Desenvolupament de jocs

Autor: Sergi Tur Badenas

LCDUI. Baix nivell

- **MIDP 1.0**
  - Falta de suport per a aritmètica de punt flotant fa molt complicat implementar jocs en 3 dimensions.
  - No hi ha suport per a àudio. Només suporta fer "bips"
  - No és possible llegir o escriure en un píxel concret fet que no permet fer manipulacions bàsiques de imatges.
  - La classe Graphics no suporta cap tipus de transparència
  - L'únic protocol suportat és HTTP.
- **MIDP 2.0:**
  - Proveeix de l'api **javax.microedition.lcdui.game**

Autor: Sergi Tur Badenas

MIDP 2.0

- **L'API de jocs proveeix de 5 classes:**
  - **GameCanvas**: subclasse de Canvas que proveeix de la funcionalitat bàsica per a un joc.
  - **Layer**: és una classe abstracta, pare de les classes *Sprite* i *TiledLayer*. Proporciona el suport per a gràfics per capes.
  - **Sprite**: és una capa animada bàsica que es capaç de mostrar un conjunt de marcs (frames) de forma animada.
  - **TiledLayer**: representa un element visual format per una graella de cel·les que pot ser omplida per amb un conjunt d'imatges Tile (rajola). Permet crear imatges grans sense la necessitat d'un objecte de tipus Imatge.
  - **LayerManager**: simplifica el desenvolupament de jocs automatitzant el procés de representació (rendering).

Autor: Sergi Tur Badenas

Esquelet d'un joc Java ME

- **MIDP 1.0. Normalment és un bucle infinit**
  - Al codi de paint es mostra l'animació bàsica del joc

```
public class MyCanvas extends Canvas implements Runnable {
    public void run() {
      while(true) {
        repaint(); // update the game state
    }

    public void paint(Graphics g) {
      // code for painting
    }

    protected void keyPressed(int keyCode) {
      // respond to key events
    }
 }
```


Autor: Sergi Tur Badenas

GameCanvas

- **GameCanvas és subclasse de l'objecte Canvas**
  - També és una **classe abstracta**.
  - Inclou mètodes que permeten consultar l'estat de les tecles del joc, i el control del flux de les imatges.
  - **getKeyStates()**
    - Control millorat dels esdeveniments de teclat. Substitueix el mètode **keyPressed()**. Permet accedir a l'estat de les tecles en qualsevol moment evitant problemes amb múltiples fils d'execució.
    - Permet detectar múltiples tecles premudes simultàniament.
  - **FlushGraphics()**
    - Control millorat del doble-buffering. S'utilitza **flushGraphics()** en comptes de **repaint()** per tal de millorar el rendiment

Autor: Sergi Tur Badenas

GameCanvas

- **Simplifica la programació de jocs**
  - Tota la funcionalitat del joc es pot controlar en un sol bucle, sota el control d'un únic fil d'execució

```
public class MyCanvas extends GameCanvas implements Runnable {
   public void run() {
      Graphics g = getGraphics();
      while(true) {
         // Actualitzar l'estat del joc
         int k = getKeyStates();
         // controlar els esdeveniments de teclat
         flushGraphics();
      }
   }
}
```


Autor: Sergi Tur Badenas

Capes (layers)

- **Classe abstracta Layer**
  - Representa un element visual d'un joc, com un Sprite o un TiledLayer
  - Té atributs com la posició, la mida, la visibilitat...
  - Les capes es poden superposar, poder ser visibles o invisibles
  - Han d'implementar el mètode **paint(Graphics g)**

Autor: Sergi Tur Badenas

Sprites

- **Sprites (follets)**
  - Jocs 2D típics (“comecocos”) tenen un personatge que es mou sobre un fons gràfic i interactua amb l'entorn.
  - El conjunt de diferents imatges d'aquests personatges són anomenades sprites
  - La classe Sprite representa una imatge gràfica a on s'emmagatzemen totes les possibles imatges d'un sprite. Un sprite es divideix en diverses imatges que anomenen marcs (frames)
  - Tots els marcs d'un sprite es proporcionen en una sola imatge (objecte Image).
  - Els marcs són tots de la mateixa mida

Autor: Sergi Tur Badenas

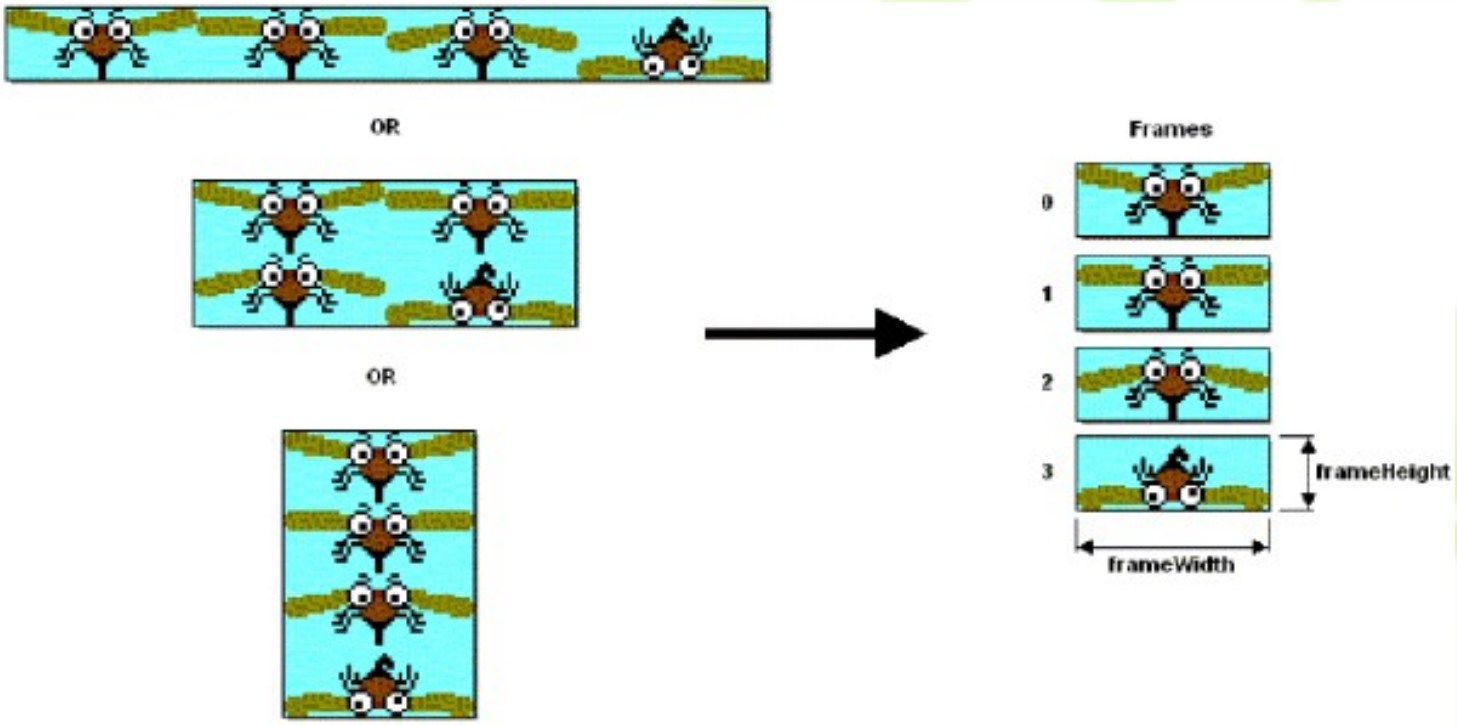
Sprites

- **Les propietats d'un sprite són:**
  - **Posició**: Coordenades cartesianes (x,y) relatives al cantó superior-esquerre de la pantalla.
  - **Seqüencia d'animació**: Es pot definir una seqüència d'imatges o conjunt de marcs de l'sprite amb el mètode **setFrameSequence(int sequence[])**.
  - Avançar un frame: **nextFrame()**
  - **Transformacions simples**: Els sprites es poden rotar, o girar.
    - Les transformacions s'apliquen a partir d'un pixel de referència

Autor: Sergi Tur Badenas

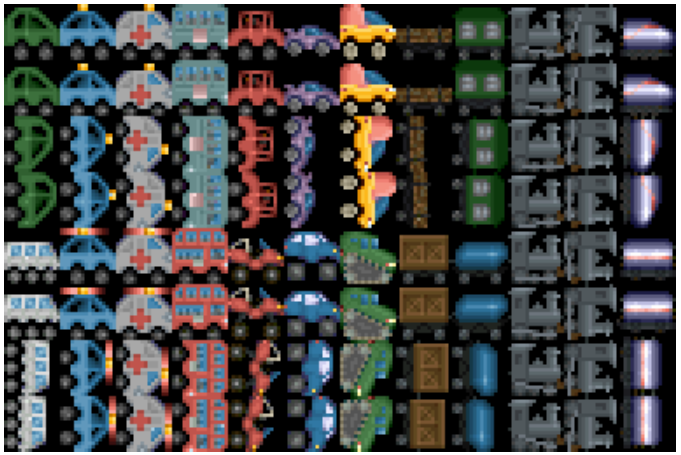
Sprites

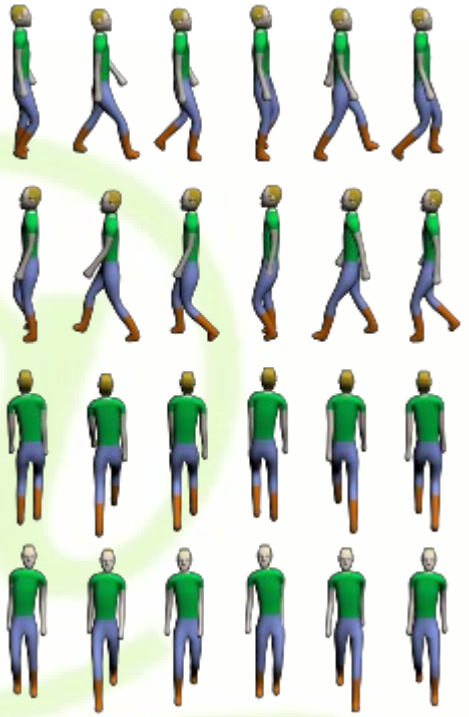
- **Es guarden en una única imatge**
  - La imatge es separa en parts iguals anomenades marcs (frames)



Autor: Sergi Tur Badenas

Sprites

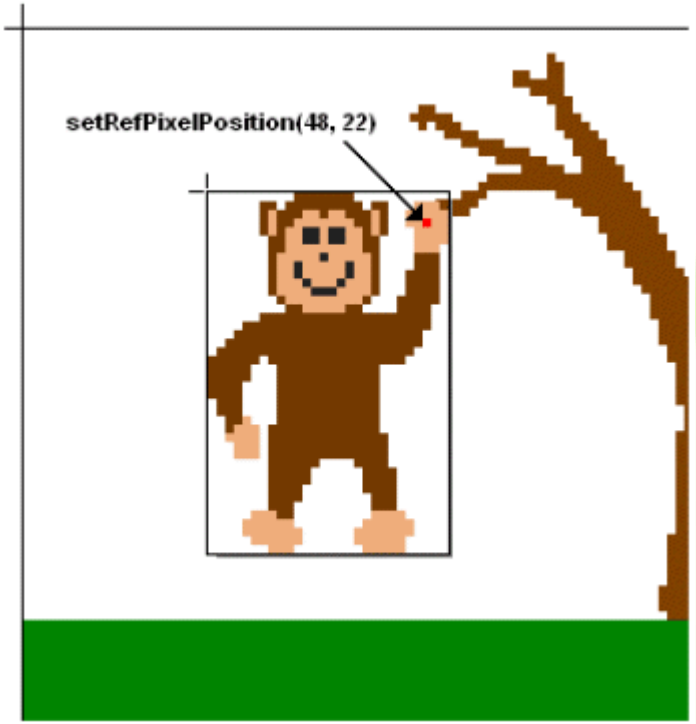


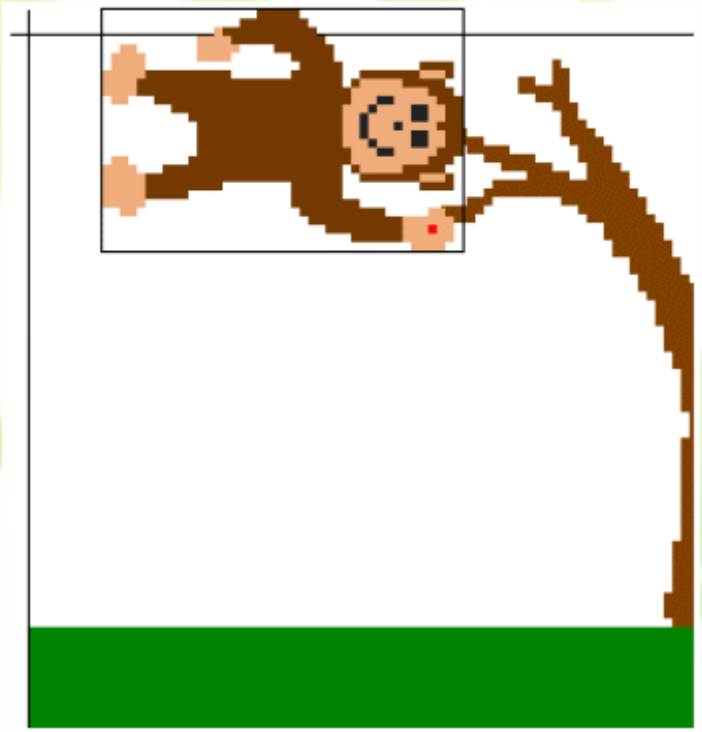


Autor: Sergi Tur Badenas

Sprites

- **Píxel de referència**
  - És la posició a partir de la qual s'apliquen les transformacions





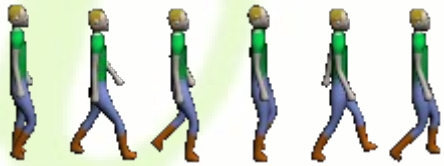
Autor: Sergi Tur Badenas

Sprites

Exemple de GameDesign

```
public int provaseq001Delay = 200;
public int[] provaseq001 = {0, 1, 2, 3, 4, 5};

public Image getMysprite3() throws java.io.IOException {
        if (mysprite3 == null) {
            mysprite3 = Image.createImage("/mysprite3.png");
              }
        return this.mysprite3;
}
public Sprite getPersona() throws java.io.IOException {
              if (persona == null) {
            persona = new Sprite(getMysprite3(), 40, 91);
            persona.setFrameSequence(provaseq001);
            }
        return persona;
}
```



Autor: Sergi Tur Badenas

Sprites

- **Canviar de marc**

```
setFrame(int)
prevFrame()
nextFrame()
```

  - 2 maneres de realitzar l'animació
    - Bucles infinits amb Thread.sleep(delay)
    - TimerTask
  - Tenim un mètode de suport. No cal utilitzar comptadors circulars
    - Els mètodes nextFrame() i prevFrame() són circulars.
  - La seqüència per defecte és 0,1,2,3,4,5....

L'animació no s'aplica sobre l'ordre original dels marcs en la imatge, sinó que s'aplica a la seqüència que heu establert

Autor: Sergi Tur Badenas

Sprites

- **Exemple amb sleep**

```
public void run() {
        Graphics g = getGraphics();
        while (true) {
            // Actualitzar l'estat del joc
            int k = getKeyStates();
                        // controlar els esdeveniments de teclat

            // Posar la pantalla en blanc
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, mWidth, mHeight);
            lm.paint(g, 0, 0);
            PersonaCaminant.nextFrame();
            flushGraphics(0, 0, this.getWidth(), this.getHeight());
            try {
                Thread.sleep(200);
            } catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
    }
```


Autor: Sergi Tur Badenas

Sprites

- **Exemple amb TimerTask**

```
public void run() {
        Graphics g = getGraphics();
        while (true) {
            ....
            g.setColor(255, 255, 255);
            g.fillRect(0, 0, mWidth, mHeight);
            lm.paint(g, 0, 0);
            flushGraphics(0, 0, this.getWidth(), this.getHeight());
        }
    }
private void init() throws IOException {
        gameDesign = new PersonaCaminantGameDesign();
        lm = new LayerManager();
        gameDesign.updateLayerManagerForEscenaPersonaCaminant(lm);
        PersonaCaminant = gameDesign.getPersona();
        TimerTask task = new TimerTask() {
            public void run() {
                PersonaCaminant.nextFrame();
            }};
        Timer timer = new Timer();
        timer.schedule(task, 0, 200);
    }
```


Autor: Sergi Tur Badenas

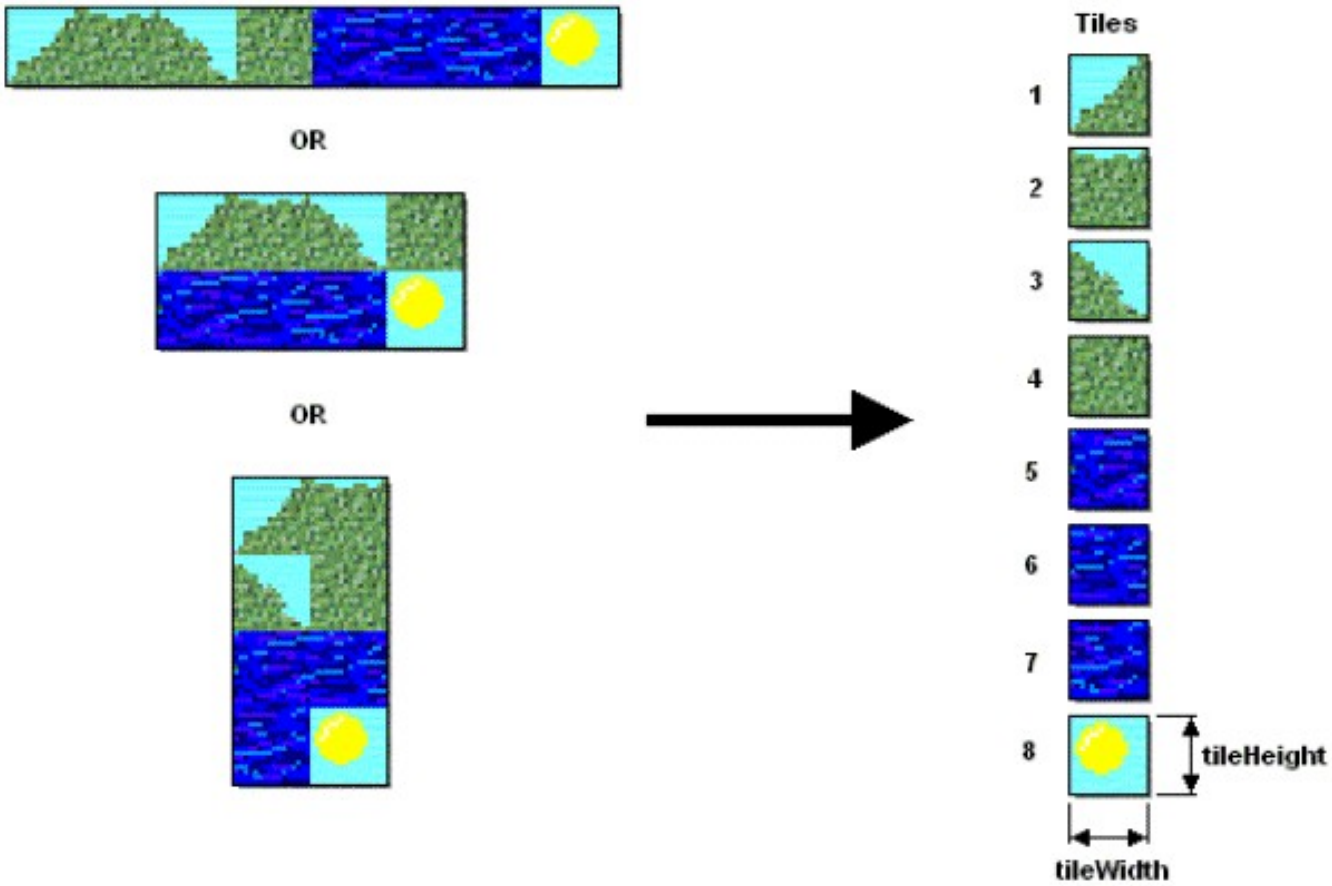
Tiled Layers

- **Tiled layers (capes de rajoles)**
  - Són similars als Sprites però tenen un objectiu diferent.
  - S'utilitzen per a crear fons de jocs.
  - No tenen transformacions, ni seqüències de marcs ni píxels de referència.
  - Una capa de rajoles és una graella formada per un conjunt de marcs. És similar a un terra de ceràmica, format per un conjunt de rajoles.
  - El conjunt de rajoles es guarda en una sola imatge, on tots els Tiles (rajoles) tenen la mateixa mida:

Autor: Sergi Tur Badenas

Tiled Layers

- Es poden guardar el conjunt de rajoles en diferents formes dins d'una imatge.



Autor: Sergi Tur Badenas

Tiled Layers

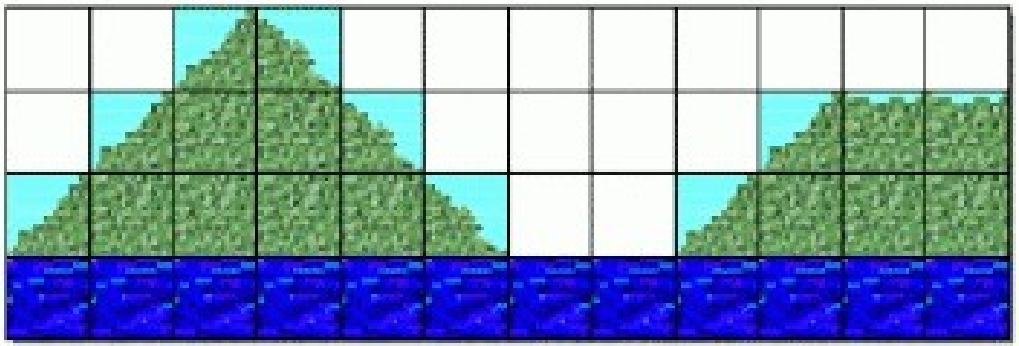
- A cada rajola se li assigna un identificador únic. Per construir un fons de pantalla, s'utilitza una matriu on a cada posició s'indica l'identificador de la rajola a mostrar en aquella posició:

Cells

0	0	1	3	0	0	0	0	0	0	0	0
0	1	4	4	3	0	0	0	0	1	2	2
1	4	4	4	4	3	0	0	1	4	4	4
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Animated Tiles

-1 = 5



Autor: Sergi Tur Badenas

Gestor de capes (LayerManager)

- **Permet gestionar les capes d'un joc**
  - Organitza les diferents capes
  - Permet definir quina és la vista (la zona visible) d'una imatge en cada moment. També anomenat **view port**
  - Podem treballar amb imatges grans i anar "navegant" per la imatge

```
LayerManager.setViewWindow(int x, int y, int width, int height)
```

- **Organització típica**
  - **Fons (background):** TiledLayers
  - **Objectes sobre el fons:** Sprites
  - L'ordre en que es pinten les capes és l'invers de l'ordre en que s'afegeixen les capes al gestor de capes

Autor: Sergi Tur Badenas

Com s'executa un joc (GameCanvas)

- **Recordeu l'esquelet d'un GameCanvas**
  - És un fil d'execució...

```
public class MyCanvas extends GameCanvas implements Runnable {
    public void run() {
        Graphics g = getGraphics();
        while(true) {
            // Actualitzar l'estat del joc
            int k = getKeyStates();
            // controlar els esdeveniments de teclat
            flushGraphics();
        }
    }
}
```


Autor: Sergi Tur Badenas

Com s'executa un joc (GameCanvas)

```
public class GameMidlet extends MIDlet {

    private DemoGameCanvas gameCanvas;
    private Thread t;
    private Display d;

    public void startApp() {
        this.gameCanvas = new DemoGameCanvas();
        this.t = new Thread(gameCanvas);
        t.start();
        d = Display.getDisplay(this);
        d.setCurrent(gameCanvas);
    }
    public void pauseApp() {     }

    public void destroyApp(boolean unconditional) {
        this.gameCanvas.stop();
    }
}
```


Autor: Sergi Tur Badenas

Controlar la velocitat d'execució

```
public class MyCanvas extends GameCanvas implements Runnable {
    public void run() {
       Graphics g = getGraphics();
       while(true) {
          // Actualitzar l'estat del joc
          int k = getKeyStates();
          // controlar els esdeveniments de teclat
          FlushGraphics();
          try {
                Thread.sleep(20);   // 20 ms entre refrescos de pantalla
          } catch (InterruptedException ex) {
                ex.printStackTrace();
          }
       }
    }
 }
```

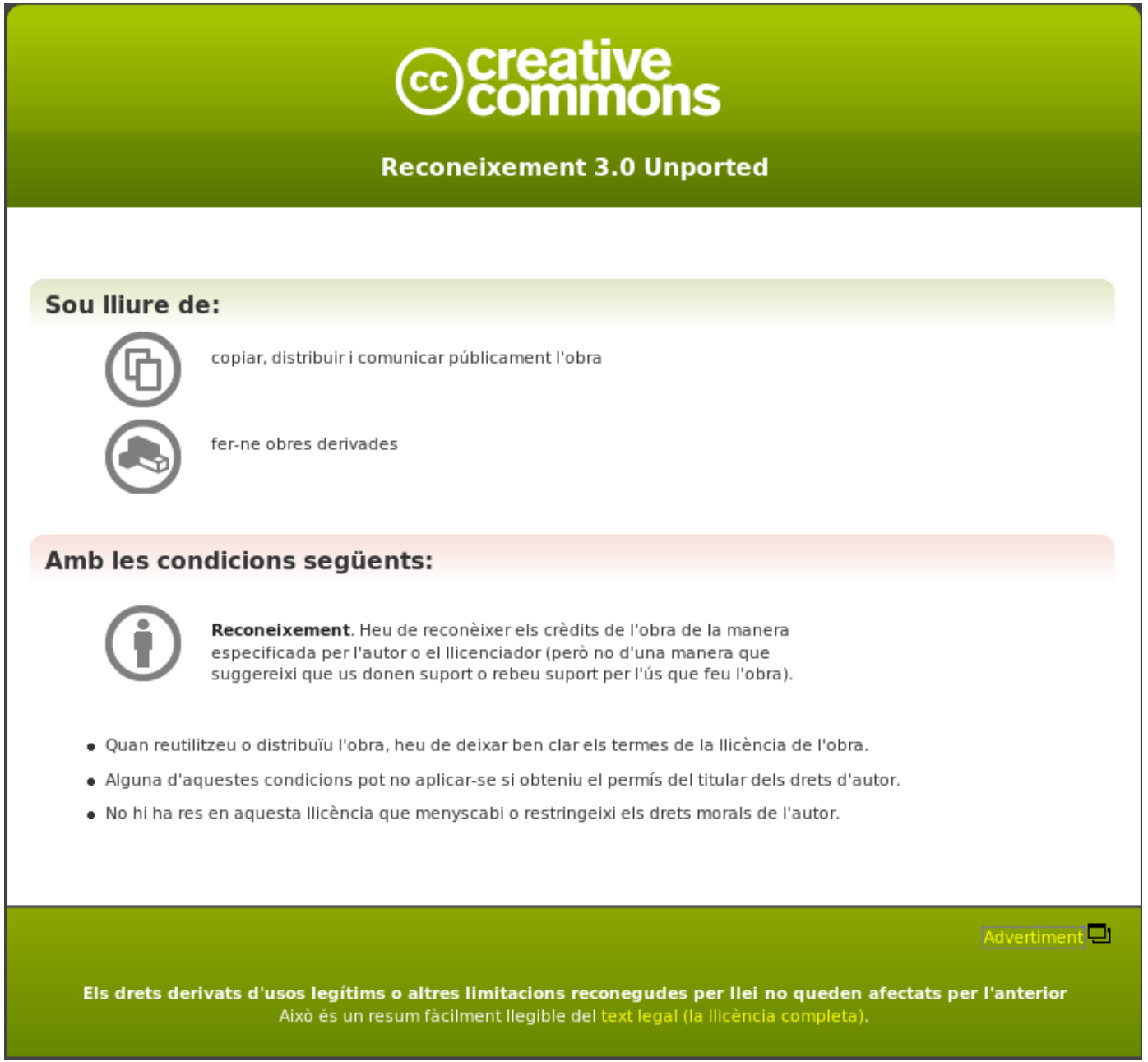

Autor: Sergi Tur Badenas

Netbeans

- **Game Builder**
  - Hi ha un wizard per a generar els elements gràfics d'un joc
    - Ctrl+N -> Java ME -> Game Builder
- **Samples**
  - Samples -> Mobile -> MIDP 2.0 Samples -> Simple Game created with Game Builder
  - Wiki de Netbeans

Autor: Sergi Tur Badenas

Llicència Creative Commons



http://creativecommons.org/licenses/by/3.0/deed.ca

Autor: Sergi Tur Badenas