Emmanuelle RICHELY, Kristina KINCELOVA, Guillaume MARIE, Noé DELARGILLIÈRE
groupe 42

Liner Rider

1) Cahier des charges détaillé

L'objectif est de créer un jeu permettant de tracer une série de segments et de courbes, afin de définir un circuit pour un objet (une bille, une luge ou bien une moto). Le but du jeu est de parvenir à lancer l'animation de l'objet, en le faisant partir d'un point de départ mobile.

Le programme met en jeu des éléments de mécanique (comme la prise en compte de l'énergie cinétique, des frottements, de la gravité, des rebonds…), ainsi que des éléments décoratifs à travers la mise en place d'un système de couleurs de traits et d'animations lors des rebonds…

Les objectifs des listes suivantes sont ambitieux, et ils ne seront peut être pas tous réalisés à la fin du projet le 4 Juin.

Le tracé des courbes :

- création d'un segment entre 2 points ;
- création d'une courbe suivant le curseur de la souris ;
- correction des tracés avec une gomme (pour effacer précisément) et une poubelle (pour tout effacer) ;
- plusieurs types de traits pour différents interactions avec l'objet.

Déroulement du jeu :

- une première phase de jeu pour créer le parcours ;
- un bouton "play" pour faire apparaître l'objet et lancer l'animation ;
- un outil pour sauvegarder un circuit en mémoire et pouvoir le reprendre une autre fois.

Interface graphique attendue :

2) Description des problèmes et résolutions

Création des lignes par l'utilisateur :

Avant d'accéder au jeu, un menu s'affiche, il suffit de cliquer sur le bouton "Jouer" pour ouvrir le jeu.

Une barre d'outils permet l'accès aux différents outils et couleurs. L'outil sélectionné est signalé par une surbrillance bleue. Deux outils permettent de créer des segments et des courbes. Les traits sont affichées provisoirement lors du tracé pour avoir un aperçu du résultat.

- crayon

On crée une arrayList de Lignes, qui est un objet qui accepte n'importe quel type de donnée, et n'a pas de limite. On calcule la longueur du segment qui relie la fin de la Ligne précédente avec la position de la souris. Dès que cette longueur dépasse une valeur fixée, la petite droite est tracé. L'ensemble de ces petites droites donne l'impression d'un tracé fluide et "à la main".

- gomme

L'utilisateur peut corriger des segments mal positionnés grâce à l'outil gomme, ou bien tout effacer avec l'outil poubelle.
Quand nous cliquons sur un endroit du gamePanel, on récupère les coordonnées de la souris et on parcourt toute la arrayList de Lignes en calculant la distance entre ce point cliqué et chaque ligne de la liste. Si une une valeur est inférieure à 30 pixels, alors la ligne est enlevé de la liste. Un repaint automatique permet de supprimer la ligne visuellement.

Icônes Play et Stop

- play

La bille part du cercle rouge. Si la bille sort du cadre de l'écran, le compteur est arrêté. On peut alors appuyer à nouveau sur play et la bille est réinitialisée à son point de départ et nous pouvons relancer l'animation.

- stop

Arrête la bille dans son trajet et un clic sur play réinitialise la bille au départ.

Gestion des collisions :

Le comportement de la bille varie beaucoup selon qu'elle glisse sur une pente ou qu'elle soit en chute libre. On doit donc savoir en permanence s'il y a collision avec une ligne ou pas.
On crée une méthode collision qui retourne une variable du type returnCollision, constituée d'un booléen qui indique s'il y a collision entre la Bille, et d'uneLigne. Celle-ci correspond à la ligne avec laquelle il y a collision, et est nulle s'il n'y en a pas. Le paramètre est la liste de lignes tracées.
On mesure la distance entre la ligne et le centre de la bille. Il y a collision si la distance est inférieure au rayon de la bille.

Gestion du mouvement de l'objet :

L'objet adopte deux comportements différents, selon qu'il soit en contact avec une ligne ou non. C'est la méthode collision qui lui donne cette information sous forme de booléen.
On commence par garder en mémoire les positions de la bille x et y en créant deux nouvelles variables. Puis on calcule les nouvelles positions de l'objet en chute libre selon les formules suivantes :

```
x=x+dx*dt
y=y-0.5*g*Math.pow(dt,2)+dy*dt
```

On teste la collision de ces nouveaux points avec toute la liste de lignes (arrayList).

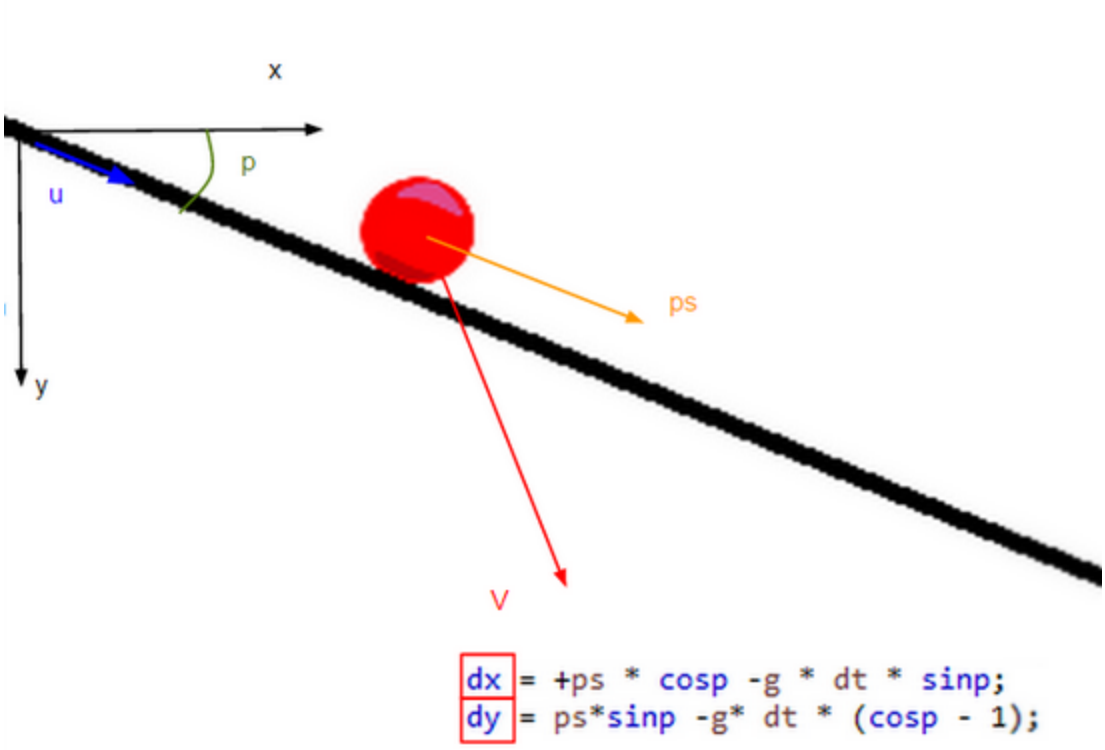
- Cas 1: pas de collision

Si il n'y a pas de collision alors on garde les coordonnées x et y calculées précédemment.
On calcule les nouvelles vitesses : dx ne change pas
dy=-g*dt+dy;
Pour augmenter la fluidité du mouvement nous avons du créer un dispositif qui permet de ne pas re-calculer les vitesses et coordonnées lorsque la bille suit une même courbe lorsqu'il n'y a pas de rebond. Cela se fait sous forme de boucle if qui garde en mémoire les 5 dernières collisions de la bille.

- Cas 2: collision

On récupère cos(p) et sin(p), les coordonnées du vecteur directeur de la ligne avec laquelle il y a collision.
On retourne à la position actuelle de la bille(enregistrée précédemment), là ou il n'y a pas encore collision.
On regarde à l'aide d'une boucle if si la norme de la vitesse de collision entraîne un rebond, c'est-à-dire si elle est supérieure à une valeur ecmax fixée auparavant.

- s'il n'y a pas de rebond :




```
dx = +ps * cosp -g * dt * sinp;
dy = ps*sinp -g* dt * (cosp - 1);
```

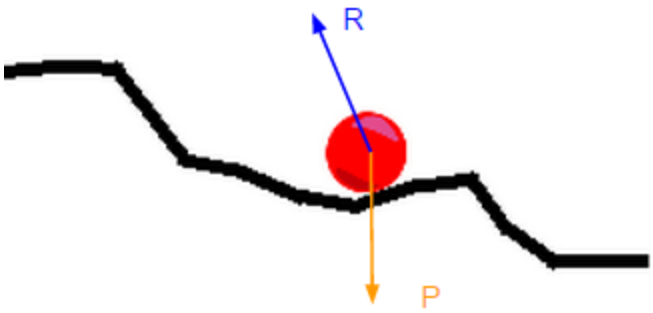
Changement de direction de la vitesse avec un coefficient dépendant de l'angle d'arrivée :

- Utilisation d'un produit scalaire entre V et u vecteur unitaire de la droite

- On projette la norme du produit scalaire obtenu selon le vecteur unitaire de la droite

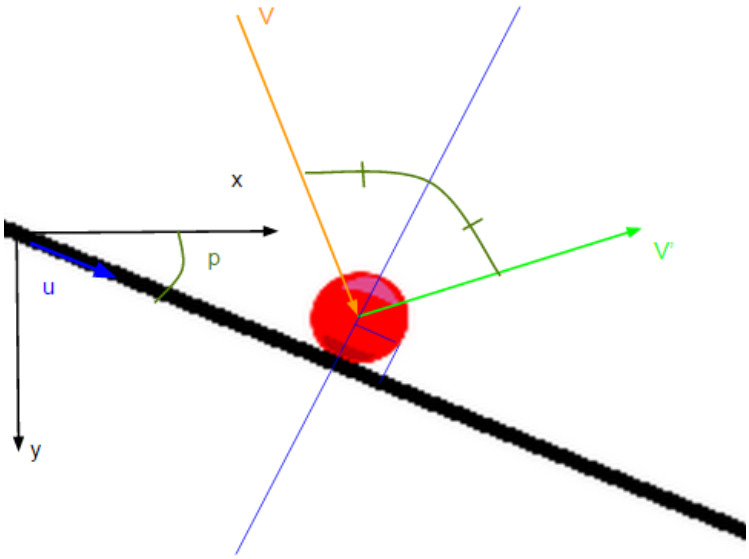
- Utilisation de la méthode position() car calculs différents en fonction d'une arrivée par en bas ou par en haut

Puis utilisation du PFD pour établir l'équation du mouvement prenant en compte le poids et la réaction du support



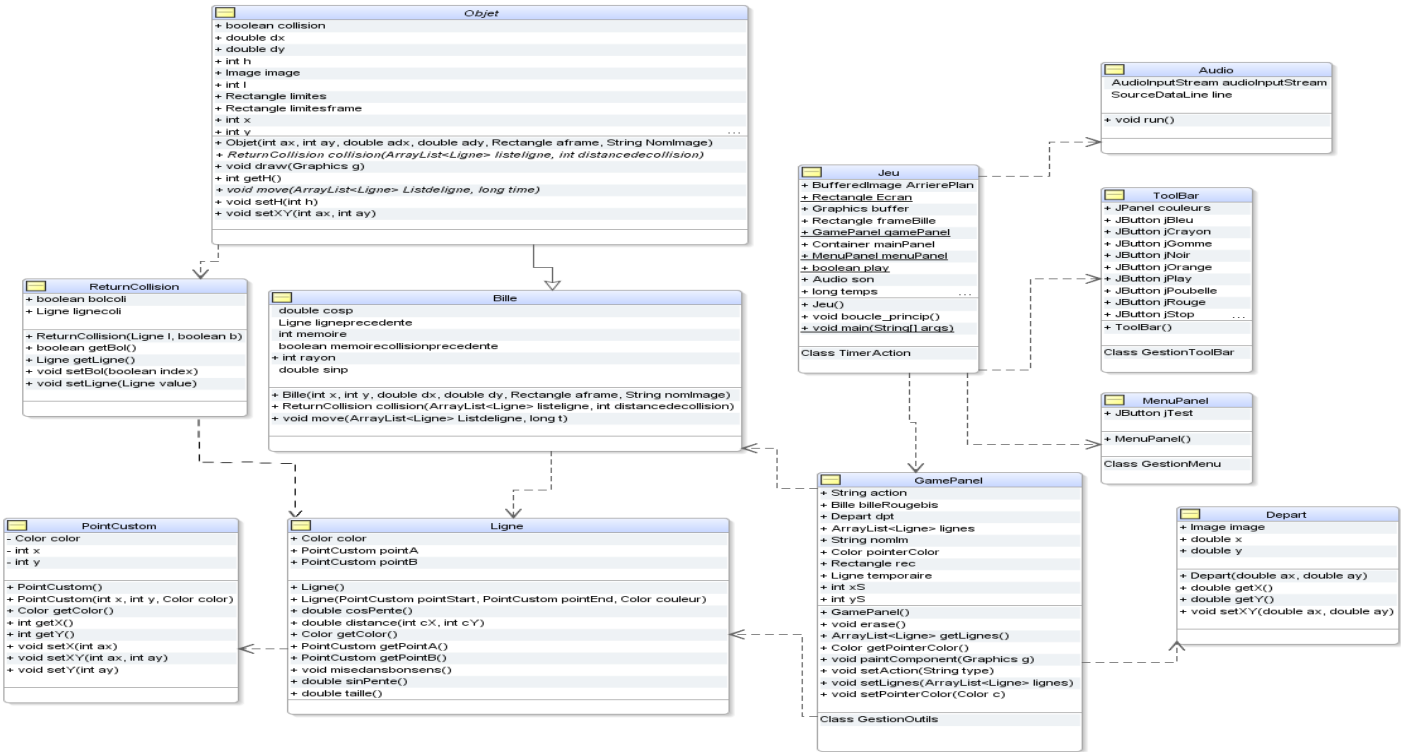

```
//Calcul de la position x et y du prochain point, avec prise en compte de la gravité
// et de la réaction du support
x = (int) (x + (-0.5) * g * Math.pow(dt, 2) * sinp + dx * dt);
y = (int) (y - 0.5 * g * Math.pow(dt, 2) * (cosp - 1) + dy * dt);
```

- s'il y a rebond:



Changement de direction de la vitesse avec un coefficient dépendant de l'élasticité voulu pour la bille

3) Structuration des données



cf dossier sources

4) Bibliographie

- Javadoc
- TP 8 Space Invaders
- exemple snake pour la structure du jeu : http://michel-douez.developpez.com/articles/java/snake/
- la classe audio de l'utilisateur "thehornycocoboy" de forum openclassrooms(http://openclassrooms.com/forum/sujet/jouer-de-la-musique-96546)
- forum http://stackoverflow.com/
- cours de mécanique (physique -1ère année et mécanique -2ème année)

5) Suggestions d'amélioration et bogues connus

1) L'objet que nous avons modélisé est une bille. Cela nous semblait plus simple car son comportement est homogène, il n'y a pas d'articulations et la plupart de nos calculs considèrent la bille comme un simple point. Une amélioration serait donc de modéliser un objet plus complexe, un bonhomme sur une

luge par exemple, comme c'est le cas dans le jeu LineRider original. L'objet n'a plus le même comportement en chacun de ses points.
2) On pourrait donner l'impression que l'objet roule en effectuant un changement d'image dont la fréquence est proportionnelle à la vitesse de la bille (roulement sur 8 images par exemple).
3) Une autre amélioration serait de créer une fenêtre non statique, à la manière d'une caméra, qui avance avec l'utilisateur et permet de créer un parcours plus long qui devient un véritable univers.
4) Nous avons décidé d'utiliser GitHub pour gérer plus facilement la mise en commun de notre travail, un logiciel de gestion de versions décentralisée ce qui permet aux différents contributeurs de créer des différentes versions du programme. Chacun peut désormais tester son code sans être limité par les actions des autres. De plus, c'est très avantageux pour faire des brouillons car on peut facilement supprimer une branche sans modifier la branche principale. On ne travaille pas constamment en ligne car nous ne sommes pas obligés d'être connecté au réseau constamment, également le fait de travailler off-line permet un fonctionnement plus rapide.

Cependant le logiciel nécessite certains pré-requis. En effet, l'application en elle-même ne permet pas d'exécuter certaines opérations tel que abandonner les modifications locales et télécharger la nouvelle version en ligne. Pour faire cela, il faut passer par une ligne de commande GitShell qui n'est pas très accessible. N'ayant pas fait beaucoup de cours sur ce sujet, nous avons rencontrés des difficultés à résoudre ces problèmes.

Ainsi, même si GitHub nous a parfois ralenti, cet outil nous a quand même permis de partager notre travail de façon optimale même quand nous n'étions pas tous en ligne. Nous n'avons pas rencontré de grands problèmes tant que nous travaillions sur des parties différentes du projets (partie graphique, mouvement de la bille), c'est en effet dans la partie finale du projet où nous avons tous mis ensemble que les choses se sont compliquées.

Pour améliorer le travail et le rendre plus efficace, un meilleur renseignement sur l'environnement GitHub serait souhaitable.
5) On pourrait mettre en place un système de curseurs pour pouvoir changer les constantes de gravité, de rebond, de frottement,...
6) Ajouter plusieurs musiques et des bruitages serait également intéressant. Pour cela, il faudrait utiliser une classe Audio différente qui prend en paramètre le nom du son voulu.
7) l'image scintille, nous pourrions utiliser un buffer pour plus de fluidité.

Bogues connus

Les bogues connus concernent essentiellement l'animation de la bille, qui a parfois des comportements non intelligibles pour nous pauvres humains. La gomme ne parvient pas à effacer le dernier trait, nous pensons que c'est uniquement un bogue graphique.
Nous n'utilisons pas les même logiciels de développement (Eclipse ou JDev), d'où la multiplicité des images et des sons dans le dossier source.

Conclusion

Ce projet d'informatique a été une expérience très intéressante car nous avons appris à utiliser nos connaissances acquises depuis 2 ans dans un cadre plus libre. Cela nous a également permis d'améliorer notre technique de travail en groupe, de répartition des tâches et de respect d'un délai imparti.