Rapport Projet Gyromite - LIFAP7

Pierre DAMEZ-FONTAINE - Raphaël RICHARD

1) Liste des fonctionnalités

Requises :

- Gravité
- Gestion des collisions
- Capacité de monter/descendre des cordes
- Capacité de ramasser les objets
- Capacité de pouvoir déplacer les piliers
- Système de points
- Règles du jeu (gameplay)

Extension :

- Niveaux pouvant être sauvegardés et chargés dans des fichiers (serialisation)
- Possibilité de passer d'un niveau à l'autre
- Niveaux plus grands que l'écran (scrolling)

2) Déroulement du développement

Le développement du jeu s'est articulé autour de releases grâce à git (commits basés sur l'implémentation de fonctionnalités majeures), mais aussi de fix (petits commits visant à corriger quelques chose).

Voici comment les releases on été publiées :

- Release #1 : Textures de base et déplacement sur les cordes
- Release #2 : IA des bots + sprite du hero sur une corde
- Release #3 : Ajout des bombes (et ramassage par le héros)
- Release #4 : Ajout des colonnes (mouvements et textures)
- Release #5 : Sérialisation des niveaux
- Release #6 : Ajout des collisions héros/bots et colonnes/entités
- Release #7 : Ajout du gameplay (mort, passage au niveau suivant, fin du jeu)
- Release #8 : Ajout du scrolling (niveaux plus grand que l'écran)
- Release #9 : Ajout des scores et de la barre d'info (scores et temps)

3) Documentation UML

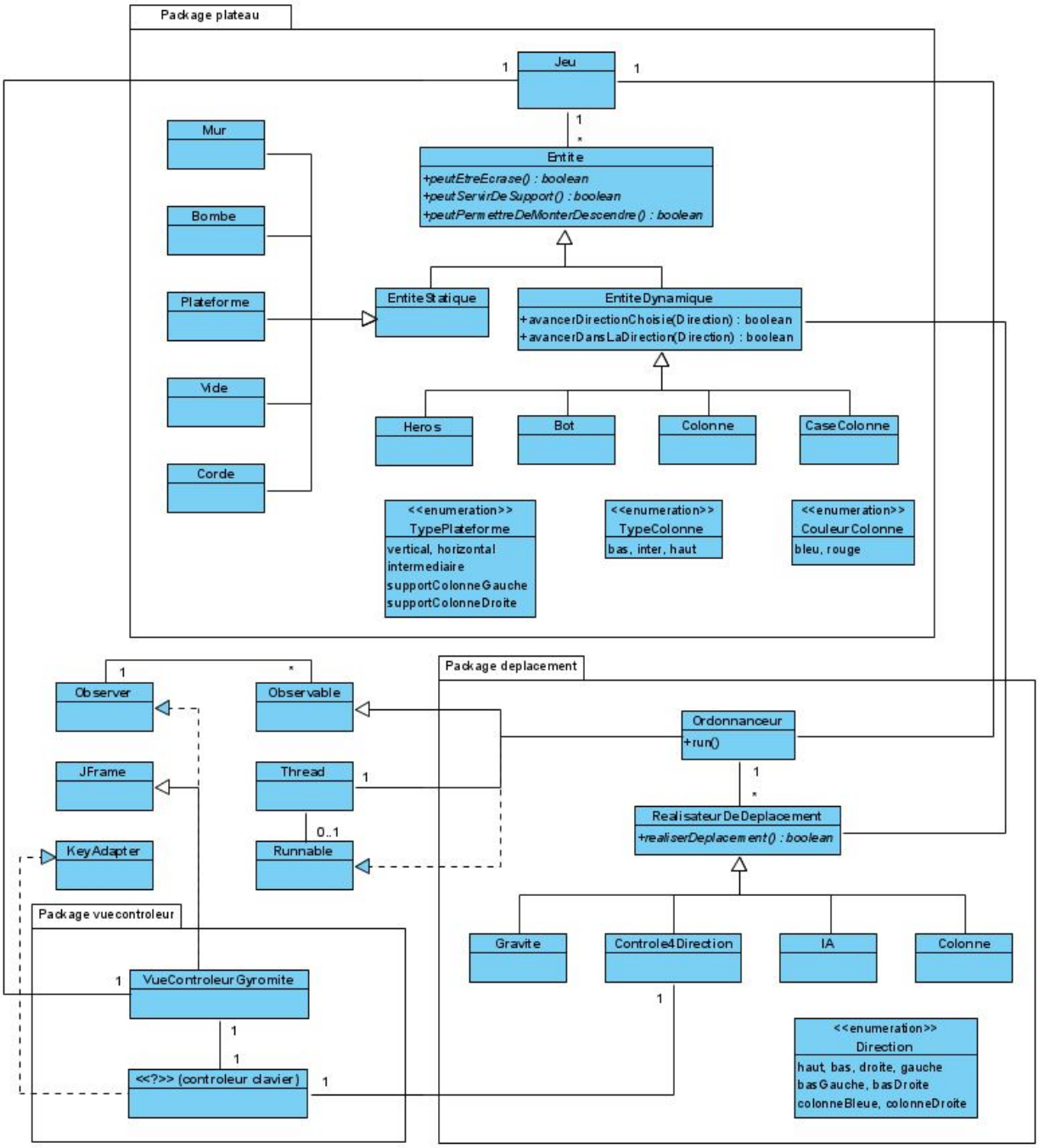


Diagramme de classe du projet Gyromite

4) Choix d'implémentation

Contrairement à la version de base proposée, toutes les entités du plateaux correspondent à une classe (vide, plateforme, …). Cela permet de simplifier les vérifications et les conditions lors de la détection des collisions (car faire un if sur une référence nulle provoque une exception).

Mais cela permet aussi d'introduire de la diversité dans les textures. Par exemple, une plate-forme peut avoir plusieurs version (verticale, horizontale, support de colonne, etc).

Les interactions des entités avec leur environnement et les autres entités sont gérées dans leur realisateurDeDeplacement respectif (au lieu de la fonction deplacerEntite dans la classe jeu). Cela permet d'avoir des interactions spécialisées à la source.

Les colonnes sont gérées grâce aux classes "CaseColonne" et "Colonne". Cette dernière contient des instances de CaseColonne mais permet de les regrouper et donc d'effecteur les vérifications liées à leur mouvement ou leur interactions avec les entités de manière bien plus simple (on peut se permettre de parcourir le tableau dans un sens ou dans l'autre en fonction de la direction des colonnes). Celles-ci possèdent également une couleur (rouge ou bleue) ainsi qu'une texture particulière (en fonction de leur placement).

La serialisation est assurée par la fonction initialisationDesEntites qui prend en paramètre le nom du fichier stockant le niveau. La fonction va ouvrir le fichier et récupérer un entier qui correspond à sa longueur horizontale. La prochaine étape est le parsing (prendre mot par mot les données du fichier pour ensuite les traiter). La plupart des symboles sont composés d'une seule lettre (comme un mur M, le vide V, etc), mais certaines entités nécessite plus comme les plateformes (PV pour une plateforme verticale, PH pour une horizontale, etc). Les entités dynamiques ont un fonctionnement plus complexe car elle nécessitent d'être ajoutées à plusieurs ordonnanceur afin de fonctionner correctement (comme gravité et IA pour le smicks). Après leur initialisation et éventuellement ajout aux ordonnanceurs, les entités sont ajoutées à la grille d'entité. Les colonnes ont un fonctionnement un peu plus complexe car il est nécessaire de récupérer plus de données (sur la couleur et le placement) ainsi que de constituer les Colonnes composées de CaseColonnes (assuré par le tableau tableColonne).

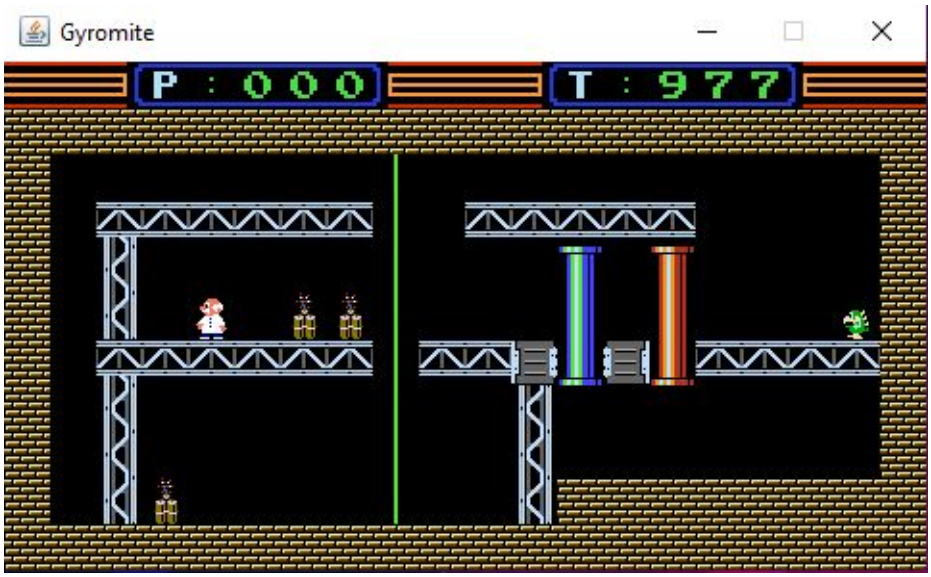
Une classe gameplay a été ajoutée pour gérer tout ce qui touche à l'avancement du niveau, le nombre de vies du joueur, le temps restant, le score ainsi que le passage au niveau suivant. A chaque mise à jour du jeu, la fonction "verifier" de la classe est appelée afin de faire les vérification nécessaires (décrémenter le compteur de temps, vérifier si le nombre de bombes actuel est inférieur au nombre de bombe total dans le niveau, etc.

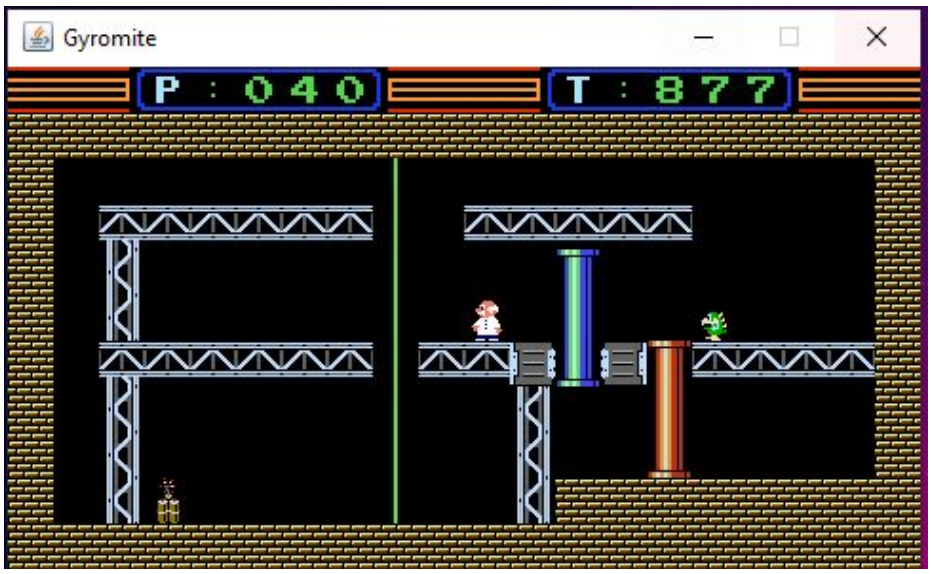
Le changement de niveau se fait aussi dans cette classe (quand toutes les bombes du niveau ont été récoltées par le joueur), c'est à partir de celle-ci que la grille des entités est vidée et que le niveau suivant est chargé (à partir de la liste de tous les niveaux stockée dans un autre fichier), tout ceci se faisant via des appels de fonctions de la classe jeu.

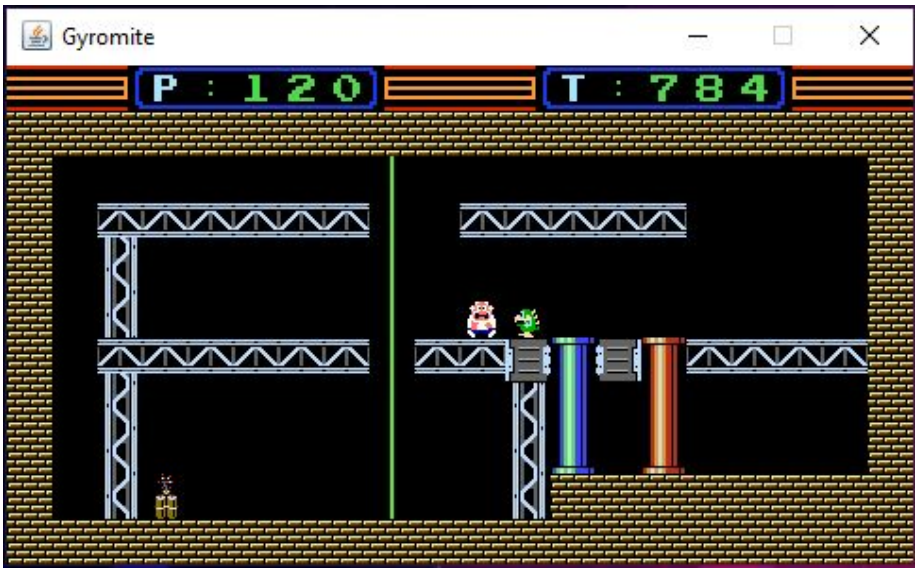
Une barre affichant le temps et le score actuel est présent sur le haut de la fenêtre. L'affichage de celle-ci est assuré par la fonction "affichageScore" dans VueControleur. La plupart de la barre est statique (seules ci cases sont modifiées en fonction des valeurs score et temps de la classe gameplay. Les valeurs sont décomposées en centaines, dizaines et unités puis affichées.

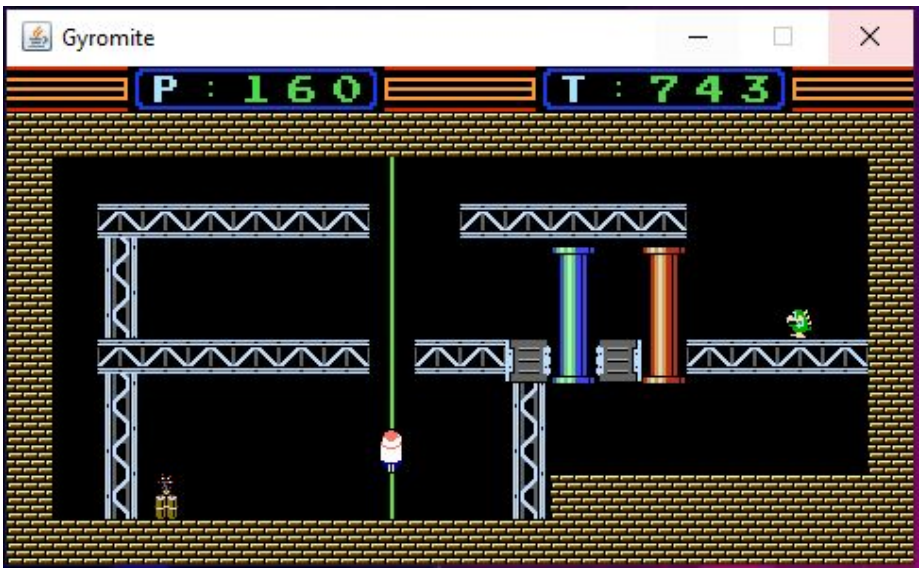
5) Captures d’écran

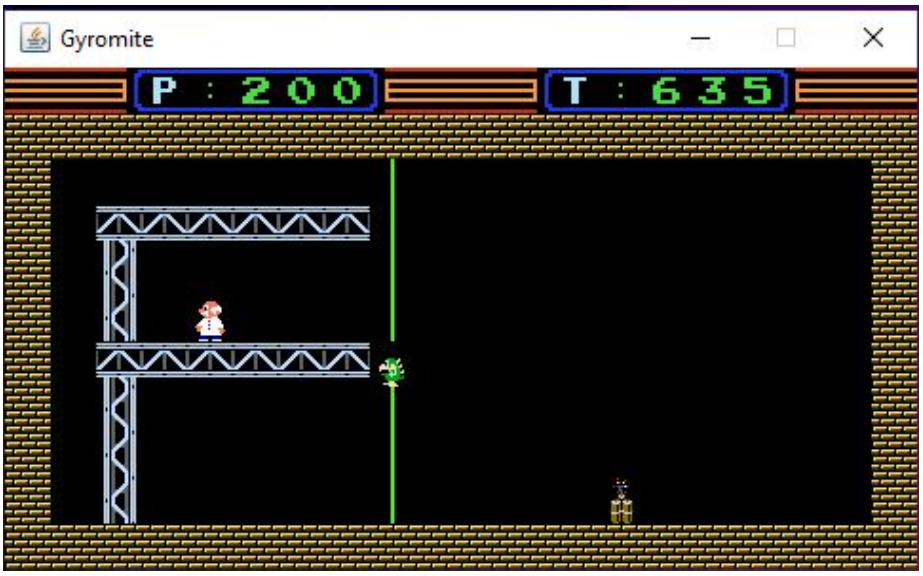
Ceci conclut le rapport sur le projet Gyromite du cours de LIFAP7. Voici des captures d’écran illustrant le jeu en fonctionnement :

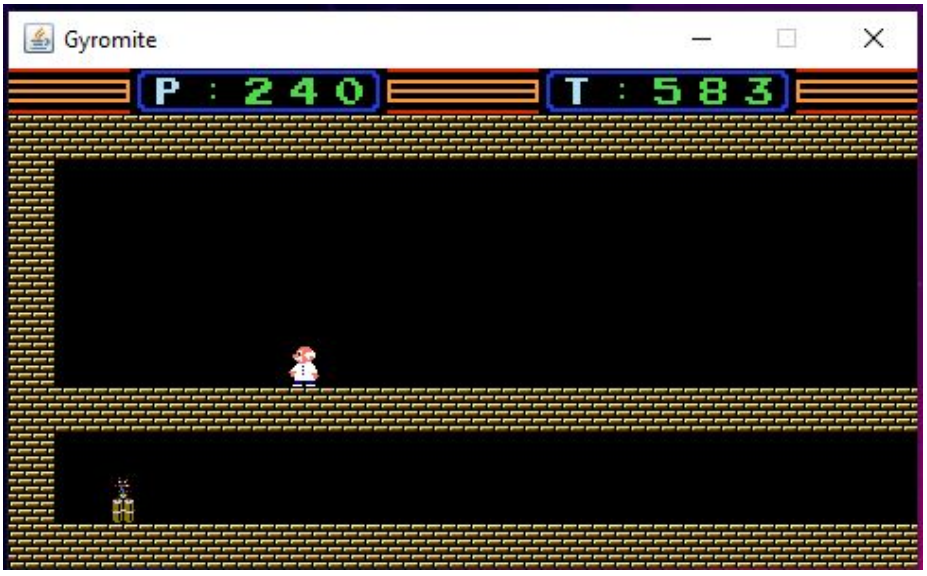


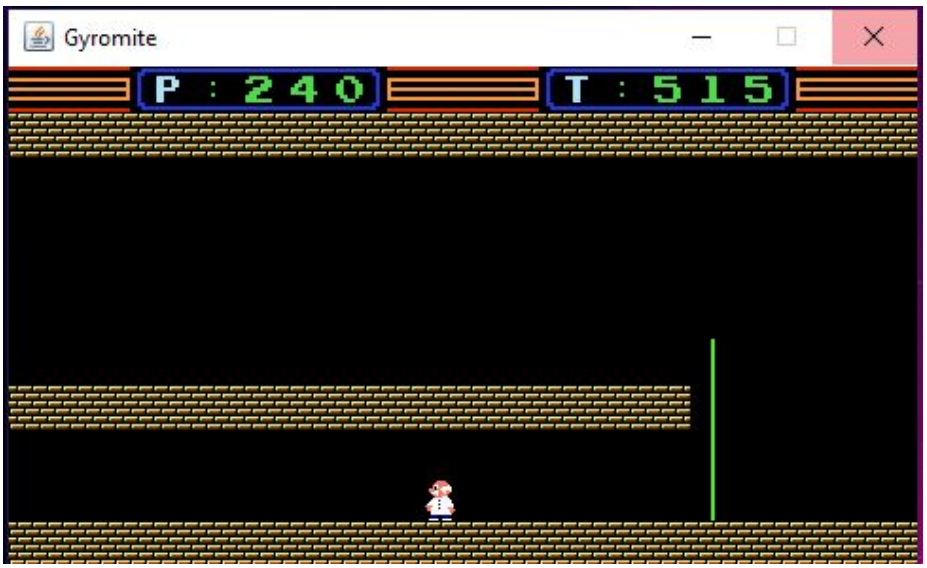


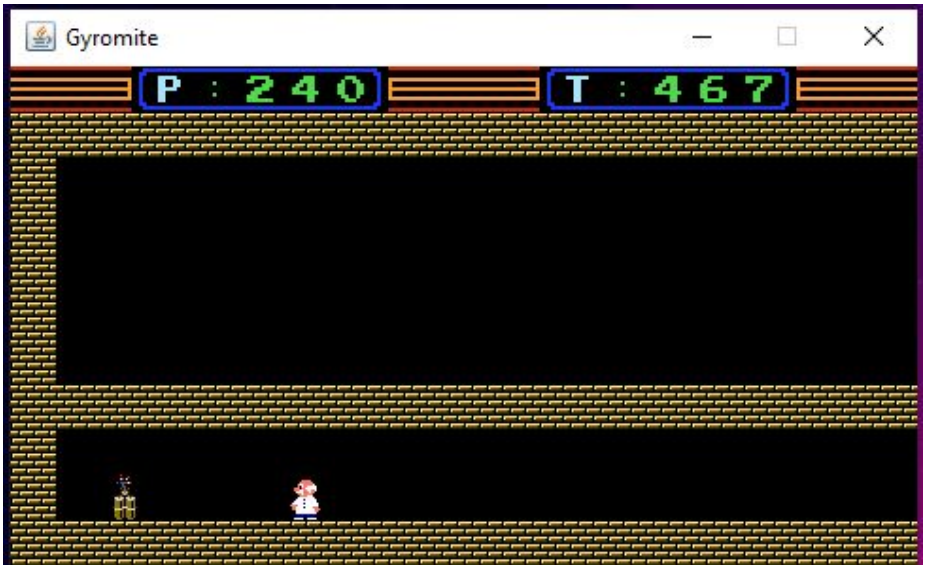



Gyromite
P : 160
T : 743


Gyromite
P : 200
T : 635


Gyromite
P : 240
T : 583






```
20
M M M M M M M M M M M M M M M M M M M M
M V V V V V V V C V V V V V V V V V V M
M V PI PH PH PH PH PH C V PH PH PH PH PH V V V V M
M V PV V V V V V C V V V CBH1 V CRH2 V V V V M
M V PV V H V B B C V V V CBI1 V CRI2 V V S V M
M V PI PH PH PH PH PH C PH PH PG CBB1 PD CRB2 PH PH PH PH M
M V PV V V V V V C V V PV V V V V V V V M
M V PV V V V V V C V V PV V V V V V V V M
M V PV B V V V V C V V PV M M M M M M M M M
M M M M M M M M M M M M M M M M M M M M
```