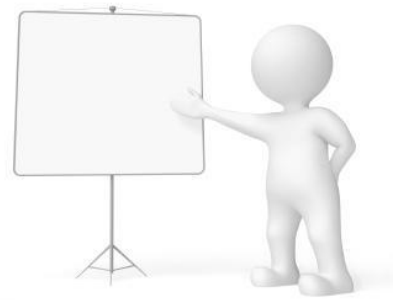
Presentation projet Java “GRAVITY”

DJEBALI Tameur - ETCHEVERRY Florian - COSTEDOAT Paul

Sommaire

1. Presentation Projet et équipe
2. Analyse UML
3. Solutions apportées
4. Evolutions futures
5. Video de demonstration



1.Présentation du projet

Présentation

En quelques mots

- L'équipe:

DJEBALI Tameur

ETCHEVERRY Florian

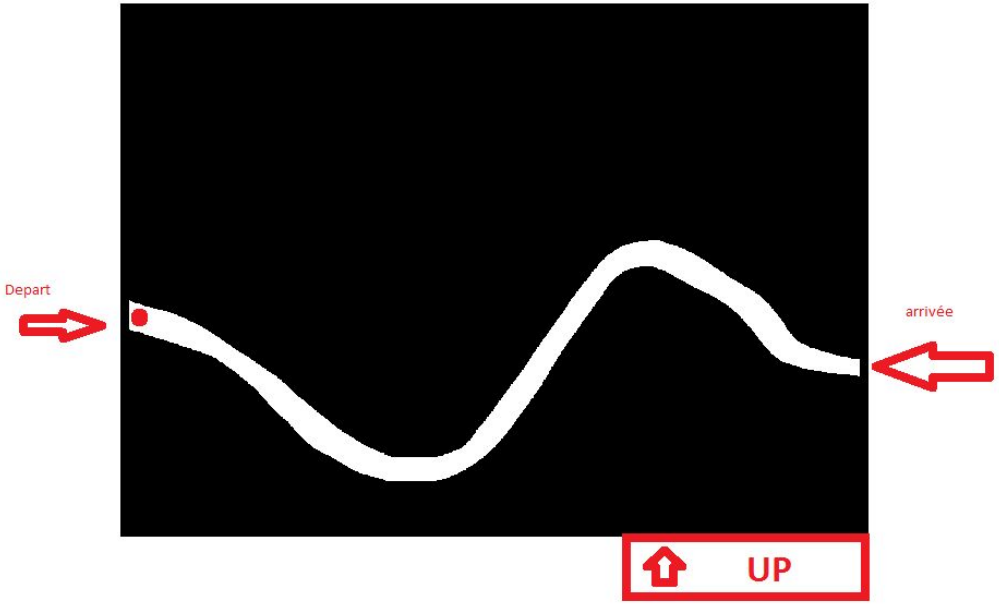
COSTEDOAT Paul

- Conception en Java d'un jeu retro
  - Le joueur contrôle un objet
  - L'objet doit traverser une fenêtre en évitant des obstacles

Présentation

Esquisse du visuel

- Première esquisse du visuel:



Fonctionnalités Attendues

- Plusieurs niveaux de plus en plus difficiles
- Identification du joueur par un pseudo
- Position de l'objet avance automatiquement sur l'axe X et descend sur l'axe Y
- Bouton qui permet de le faire monter l'objet sur l'axe Y
- On gagne si on atteint la fin du niveau
- On perd si l'objet entre en contact avec un obstacle
- Sauvegarde automatiques des scores

2.Analyse UML

Diagramme des cas d'utilisation

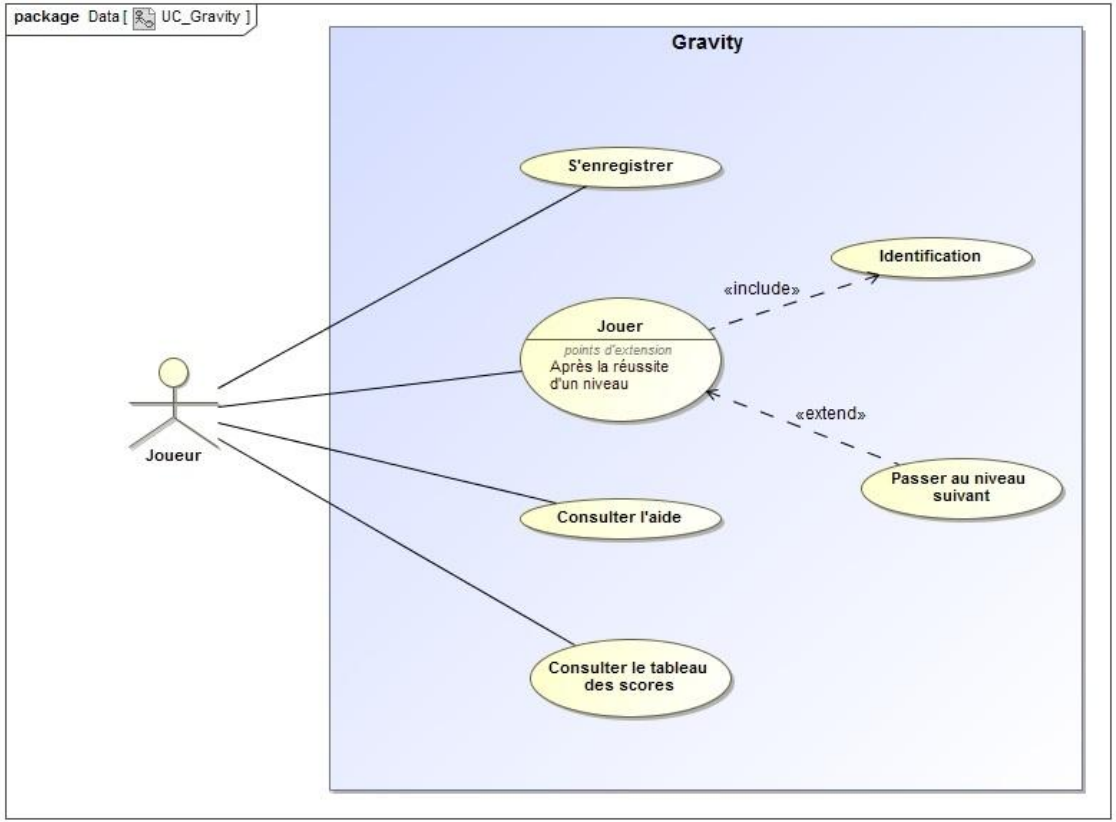
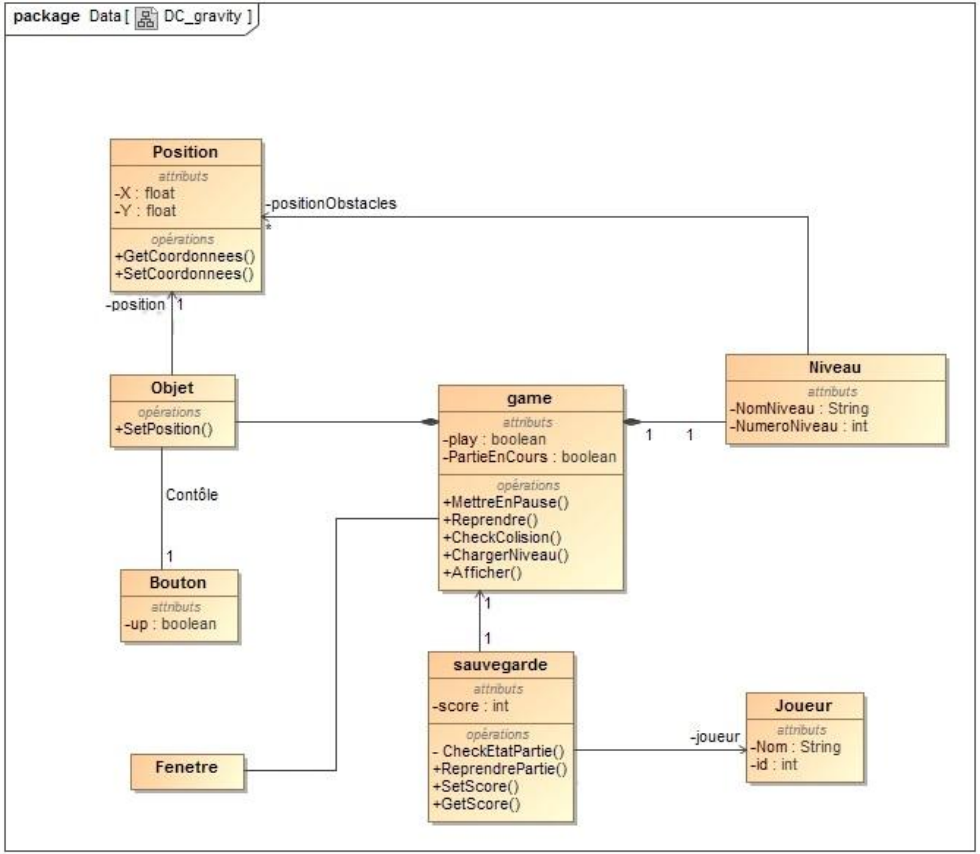


Diagramme de classe



3. Solutions apportées

Mouvement de l’objet

Génération

- L’objet contrôlé est dessiné grâce à la méthode paintComponent de la classe javax.swing.JComponent

- On redessine l’objet à chaque fois que la position de l’objet est modifiée
  - Grâce à la méthode repaint() l’objet est redessiné et l’illusion du mouvement est réalisée

Mouvement de l’objet

Mouvement

- Nous avons d’abord choisi d’écrire un code simple permettant de contrôler l’objet.
  - La position de l’objet est modifiée grâce aux setters et getters

Quand le bouton up n'est pas pressé :

```
Objet.setPosX(Objet.getPosX()+1);
// L’objet avance d’1 pixel sur l’axe X
Objet.setPosY(Objet.getPosY()+3) ;
// L’objet avance de 3 pixels sur l’axe Y
```

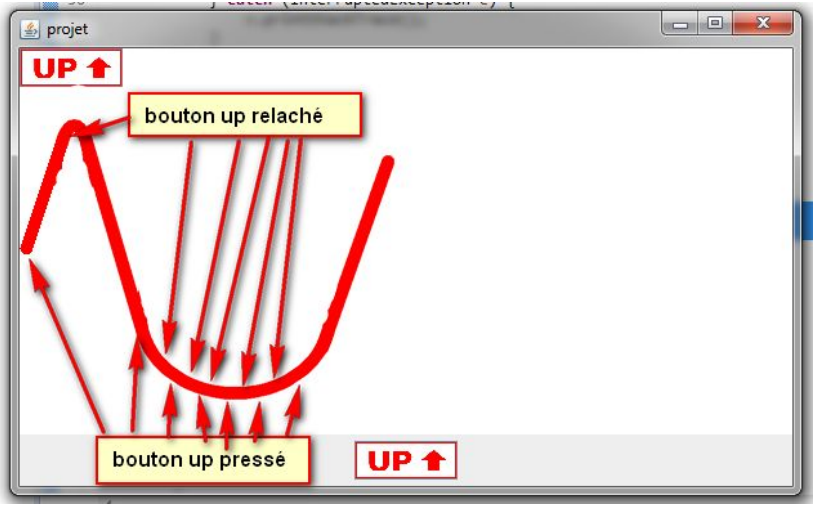
Quand il est pressé :

```
Objet.setPosX(Objet.getPosX()+1);
// L’objet avance d’1 pixel sur l’axe X
Objet.setPosY(Objet.getPosY()-3);
// L’objet recule de 3 pixels sur l’axe Y
```

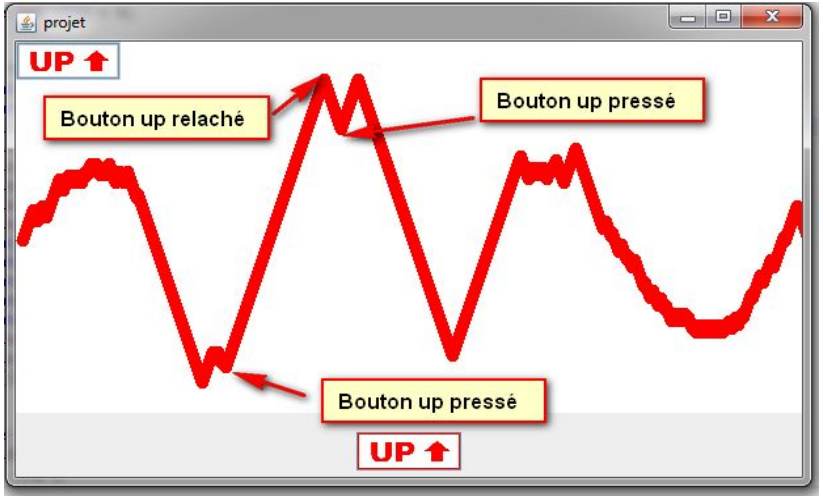
Mouvement de l'objet

Resultats

- Résultat attendu:



- Résultat obtenu:



Mouvement de l’objet

Implementation finale

- Pour donner l’effet d’inertie il fallait prendre en considération l’accélération
  - Creation d’une variable acc

Quand le bouton up n'est pas pressé :

```
acc = acc - 0.15F;
// La variable accélération (acc)
est décroissante exponentiellement
niv.setPosX(niv.getPosX() + 0.7) :
// L’objet avance de 0.7 pixel sur l’axe X
niv.setPosY(niv.getPosY() - acc);
// L’objet avance de acc sur l’axe Y
```

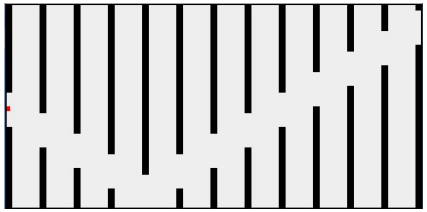
Quand il est pressé :

```
acc = acc + 0.15F;
// La variable accélération (acc)
est croissante exponentiellement
niv.setPosX(niv.getPosX() + 0.7);
// L’objet avance d’un pixel sur l’axe X
niv.setPosY(niv.getPosY() - acc);
// L’objet recule de acc sur l’axe Y
```

Gestion des niveaux

Principe

- Modélisation des obstacles des différents niveaux sous forme de barres



  - Avantage: stocker les niveaux dans un tableau et les générer facilement

- La hauteur de chaque rectangle supérieur est contenue le tableau
  - Ainsi, le rang 1 du tableau contient la hauteur des rectangles du premier niveau etc.
  - Le rectangle inférieur est généré avec un espace de 50 pixels par rapport au supérieur

Gestion des niveaux

Stockage des niveaux

- Stockage des positions des rectangles supérieurs:

```
int tab [][] ={
{130,130,130,130,130,130,130,130,130,130,130,130,130},
//Hauteurs des rectangles supérieurs du niveau 1
{130,150,170,200,150,110,90,80,70,75,90,90,100},//2
{130,100,50,40,100,200,250,250,200,200,100,150,200},//3
{130,200,10,200,100,50,200,250,200,100,50,200,70},//4
{130,10,10,200,100,10,30,250,70,241,50,10,20},//5
{130,250,0,250,0,250,0,250,0,250,0,250,0},//6
};
```

Gestion des niveaux

Génération des niveaux

- Code de génération:

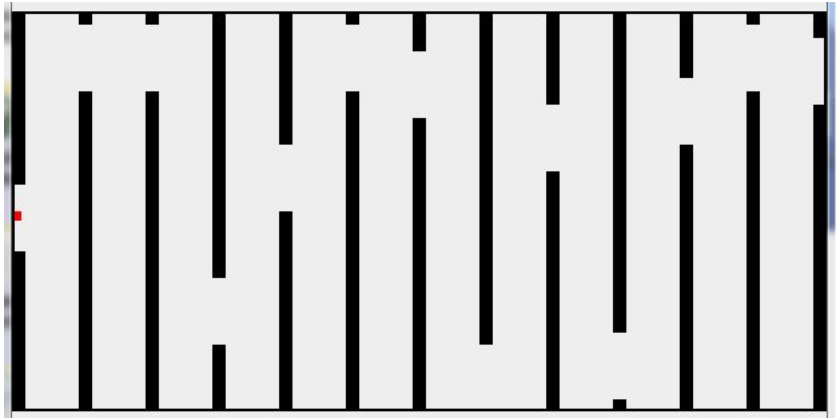
```
public void paintComponent (Graphics g){

        g.setColor(Color.RED);
        g.fillRect((int) posX, (int) posY, 7, 7);
        //Données du carré rouge

        g.setColor(Color.black);
        //Couleur define en noir
        g.fillRect(0,0,10,tab[lvl][0]);
        g.fillRect(0,tab[lvl][0]+50,10,300);
        //Rectangles de premières positions
        g.fillRect(50,0,10,tab[lvl][1]);
        g.fillRect(50,tab[lvl][1]+50,10,300);
        //Deuxièmes

        ..etc
```

- Résultat obtenu après génération:



Gestion des collisions

Principe

- Le joueur perd la partie dès que le carré touche un des obstacles ou sort de l'espace de jeu
  - Il est nécessaire de vérifier à chaque modification de position si il y avait une collision
- Utilisation de la classe Rectangle du package java.awt.
  - Comporte une méthode « intersect » permettant de savoir si deux objets de la classe rectangle entre en collision.
- Création des objet rectangles
  - Autant d'objets rectangles que d'obstacles

Gestion des collisions

Implementation

- Génération des rectangles :
  - La méthode rectangles() est appelée et les rectangles sont créés en fonction de la position des obstacles.

```
public void rectangles() {
        rect01 = new Rectangle(0, 0, 10, tab[lvl][0]);
        //Génération du rectangle supérieur de première position
        rect02 = new Rectangle(0, tab[lvl][0] + 50, 10, 250);
        //Génération du rectangle inférieur de première position
        rect11 = new Rectangle(50, 0, 10, tab[lvl][1]);
        rect12 = new Rectangle(50, tab[lvl][1] + 50, 10, 250);
```

- Gestion de la collision : la méthode Collision() va utiliser la méthode intersects() de la classe rectangle
  - On vérifie si les objets rectangles sont en contact avec le carré.

```
public void Collision() {

    Rectangle rect = new Rectangle((int) posX, (int) posY, 7, 7);
            //Génération de l'objet rectangle de même taille et position
            que le carré
    if ((this.posY > 293 || this.posY < 0 ||
            (rect.intersects(rect01))  ||  (rect.intersects(rect02))  ||
            (rect.intersects(rect11))  ||  (rect.intersects(rect12))  ||
            (rect.intersects(rect21)) || (rect.intersects(rect22)) ||
```

Gestion des joueurs

Principe

- On utilise deux arrayLists pour stocker le pseudo et le niveau du joueur
  - Le pseudo et le score sont liés grâce à la variable “idJoueur”
- Bouton création nouveau joueur
  - Le pseudo et le niveau initialisé à 0 sont ajoutés aux array lists
- Bouton pour démarrer la partie où le joueur s’identifie
  - Vérification si le pseudo est présent dans l’arraylist et récupération du niveau

Sauvegarde Automatique

Principe

- Fichiers “Score” et fichier “Noms”
  - Dans ces fichiers sont sauvegardés les scores et pseudo des joueurs
- Utilisation des classes java.io.FileWriter et java.io.BufferedWriter
  - Grâce à ces classes de la bibliothèque java on peut copier le contenu des arrayLists vers les fichiers “Scores” et “Noms”
- Sauvegarde automatique grâce aux méthodes sauvegardeScore() et sauvegardeNoms()
  - On exécute ces méthodes, dès que le joueur termine un niveau

Recuperation des scores

Principe

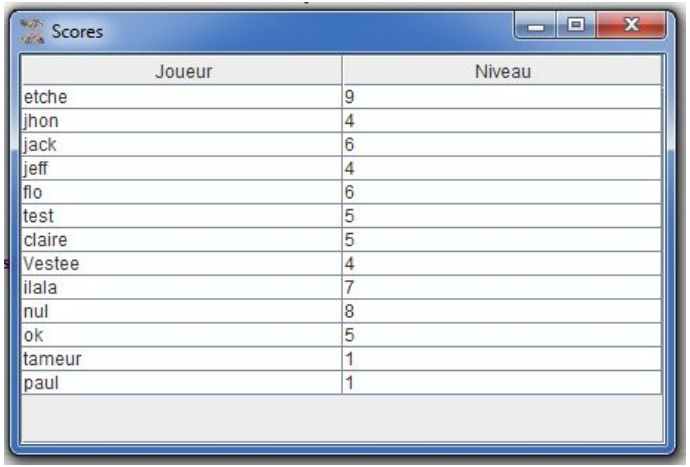
- Utilisation des classes java.io.FileReader et java.io.BufferedReader
  - Grâce à ces classes de la bibliothèque java on peut copier le contenu des fichiers “Scores” et “Noms” vers les arrayLists

- Récupération automatique grâce à une méthode lectureScore() et lectureNoms()
  - On execute ces methodes au demarrage de gravity

Tableau des scores

On doit pouvoir afficher un tableau des scores.

- Utilisation des classes JTable et JScrollPane
  - Permet d'afficher dans une fenêtre les données contenues des ArrayLists « Noms » et « Scores »
- Accès au tableau des scores sur la fenêtre d'accueil



Lancement de l'animation

Utilisation d'un thread

- Utilisation d'un nouveau thread pour exécuter le code de l'animation
  - On exécute la méthode go() dans un nouveau thread afin de pouvoir d'exécuter les tâches simultanément

```
class Play implements Runnable {
	public void run() {
		Game game = new Game();
		//création d'une fenêtre Game qui affiche le niveau
		game.go();
		//appel la méthode qui lance l'animation
	}

}
```

```
t = new Thread(new Play());
		// Création d'un nouveau thread
		t.start();
		// Lancement du thread
```

4. Evolutions possibles

Modifications futures

Comment améliorer notre application ?

Optimisation
Amélioration des algorithmes, Suppression de certaines ressources.

Aide plus explicite
Création d'un tutoriel pour les joueur qui n'ayant jamais joué.

Graphisme et son
Carré rouge remplacé par un asteroid. Ajouter des sons lors des crashs.

Autres modes de jeu
Ex : Un mode permettant de tenter de passer le plus de niveau sans perdre.

Fichier de sauvegarde crypté ou caché
Pour plus de sécurité

Difficulté progressive
Revoir les niveaux de difficulté en fonction des niveaux : vitesse de jeux, emplacement des obstacles ..

5. Video de démonstration


Bienvenue dans
GRAVITY
V1.01
Jouer
Nouveau joueur
Aide
Tableau des scores
Credit

BILAN DU PROJET