissance 4 sur 40 heures

Jeu 3

Jeu de plateformes

Dans cette troisième partie, je vais vous présenter un jeu qui n'est pas la reproduction d'un existant. Toutefois, il reprend les principaux mécanismes de trois grandes familles de jeux vidéo : plateforme, réflexion et coopération.[11]

3.1 Principe

Le jeu se joue à deux. Les joueurs doivent terminer un niveau dans un temps limité pour accéder au suivant et ainsi de suite. Le but étant de terminer tous les niveaux (`cf.` section 3.1.2).

Pour cela les deux joueurs (`cf.` section 3.1.1) doivent s'entraider en exploitant différents mécanismes de physique (`cf.` section 3.3.1).

3.1.1 Joueurs

Dans chaque niveau deux joueurs font évoluer chacun une balle. Celle-ci possède quatre propriétés, une position dans le plan selon des coordonnées (x,y), une charge positive ou négative, une vitesse et une masse.

La balle peut bouger de deux façons. Le joueur peut la déplacer vers la gauche, vers la droite et il a la possibilité d'inverser sa charge.
La balle subit aussi l'intervention des quatre forces physiques suivantes :

La force de gravité : Elle attire en permanence les balles vers la droite, le haut, la gauche ou le bas de la fenêtre. Le sens de la gravité peut être modifié durant le jeu selon les actions des joueurs.

La force de réaction : Elle permet à la balle de ne pas traverser certains blocs.

La force de frottement : Elle réduit la vitesse des balles. Cette force varie selon les blocs en contact avec la balle.

La force d'interaction coulombienne : Deux éléments de même charge se repoussent alors que deux éléments de charges opposées s'attirent.[12]

3.1.2 Niveaux

Un niveau est un ensemble de blocs statiques prédéfinis par le programmeur. Chaque bloc possède les quatre propriétés suivantes :

— une densité, un bloc est traversable ou non par les joueurs ;
— une charge, un bloc peut avoir une charge positive, négative, ou nulle ;
— une action, un bloc peut modifier le sens de la gravité vers le haut, le bas, la droite, la gauche ou bien ne réaliser aucune action ;
— une adhérence, un bloc peut plus ou moins réduire la vitesse de la balle. L'adhérence est propre à chaque bloc.

Il existe quatre types de blocs, les blocs vides, les blocs neutres, les blocs aimants et les blocs de gravité.
Les blocs vides sont les seuls traversables, ils possèdent une charge nulle et ne réalisent aucune action.
Les blocs neutres ont une charge nulle et ne réalisent aucune action.
Les blocs aimants possèdent une charge positive ou bien négative et ne réalisent aucune action.
Enfin, les blocs de gravité ont une charge nulle et réalisent une action.

Dans chaque niveau une zone d'arrivée est définie. Pour achever un niveau, la balle de chaque joueur doit se trouver dans cette zone pour accéder au suivant.

3.2 Modélisation

On modélise les objets du jeu par trois classes : balle, bloc, moteur.

La classe balle possède les `propriétés suivantes : position, vitesse, charge et masse.`
`position` est un vecteur représentant la position de la balle dans le plan.
`vitesse` est également un vecteur, il modélise la vitesse de la balle.
`charge` est un nombre qui ne peut prendre que deux valeurs, il représente la charge de la balle qui est soit positive soit négative.
`masse` est un nombre qui représente le poids de la balle.

La classe bloc possède les propriétés `position, traversable, charge, action, adhérence.`
`position` est un vecteur représentant la position du bloc sur la grille.
`traversable` est un booléen, si traversable est égal à `true,` alors le bloc est traversable par la balle, sinon non.
`charge` est une variable de type entier qui ne peut prendre que trois valeurs, il représente la charge de la balle qui est soit positive, négative ou nulle.
`Action` est également une variable de type entier, elle peut prendre quatre valeurs, 0,1,2,3 qui peuvent respectivement représenter le sens de la gravité (bas, droite, gauche, haut).

La classe `moteur` est la principale et possède les propriétés `ball, block et horloge.` Elle coordonne les éléments du jeu et organise le dessins des frames.
`Ball` est un tableau a une seule dimension qui stocke les balles présentes dans le niveau.
`block` est un tableau a deux dimensions qui représente la grille des blocs.
`horloge` mesure le temps entre chaque frame.

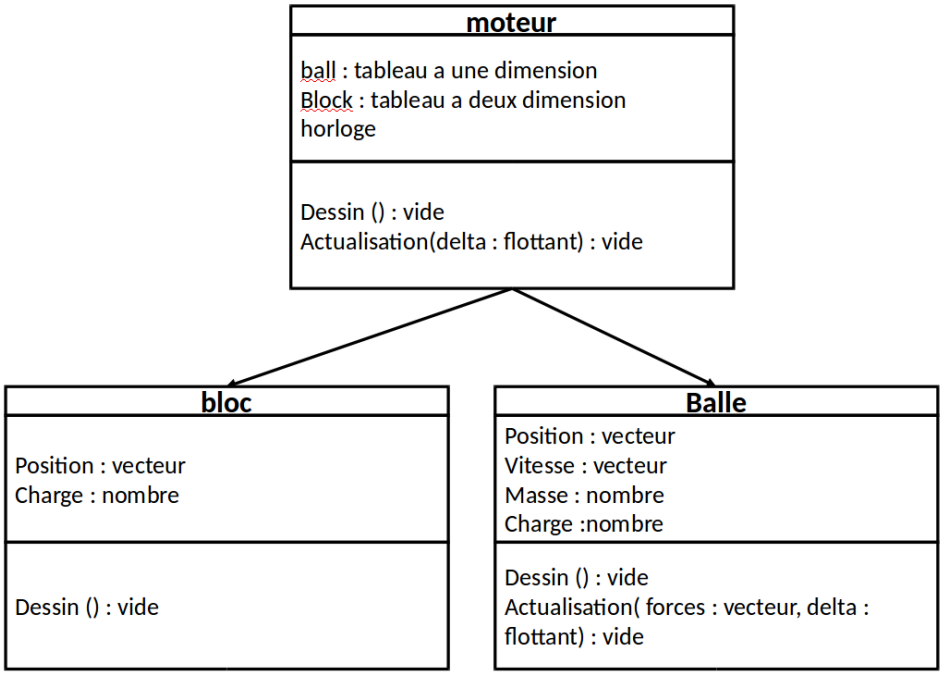

FIGURE 3.1 – Diagramme des classes

3.3 Algorithmes

3.3.1 Physique

L'algorithme `Moteur::actualisation (delta)` calcule en fonction du temps écoulé depuis la dernière `frame` un vecteur `force.` Ce vecteur est égal à la somme des forces appliquées à chaque balle et l'algorithme se déroule en 7 étapes :

1. Pour chaque balle dans ball
   (a) Initialisation d'un vecteur `force` égal au vecteur nul
   (b) Ajout du vecteur `(0,g), (0,-g), (g,0), (-g,0)` au vecteur force selon le sens de la gravité (`g` est une constante de gravité)
   (c) Si le joueur déplace sa balle vers la droite alors,
      i. Ajouter `(0,m)` au vecteur force. `m` représente la constante de mouvement
   (d) S'il la déplace vers la gauche alors,
      i. Ajouter `(m,0)` au vecteur force.
   (e) Pour chaque élément polarisé
      i. Ajouter à `force` un vecteur qui se situe sur une droite qui passe par le centre de la balle et celui de l'élément polarisé. Sa norme est égale
      (c((charge de la balle * charge de Objet polarisé) / (distance entre le centre de la balle et de l'objet polarisé)))* (`c` est une constante)
      ii. Opération sur le vecteur `force` selon les mouvements de la balle (les algorithmes de collision ne sont pas encore définis et nécessitent des recherches complémentaires).
      iii. Appeler l'algorithme `balle::actualisation(force, delta)`

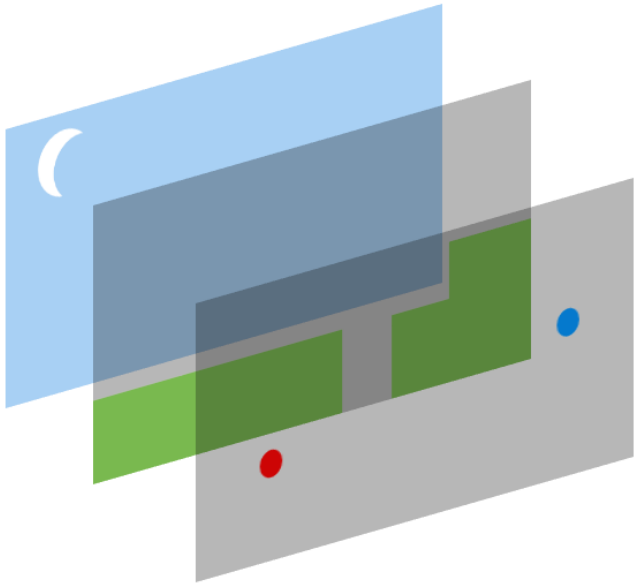

FIGURE 3.2 – Vue des différentes couches du rendu du jeu

L'algorithme `balle::actualisation(force, delta)` calcule en fonction de `force` et `delta` la position de la balle. (`delta` est le temps écoulé depuis la dernière frame). Cette algorithme se déroule en trois étapes :

1. `Accélération` est définie comme étant égal à la `force` divisée par `masse` (`masse` est une propriété de la balle)
2. `Vitesse` est égale à `vitesse + accélération * delta`
3. `Position` est égal à `position + vitesse * delta`

3.3.2 Affichage

Tous les objets dessinés à l'écran possèdent une fonction dessin.

L'algorithme `bloc::dessin()` est chargée de dessiner un bloc à ses coordonnées.

L'algorithme `Balle::dessin()` est chargée de dessiner une balle à ses coordonnées.

L'affichage du jeu est réalisé par superposition de trois couches de dessin 3.2. Sur la première couche l'arrière-plan du jeu, sur la seconde les blocs constituant les niveaux, et sur la dernière couche les balles.

La fonction moteur : :dessin() est chargée d'appeler les différentes fonctions de dessin dans un ordre précis afin d'afficher correctement le jeu à l'écran. Cet algorithme s'exécute en trois étapes :

1. Appeler l'algorithme qui dessine l'arrière-plan
2. Appelle l'algorithme `bloc::dessin` pour chaque bloc visible à l'écran.
3. Appelle l'algorithme `balle::dessin` pour chaque balle visible à l'écran.

3.3.3 Moteur

Initialisation du moteur
1. Création et initialisation de la fenêtre d'affichage du jeu ;
2. Initialisation d'horloge a 0 ;
3. Tant que la fenêtre est ouverte :
   (a) Gère la fermeture et la modification de la taille de la fenêtre ;
   (b) Calcule `delta` et réinitialise `horloge` a 0 ;
   (c) Appelle `moteur::actualisation(delta)` ;
   (d) Appelle l'algorithme `moteur::dessin()` ;

3.4 Spécifications

3.4.1 Langage

Le langage de programmation choisi pour développer ce jeu est le C++, car il possède tous les outils permettant de réaliser ce projet. De plus, c'est le langage informatique enseigné dans notre formation.
Afin de réaliser l'interface du jeu, le choix s'est porté sur la librairie SFML qui possède une interface de programmation simple et puissante et correspond parfaitement aux besoins.

3.4.2 difficultés

Les principales difficultés pour ce jeu sont la gestion des collisions entre les balles et les blocs. Les algorithmes de collision nécessitent des recherches complémentaires.[13] [14]

3.4.3 Version simple

Dans la version simple du jeu les deux joueurs jouent sur le même ordinateur et partagent le même clavier.
L'algorithme de physique utilise la méthode d'Euler. Cette méthode est simple et efficace, mais elle n'est pas très précise. Le comportement du jeu peut donc être différent selon la puissance de l'ordinateur utilisé ; plus le temps entre deux frames sera long, plus l'erreur sera élevée.
Toutefois, dans le cadre de ce jeu l'erreur devrait être minime.[15]

3.4.4 Extensions possibles

De nombreuses extensions sont possibles. Chaque joueur pourrait se connecter en réseau et ainsi utiliser un ordinateur différent, mais cela impliquerait la création d'un protocole de communication et une synchronisation de la simulation de physique.
L'algorithme de physique pourrait utiliser une méthode d'intégration plus fiable que la méthode d'Euler, comme par exemple l'intégration de Varlet ou encore la méthode de Runge-Kutta.[16] [17]

L'utilisateur pourrait avoir la liberté de personnaliser le jeu en modifiant les textures du jeu. Afin de cibler un maximum de joueurs, il serait intéressant d'optimiser la création du jeu pour les interfaces tactiles et de le porter sur les principaux systèmes d'exploitations mobiles et bureaux. Afin d'accélérer les calculs et ainsi d'augmenter la vitesse d'exécution du jeu, les algorithmes pourraient être améliorés.

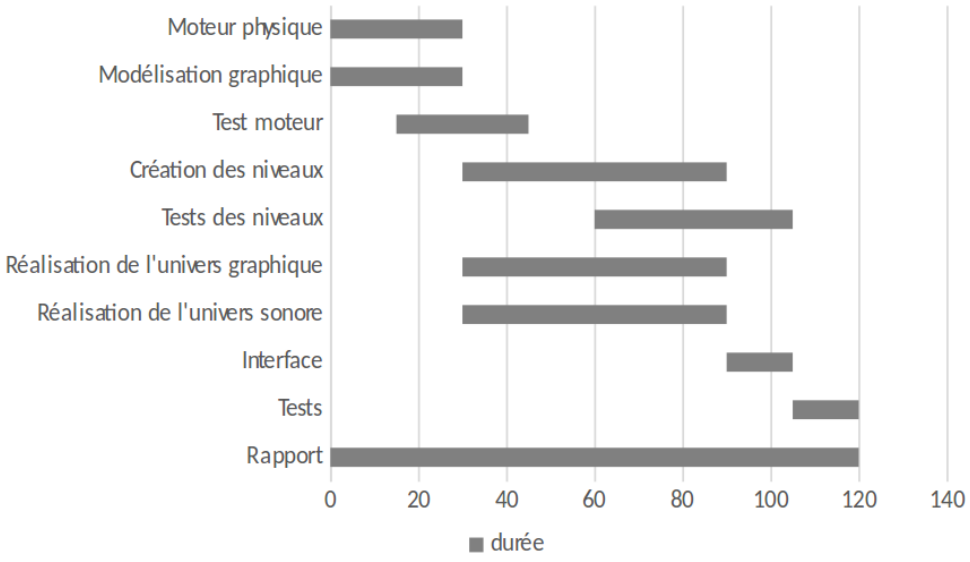

FIGURE 3.3 – Dévelopement du projet de plateforme sur 120 heures

3.5 Organisation

Les étapes du développement de ce projet peuvent se découper en différentes parties :

En premier lieu, les bases du programme doivent être mises en place, suivi par le développement du moteur physique en incluant la gestion des collisions.

En parallèle nous développerons une modélisation graphique de base ainsi qu'un niveau de test. Cela permettra la vérification et l'amélioration du moteur physique, si cela s'avère nécessaire.

Parallèlement à la conception des niveaux, nous réaliserons l'univers graphique (la mise en place des décors et des textures) et l'univers sonore (musiques, bruitages).

Les tests finaux sont prévus sur une durée de 5 heures. Le rapport final sera rédigé au fur et à mesure du développement du jeu. 3.3

Comparaison

Le jeu de la vie, puissance 4 et le jeu de plateforme présentent de nombreuses similitudes et différences. Tout d'abord, ces trois jeux abordent des principes différents ; le jeu de la vie est un automate cellulaire, puissance 4 est un jeu de stratégie et le dernier est un jeu de plateforme, réflexion et de coopération.

Le nombres de joueurs varient en fonction des jeux ; le jeu de la vie nécessite la présence d'une seule personne, celui de puissance 4 peut se jouer avec un ou deux joueurs contrairement à celui de plateforme où il est indispensable d'être deux.

Dans le jeu de la vie, le joueur choisit uniquement l'état initial puis reste inactif alors que dans le jeu puissance 4 et celui de plateforme les joueurs sont actifs en permanence.
Il est important de noter que seul puissance 4 est une simulation d'un jeu existant.

La modélisation est également différente selon les jeux. Le jeu de la vie et celui de puissance 4 sont modélisés par l'intermédiaire de tableaux comportant des cellules pouvant prendre différents états. Le jeu de plateforme est plus complexe car il utilise des classes et des objets, ce qui demande des algorithmes plus ardus.

Il existe deux sortes d'affichage pour ces jeux. Le jeu de la vie et puissance 4 utilise un affichage simple et statique contrairement au jeu plateforme qui est dynamique et qui utilise la superposition de différentes couches.

Les deux premiers jeux peuvent être développés en 40 heures avec l'intervention d'une seule personne. Le troisième jeu demande au minimum 120 heures et la collaboration de trois personnes pour sa réalisation. Le langage C++ est utilisé pour les trois jeux.
Des extensions sont possibles pour tous, toutefois le jeu de plateforme offre de plus grandes possibilités.

conclusion

Après avoir effectué la comparaison de ces trois jeux, j'ai une attirance particulière pour le jeu de plateforme qui allie le développement d'un moteur physique et la créativité .
Toutefois, sa réalisation demande un volume horaire important et une équipe de trois personne, qui serait idéalement composée de Maëlle Beuret et Mattéo Delabre.

Webographie

[1] Wikipédia. jeu de la vie. https://fr.wikipedia.org/wiki/Jeu_de_la_vie.

[2] C. Darrigan. Projet informatique en visual basic : Jeu de la vie. http://darrigan.net/blog/jeu-de-la-vie-programmer/.

[3] Antoine Cornuéjols Laurent Orseau. Automates cellulaires : Jeu de la vie. https://www.lri.fr/~antoine/Courses/AGRO/UV-Athens-Info-Vie/game-of-life.pdf.

[4] Wikipédia. Turing-complet. hhttps://fr.wikipedia.org/wiki/Turing-complet.

[5] openclassrooms. Tp corrigé jeu de la vie python. https://openclassrooms.com/courses/langage-python/tp-2-jeu-de-la-vie.

[6] Guillaume Vernade. Le jeu de la vie. http://www.normalesup.org/~vernade/td7.pdf.

[7] Wikipédia. Puissance 4. https://fr.wikipedia.org/wiki/Puissance_4.

[8] Wikipédia. Jeu de stratégie combinatoire abstrait. https://fr.wikipedia.org/wiki/Jeu_de_stratégie_combinatoire_abstrait.

[9] Mathieu Buard. Puissance 4. http://mathieu.buard.free.fr/jeux/ludotheque/puissance4.html.

[10] openclassrooms. tp projet puissance 4. https://openclassrooms.com/forum/sujet/aide-tp-projet-puissance-4-78469.

[11] Wikipédia. Type de jeu vidéo. https://fr.wikipedia.org/wiki/Type_de_jeu_vid%C3%A9o.

[12] Wikipédia. Loi de coulomb (électrostatique). https://fr.wikipedia.org/wiki/Loi_de_Coulomb_(%C3%A9lectrostatique).

[13] StackExchange GameAlchemist. Collision between AABB and circle. http://goo.gl/7E84Ef.

[14] Randy Gaul. How to create a custom 2d physics engine : The basics and impulse resolution. http://goo.gl/G0gdWU.

[15] Wikipédia. Méthode d'euler. https://fr.wikipedia.org/wiki/M%C3%A9thode_d%27Euler.

[16] Wikipédia. Méthode de verlet. https://fr.wikipedia.org/wiki/Int%C3%A9gration_de_Verlet.

[17] Wikipédia. Méthode de runge-kutta. https://fr.wikipedia.org/wiki/M%C3%A9thodes_de_Runge-Kutta.