Interface utilisateur de bas niveau

La classe Canvas

- Canvas : cette classe permet d'accéder aux éléments de bas niveau et donc d'effectuer un contrôle plus fin sur les divers éléments ou événements. Les jeux utilisent pleinement cette classe.
- Graphics : cette classe permet de réaliser des graphiques 2D. Elle correspond à la classe java.awt.Graphics de J2SE.
- Font : Comme son nom l'indiquer, cette classe définit les polices de caractères.

Canvas

- Il s'agit d'une classe de base lorsqu'on souhaite intervenir graphiquement à un niveau assez fin (pour les jeux notamment). Pour tracer quelque chose sur l'écran, il faut dans la classe Canvas implémenter la méthode paint(Graphics gg).
- Evénements:
  - protected void keyRepeated(int keyCode)
  - protected void keyPressed(int keyCode)
  - protected void keyReleased(int keyCode)
  - getGameAction(int kcode): retourne une constante définie dans la classe Canvas (UP, DOWN, RIGHT,LEFT, FIRE, GAME_A …_D,GAME_NUM0 … 9, GAME_POUND, GAME_STAR

Graphics: les tracés

- drawChar(char caractère, int x, int y, int ancre) : tracé d'un caractère à un endroit précis.
- drawString(String str, int x, int y, int ancre) : tracé d'une chaîne à un endroit précis.
- drawImage(Image img, int x, int y, int ancre) : tracé d'une image à un endroit précis.
  - L'ancre définit la position de l'image / au point, valeurs possibles:
    - Graphics.HCENTER | Graphics.TOP
    - Valeurs verticales: TOP, BASELINE, et BOTTOM
    - Valeurs horizontales: LEFT, HCENTER, et RIGHT.
- drawLine(int x1, int y1, int x2, int y2) : tracé d'une ligne définies par ses extrémités.
- drawRect(int x, int y, int width, int height) : tracé d'un rectangle
- drawArc(int x, int y, int width, int height, int startAngle, int arcAngle) : tracé d'un arc circulaire ou elliptique inscrit dans un rectangle
- setStrokeStyle(int style) : définition du type de trait

Graphics: coloriage

- setColor(int red, int green, int blue) : définition d'une couleur RGB.
- fillRect(int x, int y, int width, int height) : remplissage d'un rectangle dans la couleur courante.
- fillArc(int x, int y, int width, int height, int startAngle, int arcAngle) : remplissage d'un secteur dans la couleur courante.
- fillTriangle(int x1, int y1, int x2, int y2, int x3, int y3) : remplissage d'un triangle dans la couleur courante

informations :

- getColor() : obtention de la couleur courante
- getFont() : obtention de la police courante
- getStrokeStyle() : obtention du style de trait

Classe Font

- 3 types de polices
  - FACE_SYSTEM: police utilisée par le terminal
  - FACE_MONOSPACE
  - FACE_PROPORTIONAL.
- 3 tailles: SIZE_SMALL, SIZE_MEDIUM, et SIZE_LARGE
- 3 styles: STYLE_PLAIN, STYLE_BOLD, STYLE_ITALIC qui peuvent être combinés avec STYLE_UNDERLINED
- Font f=Font.getFont(type, style, taille)
  - Font f = Font.getFont(Font.FACE_PROPORTIONAL, Font.SIZE_LARGE, Font.BOLD);
- g.setFont(f);

Image

- public static Image createImage(Image image, int x, int y, int width, int height, int transform)
  - On choisit une zone rectangulaire de l'image qui sera extraite
  - Trasform:
    - TRANS_NONE
    - TRANS_ROT90
    - TRANS_ROT180
    - TRANS_ROT270
    - TRANS_MIRROR
    - TRANS_MIRROR_ROT90, TRANS_MIRROR_ROT180, et TRANS_MIRROR_ ROT270

Exemple (1/3)

```
package dessin;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Graphics;
import javax.microedition.midlet.MIDlet;

public class Dessin1 extends MIDlet {

    Canvas monCanvas;

    public Dessin1() {
        monCanvas = new Jeu();
    }

    public void startApp() {
        Display display = Display.getDisplay(this);
        display.setCurrent(monCanvas);
        monCanvas.repaint();
    }
```

Exemple (2/3)

```
public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}

class monCanvas extends Canvas {
    // initialisation

    int largeur = getWidth();
    int hauteur = getHeight();
    private int x = largeur / 2 - 10;
    private int y = hauteur / 2 - 10;
    private int dx = 5;
    private int dy = 5;

    public void paint(Graphics g) {
        g.setColor(0xffffff);
        g.fillRect(0, 0, getWidth(), getHeight());

        g.setColor(0xff0000);
        //  g.fillRect(x,y,20,20);
        g.fillArc(x, y, 20, 20, 0, 360);
    }
```

Exemple (3/3)

```
public void keyPressed(int keyCode) {
        int gameAction = getGameAction(keyCode);
        if (gameAction == RIGHT) {
            x += dx;
            if (x > largeur) {
                x = 0;
            }
        } else if (gameAction == LEFT) {
            x -= dx;
            if (x < 0) {
                x = largeur;
            }
        } else if (gameAction == UP) {
            y -= dy;
            if (y < 0) {
                y = hauteur;
            }
        } else if (gameAction == DOWN) {
            y += dy;
            if (y > hauteur) {
                y = 0;
            }
        }
        repaint();
    }
}
```

Utiliser un Timer (1/3)

```
package hello;
/**
*
* @author pc
*/
import java.util.*;
import javax.microedition.lcdui.*;
/* Incovénients de cette implémentation: la boucle principale du jeu est dans un thread
* la classe TimerTask, alors que les événements sont gérés dans le thread pricipal.
*/
public class Jeu
extends Canvas {
private Timer timer;
// méthode appelée au moment où le canvas devient visible dans l'écran du terminal
// initialisation
int largeur = getWidth();
int hauteur = getHeight();
private int x = largeur/2 - 10;
private int y = hauteur/2 - 10;
private int dx = 5;
private int dy = 5;
int sensX=1;
int sensY=1;
int hMin=5;
int hMax=120;
int D=20 ;
int pas=5;
int sens=1;
```

Utiliser un Timer (2/3)

```
public void paint(Graphics g)
{
g.setColor(0xffffff);
g.fillRect(0,0,getWidth(),getHeight());
g.setColor(0xff0000);
// g.fillRect(x,y,20,20);
g.fillArc(x, y, D, D, 0, 360);
}
public Jeu()
{
}
public void afficher() {
timer = new Timer();
TimerTask task = new TimerTask() {
public void run() {
miseAJourEtatJeu();
```

- }
- };
- timer.schedule(task, 0, 20);
- }/
- /Méthode appelée automatiquement si le canvas n'est plus affiché sur l'écran.
- public void masquer() {
- if (timer != null) {
- timer.cancel();
- timer = null;
- }}

Utiliser un Timer (3/3)

```
private void miseAJourEtatJeu() {
// Déplacements des objets, détection des collisions.
x+=dx * sensX;
if (x<10 || x > largeur -10)
{
sensX = -sensX;
x+=dx * sensX;
}
y+=dy*sensY;
if (y<10 || y > hauteur -10)
{sensY=-sensY;
y+=dy*sensY;
}
D+=pas * sens;
if (D <hMin || D > hMax)
sens =-sens;
repaint();
}
public void keyPressed(int kCode) {
// Gestion des événements.
}}
```