IPO – TP-projet : Hop !

`https://public.lmf.cnrs.fr/~blsk/IPO/`

Mais quelle était cette idée de descendre dans la cheminée d'un volcan, « pour voir s'il faisait chaud en bas » ? La lave est en effet brûlante, et elle est en train de remonter de plus en plus vite. Vous n'avez plus qu'à sauter de rocher en rocher pour tenter de sortir, par le haut, de cette histoire à rôtir debout.

Objectif du TP.

Ce TP-projet est à réaliser en binôme. Vous allez réaliser un nouveau jeu pour lequel vous pouvez vous inspirer de ce qui a été réalisé au TP8/8bis. Votre travail sera à compléter pour le 15 décembre (23h59) et donnera lieu à une soutenance la semaine du 16 décembre. Le tout formera votre note de contrôle continu. On recommande l'utilisation d'un gestionnaire de version (`Git`). Vous trouverez sur la page du cours une archive `.jar` contenant une version de démonstration minimale de ce jeu.

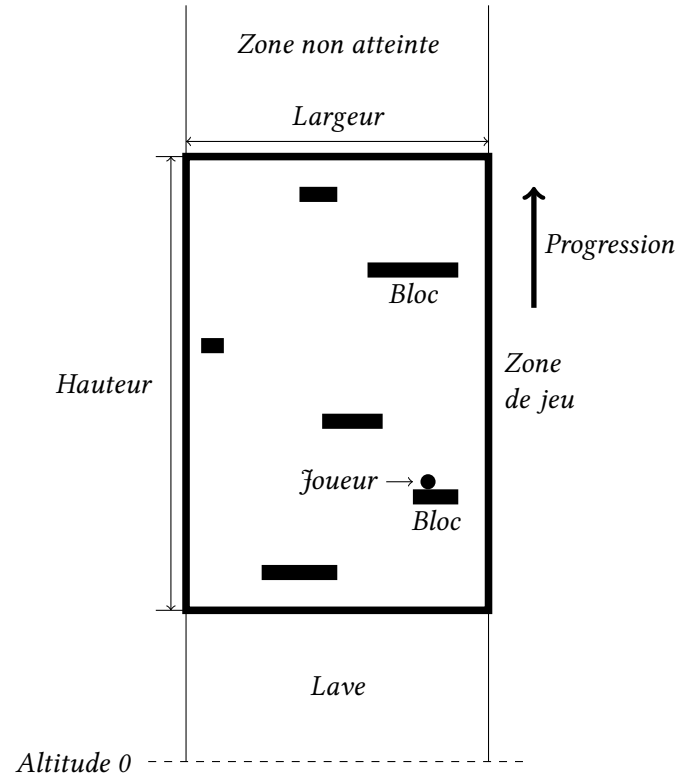
Description du jeu.

Volcan. Le volcan (classe `Field`) est un conduit vertical de largeur fixe, contenant des blocs rocheux horizontaux de tailles variées sur lesquelles le joueur peut prendre pied. Chaque élément est repéré dans le volcan par :

- une altitude, comptée à partir de 0 ;
- une coordonnée horizontale, comprise entre 0 et la largeur.

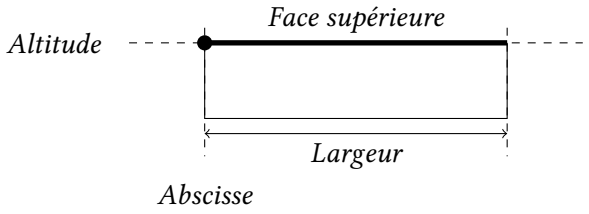
La zone de jeu est une portion du volcan de hauteur fixe à partir de l'altitude atteinte par la lave. L'état du jeu est mis à jour toutes les 40 millisecondes (25 tours de jeu par seconde), des deux manières suivantes :

- le joueur se déplace, suivant les touches actionnées par le joueur et les (dures) lois de la gravité,
- l'altitude atteinte par la lave augmente, entraînant un glissement vers le haut de la zone de jeu, qui se traduit à l'écran par un déplacement apparent vers le bas des blocs et du joueur.



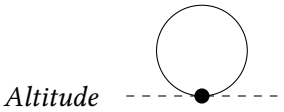
Le jeu se termine lorsque le joueur touche le bas de l'écran, que ce soit à cause d'une chute ou car il se fait simplement « rattraper » par le déplacement de la zone de jeu. De nouveau blocs apparaissent périodiquement en haut de l'écran, à mesure qu'ils sont « découverts » par la progression de la zone de jeu.

Blocs rocheux. Les blocs rocheux (classe `Block`) sont représentés par des rectangles horizontaux. Dans le jeu, on ne s'intéresse qu'à la face supérieure de chaque bloc, sur laquelle le joueur peut se poser. Cette face supérieure est caractérisée par une altitude, une position horizontale (abscisse) et une largeur.



On ne s'intéresse qu'aux blocs qui sont dans la zone de jeu. Lors de la phase montant d'un saut, le joueur peut traverser un bloc. Lors de la phase descendante en revanche, le joueur se pose sur la face supérieure du premier bloc rencontré.

Mouvements du personnage. Le personnage (classe `Axel` [1]) est représenté par un cercle, dont on prend comme coordonnée de référence le point le plus bas.



Un état booléen indique si le personnage choit ou est au repos (sur un bloc). Le personnage a également une vitesse verticale, qui décrit sa variation d'altitude lors d'un tour de jeu. Cette vitesse est positive lors de la phase ascendante d'un saut, et négative lors de la phase descendante d'un saut ou d'une chute. À chaque tour, cette vitesse est modifiée par la gravité et les éventuelles actions du joueur. Note : si le personnage est au repos, sa vitesse verticale vaut nécessairement zéro, mais l'inverse n'est pas vrai.

À chaque tour, on calcule la prochaine position du personnage en fonction de son état courant, et de quatre booléens supplémentaires indiquant, pour chacune des flèches directionnelles, si elle est actuellement enfoncée par le joueur.

Mise à jour de l'abscisse
Si les flèches gauche ou droite sont enfoncées, le joueur se déplace à chaque tour d'une distance fixe dans la direction concernée, dans les limites du terrain. En l'absence d'action du joueur, il n'y a pas de déplacement horizontal. Ces règles valent quel que soit l'état du personnage.

Mise à jour de l'altitude, pour un personnage au repos
Si la flèche haut est enfoncée, le personnage commence un saut, ce qui fixe sa vitesse verticale au maximum, et le fait passer en état de chute. Le personnage se déplace vers le haut de la distance donnée par sa vitesse. Sinon, la vitesse verticale reste à zéro et l'altitude du personnage ne change pas.

Mise à jour de l'altitude, pour un personnage en chute
La vitesse verticale est décrémentée d'une constante représentant la gravité. Si la touche bas est enfoncée, la vitesse verticale est décrémentée d'une valeur fixe supplémentaire.

- Si la vitesse verticale est positive, le personnage se déplace vers le haut de la distance correspondante.
- Si la vitesse verticale est négative, le personnage se déplace vers le bas de la distance correspondante. Exception : si ce mouvement de descente lui fait traverser la face supérieure d'un bloc, alors le personnage s'arrête sur la surface du bloc et passe en état de repos avec une vitesse verticale à zéro.

1. Question subsidiaire littérature : quel est le nom du famille du personnage ?

Travail à réaliser.

Vous trouverez sur la page du cours les squelettes d'un fichier principal `Hop` (moteur principal du jeu, et démarrage), d'une interface graphique `GamePanel` (affichage et traitement du clavier), et de classes `Field`, `Block` et `Axel`. Vous devez compléter cette base de manière à obtenir un jeu opérationnel.

Suggestion de progression. La liste ci-dessous vous suggère un ordre dans lequel introduire les différents éléments, qui permet de tester la progression du projet à chaque étape. Elle contient également quelques détails supplémentaires sur les tâches à réaliser.

1. Donner à la classe `Field` un constructeur, qui introduit un ensemble de blocs aléatoires à des intervalles d'altitude réguliers. *L'altitude étant fixée a priori, le caractère aléatoire porte sur la position et la largeur de chaque bloc. Cela peut nécessiter l'ajouter à la classe* `Block`*, de méthodes ou de constructeurs adaptés.*
2. Dans la classe `GamePanel`, compléter les méthodes d'affichage de sorte à ce que les blocs et le personnage s'affichent à l'écran.
3. Dans la classe `GamePanel`, ajouter l'annotation **`implements`** `KeyListener` et compléter les méthodes adaptées pour qu'enfoncer une touche directionnelle définisse à **`true`** l'attribut du personnage donnée par le tableau ci-dessous, et que relâcher une touche directionnelle définisse le même attribut à **`false`**.

←	→	↑	↓
`gauche`	`droite`	`saute`	`plonge`

4. Dans la classe `Axel`, compléter la méthode `computeMove` de sorte à mettre à jour les attributs `dx` et `dy` décrivant le déplacement du personnage sous l'action des touches et de la gravité (sans se soucier dans un premier temps de l'interaction avec les blocs). Modifier également la méthode `over` de la classe `Hop` pour y introduire le bon critère de fin de partie.
5. Dans la classe `Axel`, compléter la méthode `checkCollision` de sorte à déterminer si le personnage atterrit sur un bloc (ou au contraire, tombe d'un bloc après en avoir dépassé l'extrémité), et mettre à jour l'état du personnage en conséquence. *Cette méthode peut appeler une méthode auxiliaire, définie dans cette classe ou dans une autre, qui renvoie le bloc ou les blocs avec lesquels le personnage est susceptible d'interagir.*
6. Compléter la classe `Hop` d'une méthode testant si la partie est finie, c'est-à-dire si le joueur a dépassé la limite basse de l'écran, et faire en sorte que cette vérification soit faite à chaque tour. *À nouveau, cette méthode peut s'appuyer sur une méthode auxiliaire définie dans une autre classe.*
7. Introduire un nouveau composant graphique affichant en temps réel le score du joueur. On définit le score comme l'altitude maximale d'un bloc que le joueur a atteint.
8. Compléter la classe `Field` avec une méthode de mise à jour, qui doit être appelée à chaque tour de jeu, qui incrémente l'altitude de la zone de jeu. *Attention : il faut s'assurer que la zone de jeu continue à contenir des blocs à des intervalles d'altitude réguliers.*
9. Ajouter un mécanisme d'augmentation progressive de la difficulté du jeu : lorsque le joueur atteint certains score prédéfinis, les blocs deviennent plus petits et l'altitude de la zone de jeu augmente plus rapidement. Compléter l'affichage du score pour qu'il montre également le niveau atteint.
10. Ajouter, à volonté, de nouveaux éléments pour enrichir le jeu. Liste non limitative :
    - Blocs particuliers (par ex. mobiles, de taille variable, ou déclenchant un effet).
    - Pièces que le joueur peut ramasser pour augmenter son score.
    - Mouvement spécial activable une fois par saut (double saut et/ou déplacement latéral).
    - Bonus, que le joueur peut ramasser pour modifier temporairement des paramètres du jeu.
    - Bonus, que le joueur peut ramasser et utiliser plus tard.
    - Autres créatures, amies ou hostiles.
    - Menu de jeu.
    - Graphismes, sons.
    - Mode multijoueur.

Paramètres du jeu. À titre d'information, voici les valeurs utilisées dans la version de démonstration pour les différents paramètres du jeu (constantes du jeu, et valeurs dépendant du niveau de difficulté). On vous recommande de les utiliser dans un premier temps pour un jeu équilibré. Vous pouvez ensuite les ajuster en fonction de vos propres expérimentations et des améliorations que vous apportez au projet. Vous pouvez y compris transformer certains constantes en valeurs qui dépendent du niveau de difficulté.

Constantes

Classe	Constante	Signification	Valeur
`Hop`	`WIDTH`	largeur du terrain	400
	`HEIGHT`	hauteur du terrain	600
	`DELAY`	durée d'un tour de jeu (millisecondes)	40
`Field`	`START_ALTITUDE`	altitude du premier bloc	40
	`ALTITUDE_GAP`	écart d'altitude entre deux blocs	80
`Axel`	`LATERAL_SPEED`	valeur de déplacement latéral par tour	8
	`JUMP_SPEED`	vitesse verticale au début d'un saut	20
	`GRAVITY`	décroissance de la vitesse verticale à chaque tour	1
	`DIVE_SPEED`	décroissance suppl. vitesse verticale en piqué	3
	`MAX_FALL_SPEED`	limite de vitesse verticale négative	-20
`GamePanel`	`BLOCK_HEIGHT`	hauteur des blocs (affichage)	10
	`AXEL_WIDTH`	largeur du personnage (affichage)	10
	`AXEL_HEIGHT`	hauteur du personnage (affichage)	10

Valeurs dépendant du niveau

Niveau	0	1	2	3	4	5	6
atteint à l'altitude	0	80	800	2000	3200	4800	7200
Défilement vertical du terrain	0	1	2	3	4	5	6
Largeur bloc minimale	50	45	40	35	30	25	20
Largeur bloc maximale	100	90	80	70	60	50	40