Test I-Runner

Participants :

- Charlotte Richard
- Julien Guittat
- Maxime Maubant
- Lucas Ouaniche--Herbin

Nom :

Test I-Runner

Objectif/fonctionnalités du projet :

Réaliser un jeu qui se jouerait à plusieurs, dans lequel il y aurait un défilement horizontal du terrain, et les joueurs peuvent avancer/reculer/changer leur gravité.
Le but étant de survivre le plus longtemps, le dernier en vie (qui n'a pas été sorti du terrain) gagne.
Chaque joueur aura donc accès à 3 touches du clavier.

Fonctionnalités définitives :

Le jeu possède un menu dans lequel on peut : regarder les options (flèche vers la droite), quitter ou lancer une partie. Ce menu possède un petit plus caché : en cliquant sur les petits persos qui courent en bas, on peut faire apparaître de nouveaux personnages !

Lorsqu'on clique sur Jouer, on change de menu, des petits boutons pour retourner au menu et pour quitter sont disponibles en haut de l'écran.

Un menu pour choisir le nombre de joueurs apparaît ensuite.

Puis un menu de sélection du personnage de chaque joueur : on doit sélectionner le joueur via le radiobutton qui correspond, puis cliquer sur le personnage qu'il veut. Quand tous les joueurs sont satisfaits de leur personnage, la flèche vers la droite permet de lancer la partie.

Pendant la partie, le but de chacun des joueurs est de tenir le plus longtemps sans sortir de l'écran. Pour cela chacun a 3 touches : une pour ralentir, une pour accélérer et une pour changer la gravité (sauter). Ces touches sont détaillées dans les options (A pour ralentir, Z sauter et E accélérer pour le premier joueur ; 7 pour ralentir, 8 sauter et 9 accélerér pour le deuxième...).

Lorsque tous les joueurs sont sortis de l'écran, la partie est terminée, un écran de scores s'affiche alors. A partir de cet écran, on peut retourner au menu, quitter ou relancer une autre partie avec les mêmes joueurs, via la flèche correspondante.

Planning :

- 25/11/15 ET 02/12/16 : Choix du jeu que l'on veut faire, travail préliminaire au brouillon sur les classes à créer et leur hiérarchie. Début du codage dans les grandes lignes.

- 9/12/15 : Mise en place des différents menus, répartition des classes à faire

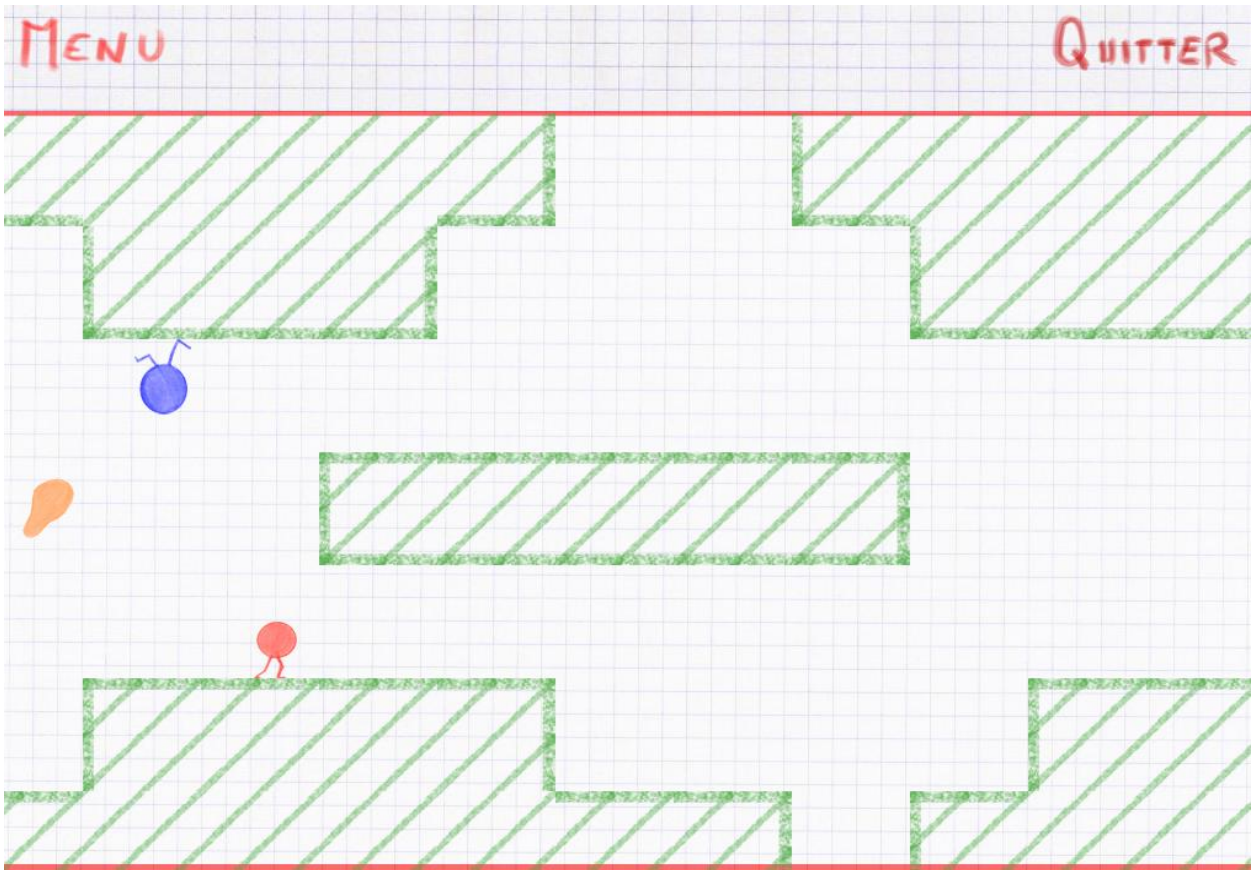
- 16/12/15 : Créer les différents Jpanels, sans la "physique" du jeu, travailler sur la génération aléatoire du terrain. Travail sur les affichages de persos et des persos retournés.

- vacances : Travail sur la "physique" du jeu : gérer les collisions, élimination si on sort... (les différents personnages peuvent se traverser). Conditions d'élimination et donc aussi de victoire. Mise en place de la navigation entre les menus. Création d'un écran de scores. Mise en place d'un menu de choix des personnages.

- 13/01/16 : Tests pour vérifier s'il n'y a pas de bug ou de problèmes pas encore détectés. Ajout d'un bouton pour relancer une nouvelle partie après l'affichage des scores.
- résolution de bugs, tests, et travail sur le document à rendre.

- 15/01/16 : Finition : fin de la rédaction du compte rendu, création du dossier que l'on rend.

Interface graphique :




Test I-Runner
Test T-Runner
JOUER
QUITTER
CLIC


Test I-Runner
MENU
QUITTER
Joueur 1
Joueur 2
Joueur 3
Joueur 4

Gestion des collisions :

Lorsqu'une collision est détectée à droite du personnage, il se fait emporter par le décor, et sa vitesse horizontale devient égale à la vitesse d'avance du terrain. Le personnage se retrouve ainsi déplacé vers la gauche par le terrain jusqu'à sortir s'il ne fait rien.

Lorsqu'un saut conduit à la collision du personnage avec le sol ou le plafond, il y a déclenchement de l'animation d'atterrissage, et la vitesse verticale du personnage redevient nulle. Il reprend sa course.

Lorsqu'il n'y a plus de contact entre le personnage et le sol (ou le plafond) mais pas de saut, le personnage chute dans le sens de la gravité jusqu'à la prochaine plate-forme, ou jusqu'à sortir de l'écran.

Le personnage peut sortir de l'écran, soit par la gauche si le joueur n'est pas allé assez vite, ou si il a été bloqué par une collision à droite, soit par le haut ou le bas, si il est tombé dans un trou, et n'a donc pas rencontré de surface pour réceptionner son saut afin de continuer sa course. Quand son personnage sort de l'écran, le joueur a perdu, la partie est terminée quand le dernier joueur perd.

Hiérarchie des classes :

- Main

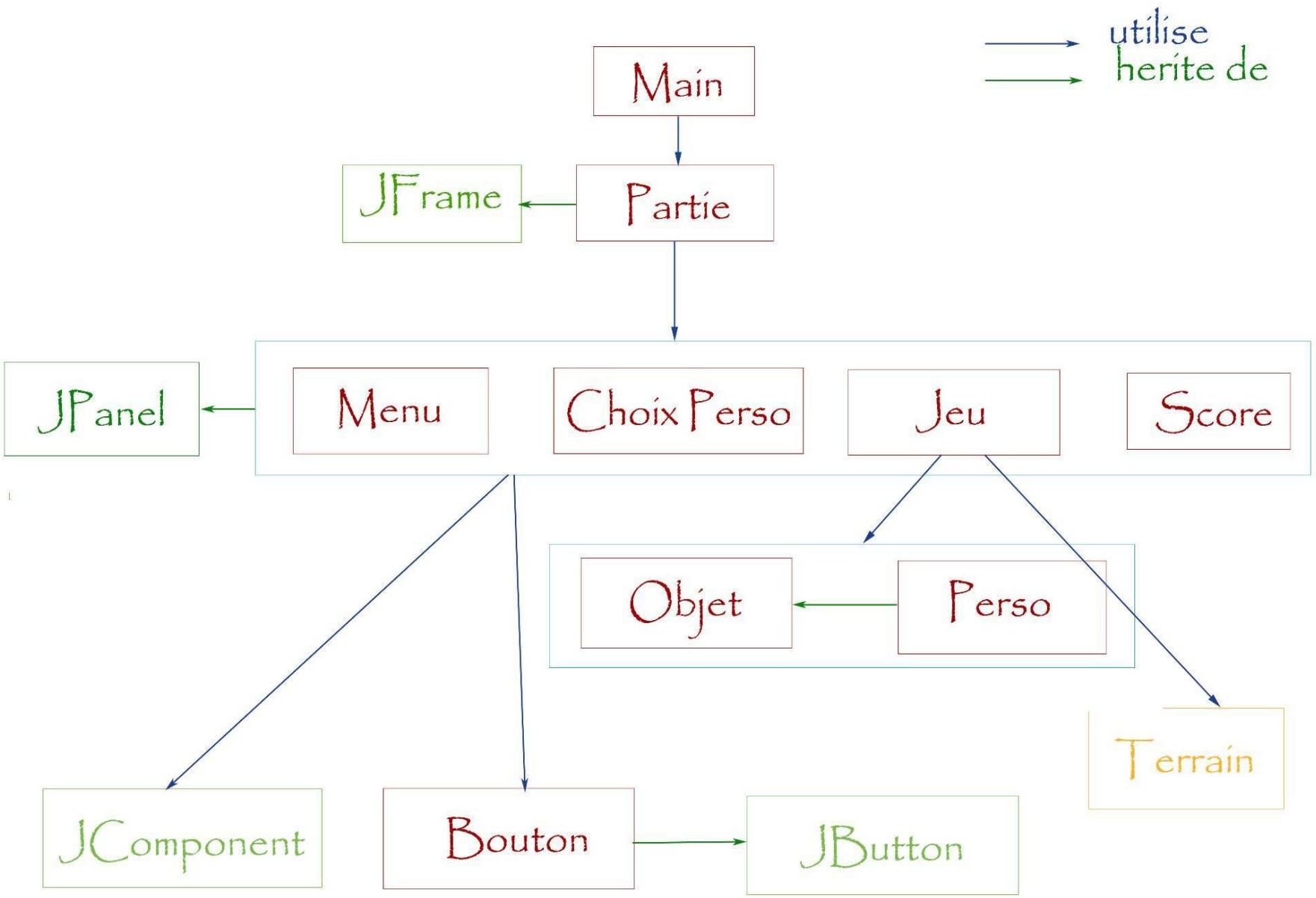
- Partie, héritant de Jframe, la classe dans laquelle on mettra les panels suivants :

- Menu héritant de Jpanel, qui aura l'affichage d'un fond de menu

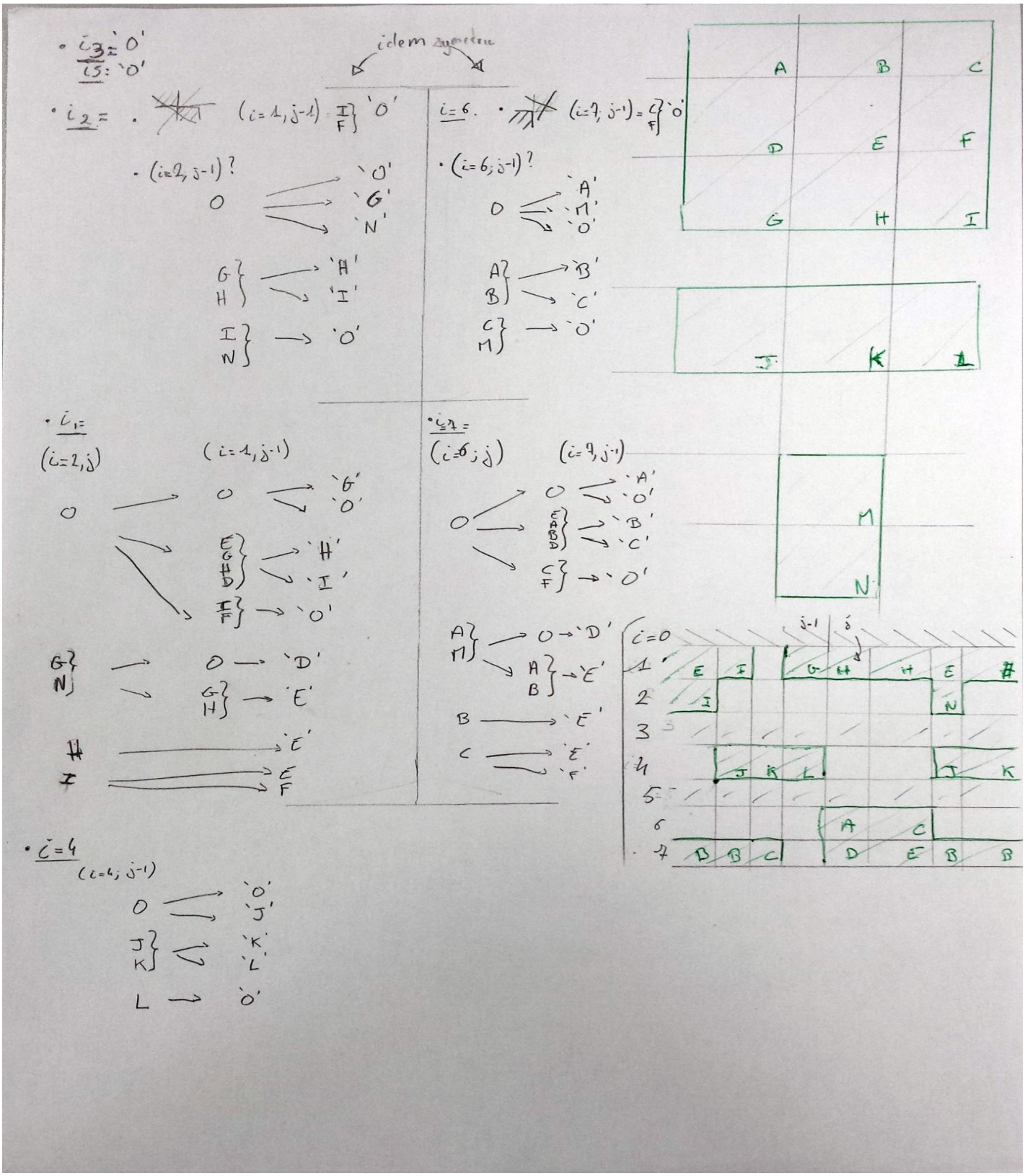
- Jeu et Scores héritant de Jpanel qui afficheraient respectivement le jeu-même puis les scores

- Terrain qui contient les algorithmes de génération automatique de terrains.

Dans ces classes on aura besoin d'éléments tels que :
- Perso, héritant de la classe Objet inspirée de celle donnée pour le TP météorites,
- Bouton, héritant de JButton pour personnaliser nos boutons.
(des classes abstraites pourront être utilisées pour les méthodes complexes telles que la génération de terrain, cela nous permet de travailler sur plusieurs fichiers différents).


utilise
herite de
Main
JFrame
Partie
JPanel
Menu
Choix Perso
Jeu
Score
Objet
Perso
Terrain
JComponent
Bouton
JButton

Schématisation de l'algorithme de la classe terrain :



=> problèmes rencontrés lors du codage **:**

Problème de hitbox, le personnage passait à travers les sols et plafonds :
exemple : si la vitesse est de 5px/frame, le perso se déplace de 5px par 5 px, il peut donc se retrouver à 1px du bord (pas de collision), puis à la frame d'après, il interpénètre le mur de 4px... Pour régler cela, on réduit les hitbox latérales d'une valeur devant être supérieur à la vitesse. Le bouton de saut faisait bien changer la gravité mais le sol et le plafond n'étaient que affichés, pas physiquement présent. On pouvait donc sortir le personnage de l'écran et y revenir.

L'augmentation de la vitesse du personnage le faisait rentrer dans le sol, provoquant ensuite des collisions latérales qui n'avaient pas lieu d'être, puisqu'il n'y avait pas de mur affiché vers les collisions.

La gestion des différents menus nous a aussi posé des problèmes sur lesquels on a du longuement se pencher, et organiser correctement notre code pour que cela puisse bien se passer : une fenêtre globale que l'on garde tout le temps et dont on remplace seulement son ContentPane par un JPanel adapté. Pour qu'un JPanel puisse se faire remplacer par un nouveau JPanel on a passé un pointeur vers la fenêtre en paramètre des classes héritant de JPanel, ces classes peuvent donc désormais modifier le contenu de la fenêtre.

On a aussi rencontré des problèmes avec le focus sur la fenêtre : lorsqu'on a des boutons, le focus reste sur ces boutons et le clavier n'est plus écouté par la fenêtre de jeu. On a dû donc utiliser un `requestFocusInWindow();` pour résoudre ces problèmes.

*=> **Limites et éventuels bugs ou parties du programme pouvant être améliorées :***

Notre programme est fini et marche. Il est possible de jouer à plusieurs sans bug majeur ou venant gêner les joueurs. Malgré cela, il reste quelques petites choses que nous pourrions améliorer avec plus de temps.

Le premier problème est un bug qui survient très rarement mais qui peut être gênant lors d'une partie. En effet si lors d'un saut, le personnage atterri sur un angle de plateforme, le programme considère que le joueur se situe sur la plateforme ET qu'il chute en même temps. Il est donc impossible pour le programme de lancer l'animation d'atterrissage, qui doit être finie pour pouvoir ressauter. Ceci engendre un blocage du personnage qui n'avance pas, ne tombe pas ou ne monte plus, mais qui ne peut pas sauter pour autant car il est considéré comme en cours de chute. Pour générer ce bug il faut que le personnage arrive au pixel près sur l'angle, c'est pourquoi nous considérons que ce n'est pas un problèmes majeur.

Ensuite, l'utilisateur du programme pourra rapidement se rendre compte de l'absence d'un « Mode solo ». Il n'est en effet seulement possible de jouer à deux minimum. Ceci peut se régler en sélectionnant le bouton **2 Joueurs** et en laissant l'un des deux perdre tout en continuant de contrôler l'autre. Nous aurions pu, une nouvelle fois avec plus de temps

reprogrammer une nouvelle classe *ModeDeJeu.java* qui se composerait de deux boutons (**Solo** et **Multijoueur**), envoyant respectivement dans les classes *ChoixPerso.java*, avec un seul personnage, et *Menu.java*, permettant de choisir le nombre exact de joueur.Il aurait aussi été envisageable de créer un mode pour jouer en solo, permettant ainsi de rajouter des vies supplémentaires au joueur, pour repartir de la ou il est mort, par exemple mais comme nous avons choisi un principe de survival nous avons privilégié le jeu à plusieurs qui est plus amusant que le jeu solo.

De même, la taille de la fenêtre étant défini dans le programme, celle-ci ne s'adapte pas à la taille de l'écran de l'ordinateur sur lequel le jeu est lancé. Dans la plupart des cas, ceci n'est pas gênant. La taille définie dans le programme (1024x768) est adaptée à la majorité des écrans d'ordinateur. Mais ceci pourrait éventuellement être amélioré pour que la fenêtre s'adapte à l'ordinateur.

Toujours en parlant de la fenêtre dans laquelle tourne notre programme, il est possible de la déplacer, ce qui peut engendrer des problèmes si l'utilisateur arrive à sortir ladite fenêtre de son écran. Mais ceci n'est qu'un problème mineur, dépendant de l'utilisateur.

On aurait pu pour améliorer le jeu en créant plusieurs niveaux de difficultés, qui influeraient sur la vitesse d'avance de l'écran, ou encore sur la difficulté liée au terrain.
Ou même plus simplement faire en sorte que la difficulté soit liée au score actuel du joueur (difficulté croissante avec le temps).

On aurait pu alors créer des bonus pour aider le joueur à faire fasse à l'augmentation de la difficulté.

On aurait également pu rajouter la possibilité de mettre le jeu en pause.

Une dernière parmi les multiples améliorations possibles serait d'afficher des pseudos des joueurs au lieu de "joueur 1", etc. Ceci a été pensé pendant la création du programme (présence d'un attribut Pseudo), qui n'est pas utilisé faute d'avoir eu le temps de faire un menu d'enregistrement des pseudos des joueurs, ou de mettre des pseudos à rentrer lorsque l'on choisit un personnage par exemple.

Bibliographie :

- Doc d'Oracle : https://docs.oracle.com/javase/7/docs/
- Google : http://google.fr
- Zeste De Savoir : https://zestedesavoir.com/tutoriels/646/apprenez-a-programmer-en-java/