Game Design and Development for Mobile Devices

Lab 2

Objective:

Develop a Java ME game using Java ME and the wireless toolkit. You will understand the APIs better and develop a simple mobile game. The game is a single player game where a car moves through an obstacle course.

Prerequisites:

J2ME Wireless Toolkit
J2SE SDK

Exercise 1:

Building the game by putting together the source code, the first thing needed is to develop the HardDriveMIDlet that extends the javax.microedition.midlet.MIDlet class.

http://www.developer.com/img/articles/2005/05/05/HardDriveMIDlet.java

This class implements the CommandListener interface to receive command events generated during application execution and then process them. Review HardDriveMIDlet and become comfortable with the listener. The command events occur when commands such as EXIT, CANCEL, BACK, OK, STOP are issued by using the buttons near the mobile phone's screen (called softbuttons) other than arrow keys and are then handled by the commandAction() method of the HardDriveMIDlet.

This class contains three other important methods, which are lifecycle methods. They are startApp(), pauseApp(), and destroyApp(). These methods are for when the applet is started and active, paused/inactive, and when it is finished/closed.

HardDriveCanvas

This canvas extends javax.microedition.lcdui.GameCanvas. This GameCanvas is a special canvas meant for drawing efficient animated graphics for the game app and is also capable of querying key status off-screen graphics buffering for smooth animation.

You will notice HardDriveCanvas implements Runnable, this enables it to run in its own thread. This is necessary to run the game loop independently. The game loop runs until the conditions that are necessary to end the game are met.

HardDriveCanvas.java

```
import java.io.IOException;
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

class HardDriveCanvas
extends GameCanvas
implements Runnable
{
    private HardDriveMIDlet midlet;
    private Sprite carSprite;

    private LayerManager layerManager;
    private ObstacleManager obsManager;


    private boolean gameRunning;
    private boolean collision = false;

    private int width;
    private int height;

    private long gameDuration;

    public HardDriveCanvas(HardDriveMIDlet hdmidlet, String carImageName, String
obsImageName)
    throws IOException
    {
        super(true);

        this.midlet = hdmidlet;  //used later

        layerManager = new LayerManager();

        width = getWidth();
        height = getHeight();

        layerManager.setViewWindow(1,1,width - 2, height -2);

        createCar(carImageName);

        obsManager = new ObstacleManager( this, layerManager, obsImageName);

    }
```

```
private void createCar(String carImageName) throws IOException
{
    // Make an instance of the car Image
    // Instantiate a new Sprite using the car image
    // Set the sprite's position to the bottom center of the screen
    // Add the car sprite to the layer manager

}

public void start()
{
    gameRunning = true;
    //Create a new Thread for the game and start it.
}

public void stop() { gameRunning = false; }

public void run()
{
    Graphics g = getGraphics();
    int timeStep = 300;
    obsManager.renderObstacles();
    long startTime = System.currentTimeMillis();
    while (gameRunning) //is true
    {
            tick();
            input();
            render(g);

            long endTime = System.currentTimeMillis();
            long duration = (int)(endTime - startTime);
            gameDuration = duration / 1000;  //game time in seconds

            try
            {
                Thread.sleep(timeStep );//- duration);
                obsManager.MoveObstacles();
            }
            catch (InterruptedException ie) { stop(); }


    }

}
```

```
    private void tick()
    {
        if(!collision)
                checkCollision();
        if (collision)
        {
                //Game over
                int score = obsManager.getScore();
                midlet.HardDriveCanvasGameOver(gameDuration,score);
                //stop();
        }
    }

    private void input()
    {
        int keyStates = getKeyStates();
        int currentY = carSprite.getY();
        int currentX = carSprite.getX( );
        if ((keyStates & LEFT_PRESSED) != 0)
        carSprite.setPosition (Math.max(0, currentX-5), currentY);
        if ((keyStates & RIGHT_PRESSED) != 0)
        carSprite.setPosition(Math.min(170, currentX + 5), currentY);

    }
    private void render(Graphics g)
    {
        g.setColor(255, 255, 255); // white
        g.fillRect(0, 0,getWidth()-3, getHeight()-3);

        layerManager.paint(g,0,0);

        flushGraphics();
    }

    private void checkCollision( )
    {
        if (obsManager.hitTest(carSprite) )
                collision = true;
    }
}
```

Notice the game loop has calls to method in an order: tick(), input(), and render(). Tick() checks to see if conditions to stop the game are met. The input() method handles game key (keys assigned to perform actions, like movement) inputs and performs the associated actions. Rendering according to the situation or state is done with render().

The LayerManager that is used in HardDriveCanvas adds and manages multiple layers of that HardDriveCanvas, each layer representing a sprite. A sprite is a basic visual element of a game (a character, obstacles, etc).

Exercise 2:

In this example, HardDriveCanvas instantiates an object of the ObstacleManager class, which renders and moves obstacle sprites on the game canvas as well as checks to see if they've collided with the car.

http://www.developer.com/img/articles/2005/05/05/ObstacleManager.java

Review the code above and add it your existing project.

http://www.developer.com/img/articles/2005/05/05/GameOverCanvas.java

Above is the game over canvas for this project, attempt to create your own variation of a game over screen and add it to the project.

Exercise 3 - Compile the game:

The wireless toolkit provides a KToolbar, a useful tool for compiling, verifying, packing and testing the mobile applications. Since the game code is ready, it needs to be organized in the following directory structure (provided by KToolbar).

To achieve this, start KToolbar and create a new project, name is HardDriveGame. This will create a directory structure within the Wireless Toolkit's installation folder

```
HardDriveGame   (game project name, user-defined)
|___src
|
|___bin
|
|___classes
|
|___res
|
|___lib
|
|___tmpclasses
|
|___tmplib
```

Copy all your .java files into the src folder and the car.png and obstacle.png files into the res folder.
Open the project button, choose HardDriveGame, then click Build.
After the build is complete, it can be run in the Emulator