Utilisation du moteur de jeu en JAVA.

vincent.thomas@loria.fr

L'objectif des classes fournies est de proposer un moteur simple pour faire tourner des jeux ou des simulations en java.

Récupération des fichiers

Le fichier « **`moteurJeu.zip`** » contient les fichiers java à la base du moteur de jeu. Ces fichiers sont organisés en package

- le package **`moteur`** contient les classes permettant de gérer l'affichage et d'exécuter le jeu
- le package **`sprite`** contient les classes pour charger et dessiner des sprites.

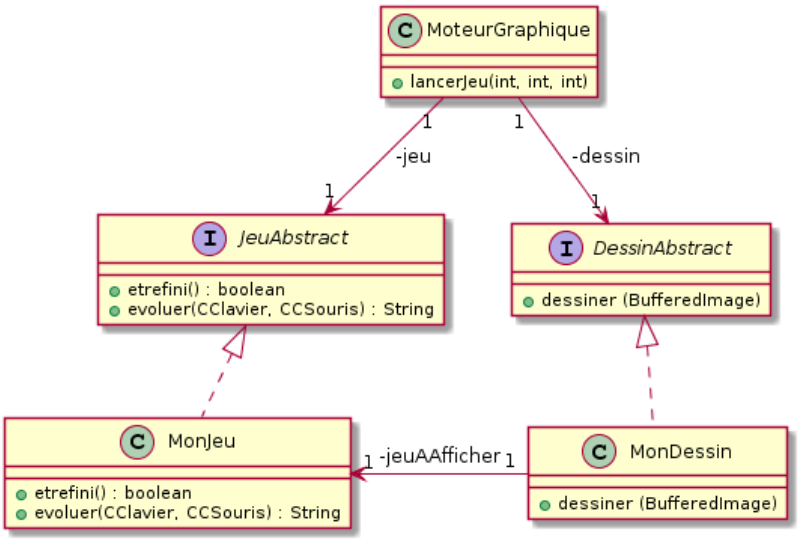
Package Moteur

Le package moteur contient les classes permettant de lancer une application.

La classe fondamentale est la classe **`MoteurGraphique`** qui nécessite deux objets:

- un objet **`jeu`** de de type **`JeuAsbtract`** censé contenir les règles et les lois d'évolution du jeu ;
- un objet **`dessin`** de type **`DessinAbstract`** censé contenir la manière d'afficher le jeu à l'écran.

Proposer un jeu consiste donc à définir deux classes implémentant l'interface **`JeuAbstract`** et l'interface **`DessinAbstract`** (ici **`MonJeu`** et **`MonDessin`**).



La classe **`MoteurGraphique`** permet alors de faire tourner une boucle de jeu régulière à partir des règles du jeu et de la classe d'affichage.

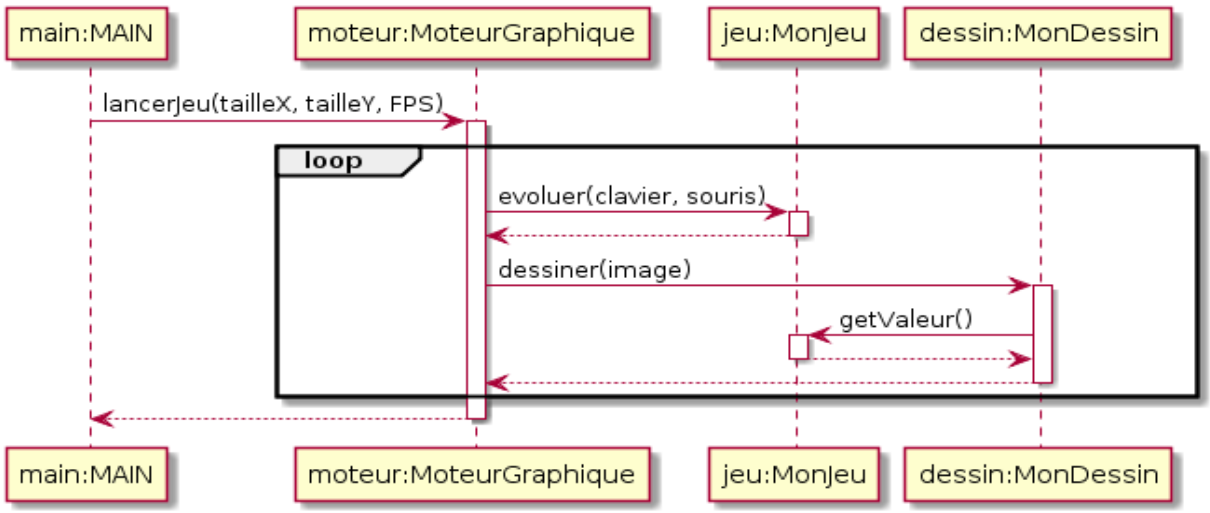
Pour cela, on crée un objet **`moteur`** de type **`MoteurGraphique`** en passant en paramètre l'objet

jeu et le **dessin** associé. On appelle la méthode **lancerJeu** du moteur en passant en paramètre la taille de la fenêtre ainsi que le nombre de frames par secondes.

Le **moteur** à son tour lance la boucle de jeu. Cela consiste à appeler de manière cyclique :

- la méthode **evoluer** du **jeu** possédé en attribut ;
- puis la méthode **dessiner** du **Dessin** possédé en attribut, le **dessin** utilise le **jeu** pour savoir quelle information afficher et quoi en faire.

Le diagramme de séquence ci-dessous représente de manière graphique ce que fait la boucle de jeu à chaque itération.



Mise en place d'un jeu simple

On va partir sur un "jeu" très simple avec un personnage déplaçable en appuyant sur les touches du clavier (dans le package **exemple.exempleSimple**).

Classe Jeu

Proposer un jeu consiste tout d'abord à définir la classe **MonJeu** implémentant l'interface **JeuAbstract**. Cette classe représente les règles du jeu.

Il est nécessaire pour cela de définir les deux méthodes

- la méthode `public String evoluer(CClavier clavier, CSouris souris);` qui contient le code à executer lorsque la boucle de jeu demande au jeu de se mettre à jour ;
- la méthode `public boolean etreFini();` qui retourne un booléen valant vrai si et seulement le jeu doit s'arrêter (fin du jeu, game over, sortie utilisateur). Ce booléen met alors fin à la boucle de jeu.

Une première version vide est donc la classe suivante :

```
public class MonJeu implements JeuAbstract{

	// Definit l'evolution du jeu
	public String evoluer(CClavier clavier, CSouris souris) {
		// TODO A completer
		return null;
```

```
	}

	// Definit la fin du jeu
	public boolean etreFini() {
		// TODO A completer
		return false;
	}

}
```

Pour gérer la fin du jeu, on va simplement ajouter un **`boolean fini`** en attribut au jeu. La valeur de ce booléen pourra changer au cours de l'évolution du jeu et marquera la fin du jeu.

```
public class MonJeu implements JeuAbstract{

	//fin du jeu
	boolean fini = false;

	// Definit l'evolution du jeu
	public String evoluer(CClavier clavier, CSouris souris) {
		// TODO Auto-generated method stub
		return null;
	}

	// Definit la fin du jeu
	public boolean etreFini() {
		return fini;
	}

}
```

Pour la méthode **`evoluer`**, on va

- représenter un personnage par deux entiers x et y ;
- mettre à jour les coordonnées du personnage en fonction des touches appuyées ;
- passer le **`boolean fini`** à vrai si on appuie sur la touche escape.

Pour accéder aux actions utilisateurs via le clavier, il suffit d'utiliser l'objet **`clavier`** passé à la méthode **`evoluer`**. L'objet **`clavier`** possède en effet les méthodes suivantes

- `public boolean isPressed(int charCode)` qui retourne vrai si une touche a été pressée depuis l'itération précédente ;
- `public boolean getTyped(int charCode)` qui retourne vrai si une touche a été tapée (appuyée puis relâchée) depuis l'itération précédente.

Les entiers code de caractères à passer en paramètre sont des constantes définies dans la classe **`KeyEvent`** (exemple **`KeyEvent.VK_LEFT`** pour la flèche pointant vers le haut). Il suffit alors de tester la valeur retournée par ces méthodes en passant en paramètre les touches que l'on souhaite tester.

La méthode **`evoluer`** devient donc simplement:

```
	// Definit l'evolution du jeu
	public String evoluer(CClavier clavier, CSouris souris) {

		//si la touche gauche est appuyee
		if (clavier.isPressed(KeyEvent.VK_LEFT))
```

```
                this.x=this.x-1;

            //si la touche droite est appuyee
            if (clavier.isPressed(KeyEvent.VK_RIGHT))
                this.x=this.x+1;

            //si la touche haut est appuyee
            if (clavier.isPressed(KeyEvent.VK_DOWN))
                this.y=this.y+1;

            //si la touche haut est appuyee
            if (clavier.isPressed(KeyEvent.VK_UP))
                this.y=this.y-1;

            //si la touche escape est appuyee
            if (clavier.isPressed(KeyEvent.VK_ESCAPE))
                this.fini=true;

            return null;
      }
```

Il est aussi possible d'accéder aux coordonnées et aux clicks de la souris via l'objet **souris** à travers :

- la méthode `public boolean getClicked()` qui retourne vrai si il y a eu un click de souris depuis la dernière itération ;
- la méthode `public boolean isPressed()` qui retourne vrai si un bouton de la souris est appuyé (il est possible de determiner le bouton appuyé en modifiant la classe **CSouris**) ;
- les mèthodes `int getX()` et `int getY()` qui retournent les coordonnées de la souris.

Classe Dessin

Une fois le jeu créé, il est nécessaire de construire la classe d'affichage implémentant l'interface **DessinAbstract**. Cette classe a pour objectif d'afficher le rendu dans l'image passée en paramètre à chaque itération de la boucle de jeu.

```
public class MonDessin implements DessinAbstract{

      // permet de dessiner la nouvelle image du jeu
      public void dessiner(BufferedImage image) {
            // TODO A completer
      }
}
```

Pour effectuer l'affichage, il suffit de dessiner dans l'objet **image** passé en paramètre à la méthode **dessiner**. Le dessin peut se faire grâce un objet de type **Graphics2D** construit à partir de l'objet **image** (cf javadoc de classe **Graphics2D**)
⇒ https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html

Par exemple, si on souhaite afficher à chaque itération le même texte à l'écran, il suffit simplement de redéfinir la méthode **dessiner** de la manière suivante

```
      // permet de dessiner la nouvelle image
      public void dessiner(BufferedImage image) {
            // on recupere un crayon sur l'image
            Graphics2D g = (Graphics2D) image.getGraphics();
```

```
            // on dessine sur l'image
            g.setColor(Color.BLUE);
            g.drawString("appuyer <ESC> pour sortir", 200, 300);

            // on rend le crayon
            g.dispose();
      }
```

Cependant, si on veut pouvoir afficher des éléments en lien avec le jeu, il est nécessaire que l'affichage dispose d'un lien vers le jeu à afficher. On ajoute donc un attribut de type **`MonJeu`** dans la classe **`MonDessin`**.

```
      // le jeu
      MonJeu jeu;

      // creer un dessin a partir d'un jeu
      public MonDessin(MonJeu j) {
            this.jeu=j;
      }
```

Maintenant que le dessin connaît le jeu, on peut utiliser les attributs du jeu à afficher.

```
public class MonDessin implements DessinAbstract {

      // le jeu
      MonJeu jeu;

      // creer un dessin a partir d'un jeu
      public MonDessin(MonJeu j) {
            this.jeu = j;
      }

      // permet de dessiner la nouvelle image
      public void dessiner(BufferedImage image) {
            // on recupere un crayon sur l'image
            Graphics2D g = (Graphics2D) image.getGraphics();

            // on dessine sur l'image
            g.setColor(Color.BLUE);
            g.drawString("appuyer <ESC> pour sortir", 200, 300);

            // on dessine le personnage
            int taille = 10;
            int debutX = this.jeu.x - taille;
            int debutY = this.jeu.y - taille;
            g.setColor(Color.PINK);
            g.fillOval(debutX, debutY, 2 * taille, 2 * taille);
            g.setColor(Color.BLACK);
            g.drawOval(debutX, debutY, 2 * taille, 2 * taille);

            // on rend le crayon
            g.dispose();
      }

}
```

Lancement du jeu

Une fois que les classes représentant le jeu et la classe représentant l'affichage ont été décrites, il suffit de créer un **`moteurGraphique`** mettant en relation les éléments

- On crée un objet de type **`MonJeu ;`**
- On crée ensuite un objet de type **`MonDessin`** connaissant le jeu à afficher ;
- On crée un objet **`MoteurGraphique`** en passant le jeu et le dessin en paramètre ;
- Il suffit enfin d'appeler la méthode **`lancerJeu`** sur l'objet **`moteur`** créé en précisant la taille de l'écran (ici 800x600) ainsi que le nombre de frames par seconde souhaité (ici 100).

```
public class Main {

	public static void main(String[] args) {
		// creation du jeu et du dessin
		MonJeu jeu = new MonJeu();
		MonDessin dessin = new MonDessin(jeu);

		// creation du moteur
		MoteurGraphique moteur = new MoteurGraphique(jeu, dessin);
		moteur.lancerJeu(800, 600, 100);

	}
}
```

Exemples fournis

Trois exemples sont fournis dans le fichier zip dans des sous-package du package exemple:

- le package **`exemple.exempleSimple`** contient l'exemple construit dans ce fichiers
- le package **`exemple.physique`** contient un exemple avec une balle qui rebondit
- le package **`exemple.physiqueAvecInteraction`** contient un exemple avec une balle qui saute en fonction de l'action utilisateur.

Chaque exemple possède son propre **`main`** qui lance le jeu correspondant.

Exemple exempleSimple

Lancer l'exemple simple se fait via la commande

```
java exemple.exempleSimple.Main
```

La classe de jeu et la classe de Dessin sont celles présentées dans ce document

- le jeu modifie simplement les coordonnées x et y d'un personnage en fonction des touches appuyées.
- Le dessin affiche un texte constant et le personnage sous la forme d'un cercle.

Exemple physique

Lancer cet exemple se fait via la commande

```
java exemple.physique.Main
```

La classe **JeuPhysique** intégre un moteur simple basé sur la dynamique du point avec des rebonds et une gravité constante.

La classe **DessinPhysique** dessine simplement la balle qui rebondit (en inversant l'axe des ordonnées).

Exemple physiqueAvecInteraction

Lancer cet exemple se fait via la commande

```
java exemple.physiqueAvecInteraction.Main
```

La classe **JeuPhysiqueInteraction** fonctionne de la même manière que la classe précédente, mais arrête la balle selon l'axe vertical lorsqu'elle atteint le sol ($y<0$) et donne une vitesse $vy>0$ lorsque l'utilisateur appuie sur la touche espace.

La classe **DessinPhysiqueInteraction** fonctionne de la même manière que précédemment mais affiche en plus un texte et les traits verticaux et horizontaux représentant la surface dans laquelle la balle évolue.