Langages et Génie Logiciel
Projet

2008—2009

Le but du projet est de produire un jeu vidéo à la Worms. Plusieurs personnages se trouvent dans un décor en deux dimensions. Ils s'envoient des missiles les uns sur les autres, ce qui a pour effet secondaire de détruire progressivement le décor. L'étudiant non-violent pourra éventuellement remplacer les missiles par des fleurs.

1 Travail demandé

Vous devez implémenter le jeu en OCaml. Vous n'avez pas à dessiner les graphismes, qui vous sont fournis. Vous devez vous concentrer sur la structure de votre projet en utilisant les notions vues en cours, notamment :

- en utilisant des modules répartis en plusieurs fichiers ;
- en choisissant avec soin les types de vos structures de données ;
- en utilisant à bon escient la notion d'abstraction.

La note de votre projet en prendra largement compte.

Vous commencerez par implémenter la base du jeu présentée dans la section 3. Ensuite, vous pourrez implémenter les autres caractéristiques présentées dans la section 4.

Vous devrez rendre votre projet par mail[1] sous la forme d'une archive `.tar.gz` comprenant les fichiers nécessaires pour compiler, ainsi qu'un `Makefile`. La commande `make` doit compiler votre projet sans erreur et sans warning.

2 Base fournie

Les fichiers suivants vous sont fournis[2] :

- les images du jeu ;
- le module `Draw` contenant des fonctions pour dessiner le décor, les personnages et les missiles ;
- le module `Input` permettant de gérer les entrées clavier et souris.

[1] `bardou@lri.fr, conchon@lri.fr`

[2] `http://romain.bardou.fr/teaching/lglworms/base.tar.gz`

Vous pouvez modifier ces fichiers à votre convenance.

Pour compiler votre projet vous aurez besoin d'inclure les librairies `bigarray`, `sdl`, `sdlloader` et `sdlttf`.

Pour fermer la fenêtre du jeu, vous devez utilisez la fonction `Sdl.quit`, de type `unit -> unit`.

3 Jeu de base

Le but de cette partie est d'avoir une base du jeu dans laquelle un personnage est capable de se déplacer à gauche et à droite dans un décor en deux dimensions. Le décor peut avoir une forme quelconque et le personnage doit pouvoir grimper les pentes et tomber dans les trous.

Déroulement du temps L'état du jeu contient toutes les informations nécessaires comme la forme du décor ou la position des personnages. Toutes les T secondes, où T est une durée bien choisie telle que 0.01, l'état du jeu est mis à jour et réaffiché. Le passage de l'état courant à l'état suivant sera appelé une *étape* du jeu.

Pour mesurer le temps on pourra utiliser la fonction `Sdltimer.get_ticks`, de type `unit -> int`. Pour bloquer le jeu quelques instants on pourra utiliser la fonction `Sdltimer.delay`, de type `int -> unit`.

Décor Votre décor est une matrice dont chaque case correspond à l'état d'un point de l'écran (un *pixel*) : *solide* ou *vide*. Un point solide ne peut pas être traversé par les personnages.

L'axe des abcisses est appelé X et ira de gauche à droite, le 0 correspondant à la limite du décor à gauche. L'axe des ordonnées est appelé Y et ira de haut en bas, le 0 correspondant à la limite du décor en haut.

On pourra éventuellement écrire le jeu comme un foncteur paramétré par, entre autres, le codage du décor, qui pourra être réalisé à l'aide d'une matrice ou d'une autre structure de données que vous estimerez efficace.

Notez que dans la base fournie, le module `Draw` construit une fenêtre de taille 800×600. Pour simplifier, vous pouvez utiliser une taille identique pour le décor.

Personnage Votre personnage est représenté par les informations suivantes :
- position $\vec{P}$;
- vitesse $\vec{V}$.

La vitesse indique le nombre de pixels qui doit être ajouté à la position courante $\vec{P}$ pour obtenir la nouvelle position $\vec{P'}$ du personnage au prochain état :

$$\vec{P'} = \vec{P} + \vec{V}$$

La vitesse du personnage est modifiée lorsque le personnage appuie sur une touche (gauche ou droite), lorsque le personnage entre en collision avec le décor, et par la gravité.

Il sera une bonne idée d'implémenter un module `Vector` pour manipuler des vecteurs de flottants.

Gravité Pour simuler la gravité, il suffit de rajouter, à chaque étape, un vecteur constant $\vec{G}$ à la vitesse du personnage. La vitesse $\vec{V}$ devient alors :

$$\vec{V'} = \vec{V} + \vec{G}$$

A vous de choisir la valeur de $\vec{G}$, en général dirigé vers le bas.

Collisions avec le décor Pour ne pas que votre personnage traverse le décor, avant d'ajouter sa vitesse à sa position pour le déplacer, on vérifie que la nouvelle position est une position valide. Une position est valide si le personnage n'est pas coincé dans le décor.

Pour tester si une position $\vec{P}$ est valide, une méthode simple consiste à tester si le pixel $\vec{P}$ est solide ou non. Mais cette méthode est imprécise, car le joueur n'occupe pas qu'un seul pixel. Une méthode plus précise consiste à tester si l'un des pixels occupés par le personnage est solide. On aussi compter le nombre de pixels occupés par le personnage et décider d'un seuil au-delà duquel on considère que la position n'est pas valide.

Si la nouvelle position $\vec{P'}$ n'est pas valide, il y a deux cas. Soit le personnage essaye de grimper une pente, soit il essaye de traverser un mur. Pour distinguer ces deux cas, il suffira de regarder quelques pixels au-dessus de $\vec{P'}$. Si l'un de ces pixels est valide, alors le personnage essaye de grimper une pente (autrement dit, le mur qui est devant lui est en fait un petit muret). Pour le faire grimper il suffira de changer $\vec{P'}$ par cette position valide. Si, au contraire, le joueur entre dans un mur, il ne faut pas mettre à jour sa position mais mettre sa vitesse à zéro.

4 Suppléments

Il s'agit maintenant d'étendre le jeu pour le rendre plus intéressant.

Missiles Donnez la possibilité au joueur de lancer un missile. Par exemple, le joueur pourra cliquer sur l'écran à une position $\vec{P'}$ pour envoyer un missile avec une vitesse de départ de $k(\vec{P'} - \vec{P})$ où $\vec{P}$ est la position de son personnage et k une constante bien choisie. Le missile devra subir la gravité et exploser (disparaître) s'il entre en collision avec le décor.

Points de vie Chaque personnage a un nombre de points de vie de départ. Ce nombre doit être affiché au-dessus du personnage. Il diminue si le personnage est proche de l'explosion d'un missile. S'il devient négatif, le personnage disparaît.

Destruction du décor Lors de l'explosion d'un missile, tous les pixels solides dans un rayon donné autour de l'explosion passent à l'état vide. Vous rajouterez un état *indestructible*. Les pixels indestructibles sont solides (ils ne peuvent être traversés) mais ne sont pas détruits par les explosions.

Multijoueur Donnez la possibilité à plusieurs joueurs de jouer en même temps avec chacun leur personnage, tout le monde jouant sur le même clavier. Vous pouvez choisir entre un mode temps réel, où tout le monde joue en même temps avec chacun des touches différentes, ou un mode tour par tour, où les joueurs contrôlent leur personnage chacun leur tour.

Armes multiples Donnez la possibilité au joueur de choisir entre plusieurs armes. L'une de ces armes sera le missile de base décrit plus haut. Les autres armes pourront avoir des trajectoires différentes, être plus rapides, lancer plusieurs missiles de moindre puissance, avoir des munitions limitées, modifier le décor différemment, ...

Autres idées Vous pouvez implémenter d'autres idées mais rappelez-vous que le plus important est d'implémenter chaque nouvelle possibilité d'une façon intelligente et modulaire. La qualité du code produit sera plus importante que la quantité d'idées implémentées.