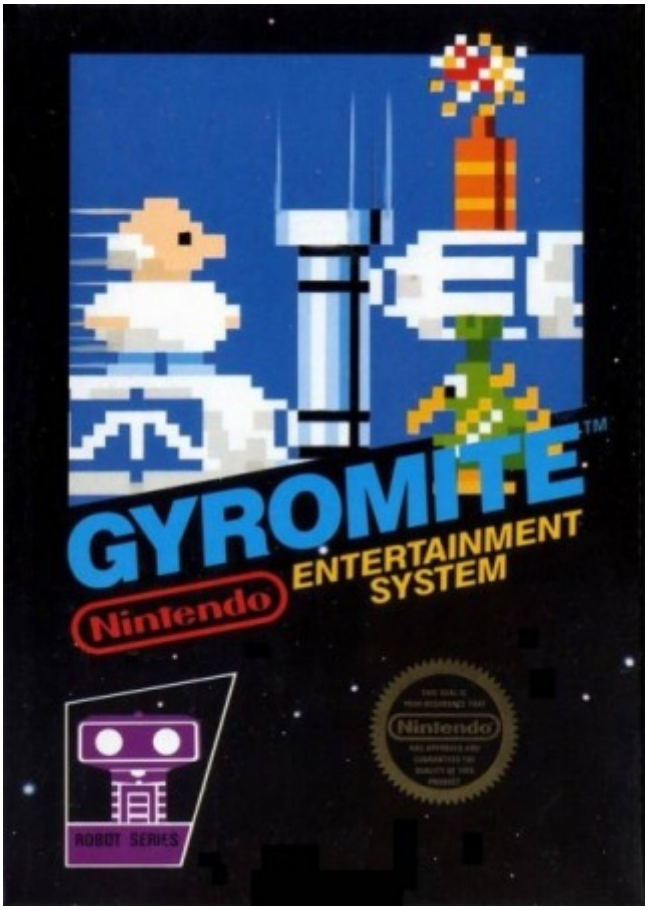
Projet - POO en Java
Gyromite

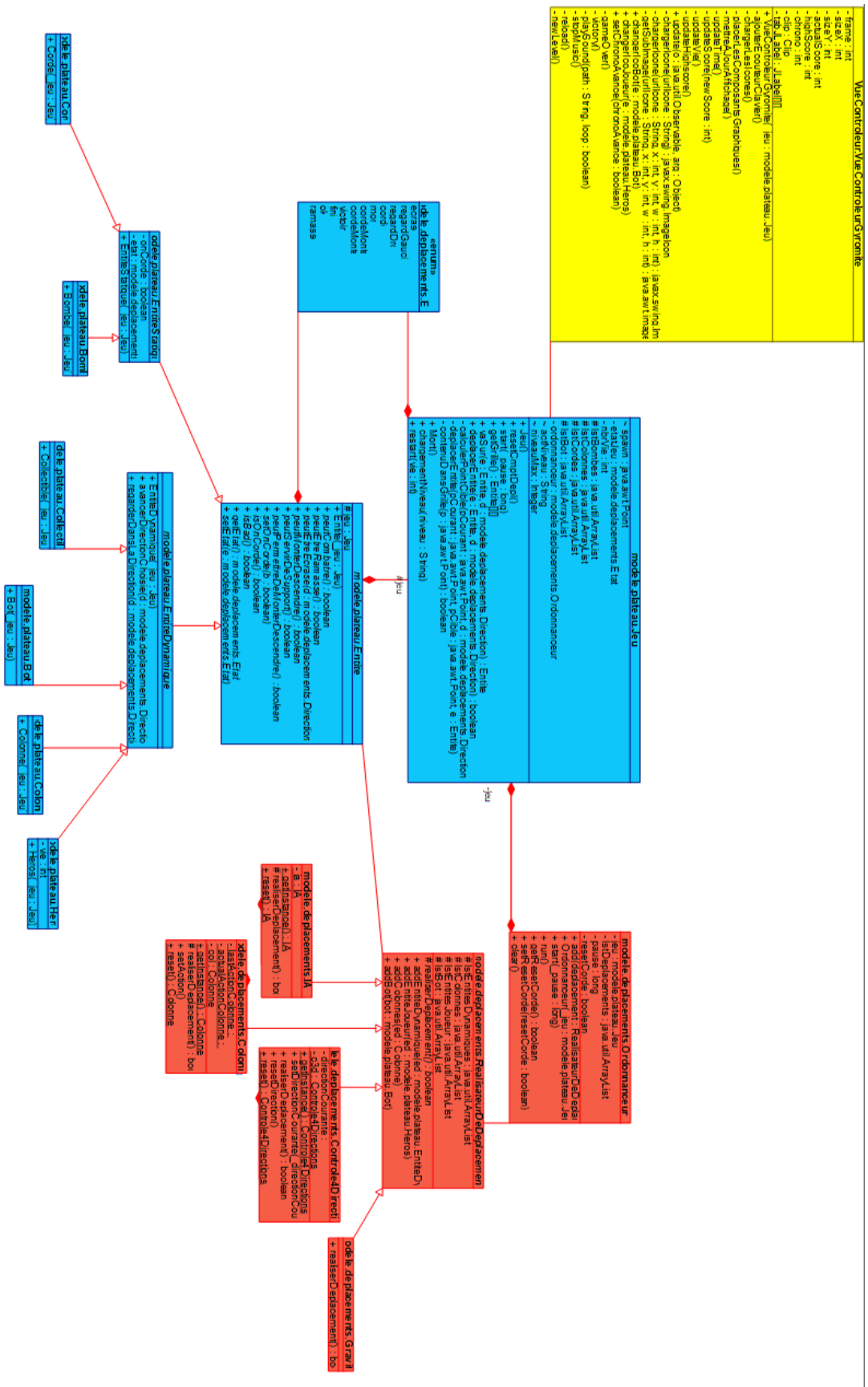


3ème année d'Informatique, Université Lyon1

Étudiants :

Thomas AUBOURG
Yanis AUMASSON

Diagramme de classe:



Listes des fonctionnalités et extensions:

- La gravité:
    - Les entités dynamiques étant soumises à la gravité, celles-ci tombent s'il n'y a rien sous elles

- Les collisions
    - Lorsqu'un prof rentre dans une entité pouvant servir de support (Mur, colonne, …) il ne se passe rien, cependant si une entité pouvant servir de support va sur un prof, c'est-à-dire que c'est elle qui effectue le déplacement, si le professeur peut être écrasé (que si la colonne déplaçait le prof d'une case dans sa direction courante, celui-ci serait écrasé par un autre support) celui-ci meurt d'écrasement.
    - Lorsqu'un prof rencontre un Smick ou qu'un Smick rencontre un prof, le prof meurt.
    - Lorsque deux Smicks se croisent il ne se passe rien, ceux-ci se traversent

- Les cordes
    - Lorsqu'un prof va sur une corde, celle-ci lui permet d'effectuer plusieurs sauts à la suite donnant l'impression de grimper. Il est cependant toujours affecté par la gravité et donc descend s'il ne bouge plus.

- Les bombes
    - Lorsqu'un prof va sur une bombe, il ramasse cette dernière. S'il a ramassé toutes les bombes il va au niveau suivant

- Les piliers
    - Lorsqu'une entité dynamique se trouve sur un pilier, celle-ci sera déplacée par ce pilier, celui-ci peut alors le monter ou le descendre voir l'écraser (voir gestion des collisions).
    - Pour déplacer un pilier, il suffit d'appuyer sur la touche MAJ du clavier. Le pilier se déplace de haut en bas jusqu'à atteindre un mur et suivant l'ancienne direction qu'il a effectuée (par défaut il commence par haut)

- Système de points
    - Lorsqu'un prof ramasse une bombe, ce dernier reçoit 100 points. A la fin on effectue un calcul qui multiplie le nombre de vies par l'addition du temps restant avec le nombre de points collectés pendant la partie.

- Gestion du chargement des niveaux
    - Il y a trois évènements qui peuvent amener à charger un niveau: Parce que le jeu commence, parce qu'on était dans un menu et qu'on a pressé la touche entrée ou parce qu'on a ramassé toutes les bombes d'un niveau.
    - Dans tous ces cas, on va simplement lancer le chargement des niveaux en indiquant quel niveau charger. Sachant qu'on a une variable qui enregistre à l'avance le nombre de niveaux et qu'on sait donc quand lancer le menu de victoire indiquant la fin du dernier niveau.

- La création des niveaux
    - La création du niveau se fait au lancement de ce dernier. Pour ce faire, on enregistre dans des fichiers CSV les informations sur où placer quelles entités et dans la fonction de lancement on va indiquer quel fichier lire, et donc quel niveau créer.

- La barre d'information
    - En haut de l'écran le joueur peut retrouver une barre lui donnant des informations sur le nombre de vies qu'il lui reste, le temps qu'il lui reste, son score actuel et son meilleur score.

- La fin de jeu
    - La fin de jeu arrive lorsque, soit un joueur a fini le dernier niveau, dans quel cas un écran de victoire lui est affiché et lui propose de recommencer en appuyant sur la touche entrée, soit quand le joueur n'a plus de vie ou de temps, dans quel cas c'est un écran game over qui lui est affiché et qui lui propose de recommencer en appuyant sur la touche entrée.
    - Quand le jeu recommence, le joueur est replacé au niveau 1 et ses informations de parties son réinitialiser, à l'exception du meilleur score

- Gestion des sprites
    - Plusieurs sprites sont proposés pour le prof et les bots. Pour ces derniers on retrouve uniquement les Sprites de gauches et de droites activées suivant leur direction courante. Pour le prof il y en un peu plus. Ce dernier peut avoir en plus un sprite quand il est sur une corde (en fait il y en a 2 différents qui s'alternent pour donner l'impression qu'il grimpe mais ceux-ci étant trop similaire on ne voit pas la différence), ou quand il se fait écraser, ou encore quand il est tué par un bot.

- Gestion de la musique
    - Une musique est lancée en même temps que le premier niveau et celle-ci change quand, soit le joueur a gagné, dans quel cas une musique de victoire est lancée, soit quand il a perdu, dans quel cas c'est une musique de game over qui est lancée.

- Le High Score
    - A chaque fois que le joueur bat son meilleur score, le jeu va modifier un fichier qui enregistre le meilleur score et par la suite ce sera donc ce nouveau meilleur score qui sera affiché.

- Les Smicks
    - Ils se déplacent de gauche à droite et changent de direction lorsqu'ils sont face à un mur, au vide, à une colonne ou à une corde. De plus, ils disposent d'un facteur aléatoire leur permettant de changer de direction de manière imprévisible.

Tâches	Proportion de temps
La gravité	Intermédiaire
Les collisions	Long
Les cordes	Intermédiaire
Les bombes	Intermédiaire
Les piliers	Long
Système de points	Rapide
Gestion du chargement des niveaux + “Restart”	Long
Création des niveaux	Rapide
La barre d’information	Intermédiaire
La fin du jeu	Long
Gestion des sprites	Intermédiaire
Gestion de la musique	Intermédiaire
Le High Score	Intermédiaire
Les Smicks	Intermédiaire

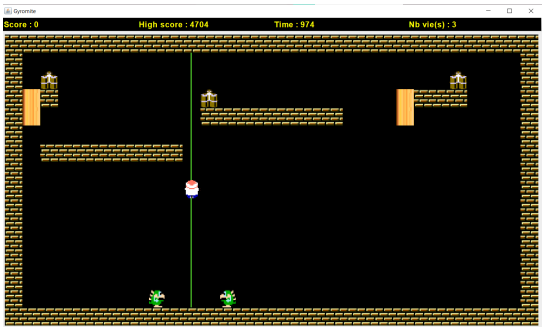
Justification de notre analyse :

Le modèle-vue-contrôleur (MVC) est un patron de conception logiciel qui sépare les données d'une application (le modèle), sa présentation (la vue) et la logique de contrôle (le contrôleur). Dans une implémentation standard du MVC, la vue est chargée de l'affichage des données pour l'utilisateur, tandis que le contrôleur gère les interactions avec l'utilisateur et met à jour le modèle en conséquence. Ici nous avons fait le choix de réunir la vue et le controller pour plus de simplicité. Nous avons décidé de créer un classe “Colonne” pour mieux gérer leurs déplacements, de même pour les bots qui possèdent eux aussi des règles de déplacement particulières.
De plus, nous avons essayé de respecter au maximum les règles de programmation orientée objet. Afin de garder un code plus robuste, plus lisible et plus modulaire, nous nous sommes limités sur l’utilisation de l’opérateur “instanceof”, nous avons seulement garder ceux présent dans le code de base.

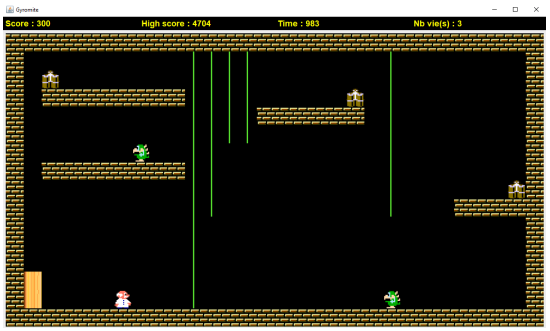
Déroulement d'une partie :

Voici les 2 niveaux présents dans le jeu :

Niveau 1

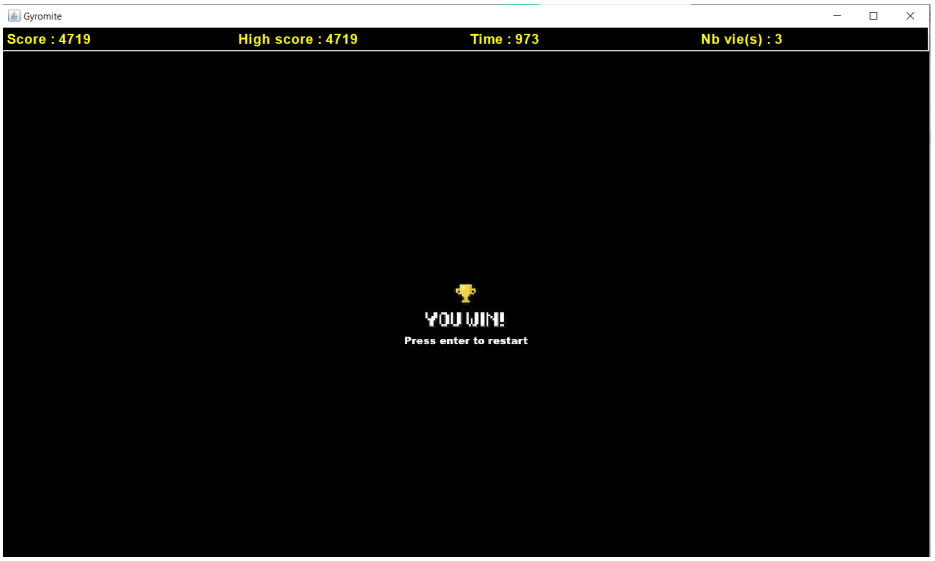


Niveau 2



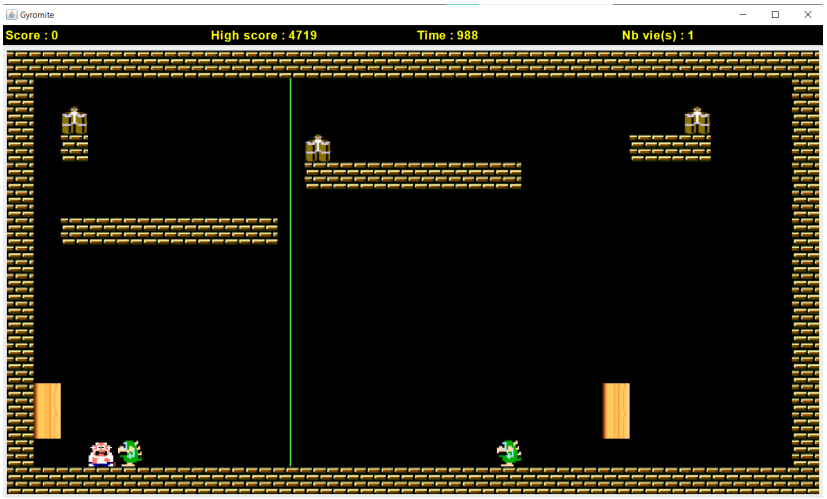
Lorsque le joueur lance le jeu, il se retrouve dans le niveau 1, il a alors pour objectif de récolter toutes les bombes présentes dans le niveau pour accéder au niveau suivant. Pour obtenir la victoire, il doit terminer l'ensemble des niveaux dans le temps imparti et sans que son compteur de vie n'atteigne "0". Pour se faire il peut s'aider des éléments de gameplay mis à sa disposition (Cordes, Colonnes…), il peut également avoir un aperçu de son score actuel, le meilleur score réalisé, le temps restant et pour finir son nombre de vies. Toutes ces informations s'actualisent en temps réel dans la barre située tout en haut de la fenêtre.

Si les conditions de victoire sont remplies l'écran de "Victoire" s'affiche et la musique correspondante est jouée. De plus, en cas de victoire, si le score actuel est supérieur au High Score précédemment sauvegardé alors celui-ci devient le nouveau High Score. Le joueur peut alors appuyer sur sa touche "Entrée" pour recommencer une partie sans redémarrer le jeu.



Écran de "Victoire"

Lors d'une partie le joueur peut être amené à croiser la route d'un Smick. Au contact de celui-ci le joueur meurt le forçant à recommencer depuis le début du niveau (il ne recommence pas au début du jeu). Il conserve son nombre de bombes ramassées mais il perd cependant une vie.



Le joueur rencontre un Smick

Si le joueur ne parvient pas à terminer le jeu dans le temps imparti ou que son compteur de vie atteint "0", l'écran de "Game Over" se lance accompagné de la musique correspondante. Bien sûr, son score ne sera pas comptabilisé et il pourra appuyer sur sa touche "Entrée" pour rejouer.

Écran de "Game Over"