Lab Exercise 5

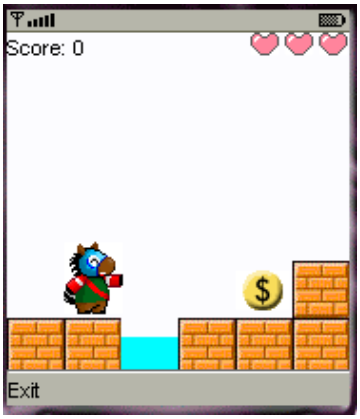
Please follow the instruction in Workshop Note 5 to complete this exercise.

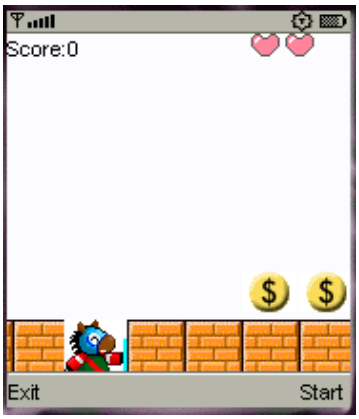
1. Turn on your computer and startup **Windows XP** (*English version*), download and install **J2ME Wireless Toolkit 2.2** to your computer. Then download the laboratory resource files from http://www.peter-lo.com/Teaching/I123-1-A/Source5.zip.

2. Create your simple animation by using several pictures in a sequential order. This is the first program that you need to use thread. (*Page 1 – 3*)

3. Use "***keyPressed***" to capture the numerical keys and game keys. (*Page 4 – 7*)

4. It's time for you to design a simple game, please follow the instruction to complete (*Page 8 – 40*).





5. If you still have time, let's design your own action game.

1. Display Animation on your mobile

1. Create a new project, name the project and the class at "***MyDrawing***" and "***MyClass***".

2. Name the pictures in a sequential order. The put the images into the "***res***" folder.

img0.png img1.png img2.png img3.png img4.png img5.png img6.png img7.png

3. Then create two java files named "***MyClass.java***" and "***ImageCanvas.java***" in the "***src***" folder.

MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*****************************************************************************/
/* All MIDlet applications must extend the MIDlet class.                     */
/*****************************************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Define the GUI components
  private ImageCanvas MyCanvas;     // Canvas
  private Command cmdExit;          // Exit Button
  private Thread MyThread;          // The Thread

  // Define the no-argument constructor
  public MyClass() {
    // Define the Exit Button
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Define the canvas
    MyCanvas = new ImageCanvas();

    // Define the thread
    MyThread = new Thread(MyCanvas);

    // Start the thread
    MyThread.start();
  }

  // Called by application manager to start the MIDlet
  public void startApp() {
    // Add the Exit button and the Command Listener
    MyCanvas.addCommand(cmdExit);
    MyCanvas.setCommandListener(this);

    // Set the current display of the midlet to the Canvas
    Display.getDisplay(this).setCurrent(MyCanvas);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
```

```
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Image imgGirl[] = new Image[10];   // Define the image array
  private int ImageNum = 0;                  // Image Counter
  private String imageName;                  // Image Name
  private boolean ExitFlag = false;          // The Flag to Exit

  public void run() {
    try {
      for (int i=0 ; i<8; i++) {
        imageName = "/img" + i + ".png";
        imgGirl[i] = Image.createImage(imageName);
      }

     // Continuous to repaint the screen until the ExitFlag = True
      while (!ExitFlag) {
        repaint();          // Call paint() to repaint the screen
        Thread.sleep(100);  // Sleep for 0.1 second
      }
    } catch (Exception e) {
    }
  }

  public void paint(Graphics g) {
    // Clear the screen
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, getWidth(), getHeight());

    // Draw the image on screen
    g.drawImage(imgGirl[ImageNum], 0, 0, g.LEFT|g.TOP);

    // Increment the counter
    ImageNum ++;

    // Reset the counter to 0 if it > 7
    if (ImageNum >= 7) {
       ImageNum = 0;
    }
  }
}
```

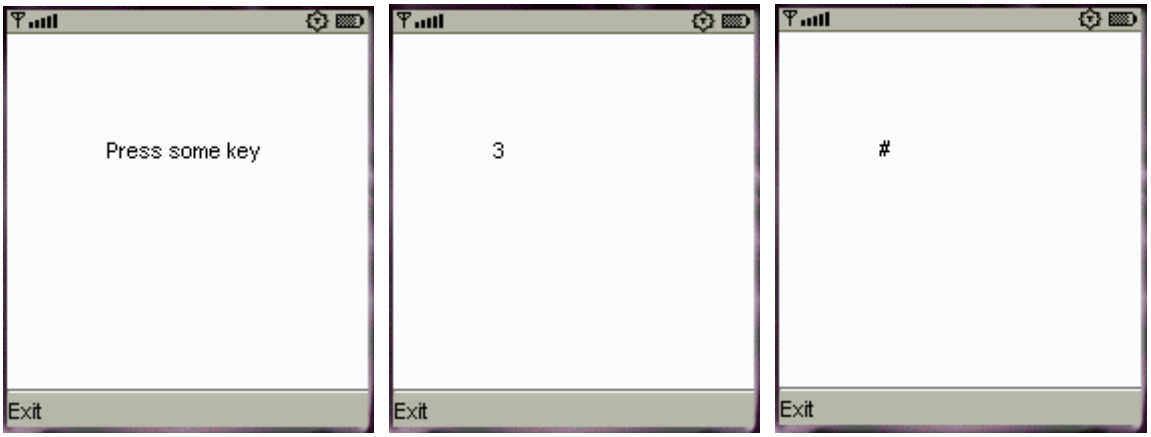
4. Compile and run the program, can you see this simple animation?

2. Capture Num Key Press

1. Open the previous project "***MyDrawing***", and then modify the java source file "***ImageCanvas.java***" as follow:

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private String KeyString = "Press some key";   // Initial String
  private boolean ExitFlag = false;              // The Flag to Exit

  public void run() {
    try {
     // Continuous to repaint the screen until the ExitFlag = True
     while (!ExitFlag) {
       repaint();          // Call paint() to repaint the screen
       Thread.sleep(100);  // Sleep for 0.1 second
     }
    } catch (Exception e) {}
  }

  public void keyPressed(int keyCode) {
    if (keyCode == KEY_NUM0) {
      KeyString = "0";
    } else if (keyCode == KEY_NUM1) {
      KeyString = "1";
    } else if (keyCode == KEY_NUM2) {
      KeyString = "2";
    } else if (keyCode == KEY_NUM3) {
      KeyString = "3";
    } else if (keyCode == KEY_NUM4) {
      KeyString = "4";
    } else if (keyCode == KEY_NUM5) {
      KeyString = "5";
    } else if (keyCode == KEY_NUM6) {
      KeyString = "6";
    } else if (keyCode == KEY_NUM7) {
      KeyString = "7";
    } else if (keyCode == KEY_NUM8) {
      KeyString = "8";
    } else if (keyCode == KEY_NUM9) {
      KeyString = "9";
    } else if (keyCode == KEY_STAR) {
      KeyString = "*";
    } else if (keyCode == KEY_POUND) {
      KeyString = "#";
    }
  }

  public void paint(Graphics g) {
    // Clear the screen
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, 180, 180);

    // Set the color to black
    g.setColor(0, 0, 0);
    // Draw the string on screen
    g.drawString(KeyString, 50, 50, g.TOP|g.LEFT);
  }
}
```

2. Compile and run the program, then press the key [0] – [9], [*] and [#].



3. Capture Game Key Press

1. Open the previous project "***MyDrawing***", and then modify the java source file "***ImageCanvas.java***" as follow:

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private String KeyString = "Press some key";   // Initial String
  private boolean ExitFlag = false;              // The Flag to Exit

  public void run() {
    try {
     // Continuous to repaint the screen until the ExitFlag = True
     while (!ExitFlag) {
       repaint();          // Call paint() to repaint the screen
       Thread.sleep(100);  // Sleep for 0.1 second
     }
    } catch (Exception e) {}
  }

  public void keyPressed(int keyCode) {
    if (getGameAction(keyCode) == LEFT)
      KeyString = "Left";
    else if (getGameAction(keyCode) == RIGHT)
      KeyString = "Right";
    else if (getGameAction(keyCode) == UP)
      KeyString = "Up";
    else if (getGameAction(keyCode) == DOWN)
      KeyString = "Down";
    else if (getGameAction(keyCode) == FIRE)
      KeyString = "Fire";
    else if (getGameAction(keyCode) == GAME_A)
      KeyString = "Game A";
    else if (getGameAction(keyCode) == GAME_B)
      KeyString = "Game B";
    else if (getGameAction(keyCode) == GAME_C)
      KeyString = "Game C";
    else if (getGameAction(keyCode) == GAME_D)
      KeyString = "Game D";
  }

  public void paint(Graphics g) {
    // Clear the screen
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, 180, 180);

    // Set the color to black
    g.setColor(0, 0, 0);
    // Draw the string on screen
    g.drawString(KeyString, 50, 50, g.TOP|g.LEFT);
  }
}
```

2. Compile and run the program, then press the arrow key, select key and numeric key to see the result.

4. Game Screen Design – Action Game

1. Create a new project, name the project and the class at "***Action***" and "***MyClass***".

2. Put the hero, brick and life images (PNG format) into the "**res**" folder (C:\WTK22\apps\Action\res).

3. Create two java files named "***MyClass.java***" and "***ImageCanvas.java***" in the "***src***" folder (C:\WTK22\apps\Action\src).

MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/**************************************************/
/* Sample Shooting Game.                          */
/**************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Define the GUI components
  private ImageCanvas MyCanvas;     // Canvas
  private Command cmdExit;          // Exit Button
  private Command cmdStart;         // Start Button

  // Define the no-argument constructor
  public MyClass() {
    // Define the Exit Button
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Define the Start Button
    cmdStart = new Command("Start", Command.SCREEN, 1);

    // Define the canvas
    MyCanvas = new ImageCanvas();
  }

  // Called by application manager to start the MIDlet
  public void startApp() {
    // Add the Command button and the Command Listener
    MyCanvas.addCommand(cmdExit);
    MyCanvas.addCommand(cmdStart);
    MyCanvas.setCommandListener(this);

    // Set the current display of the midlet to the Canvas
    Display.getDisplay(this).setCurrent(MyCanvas);
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
```

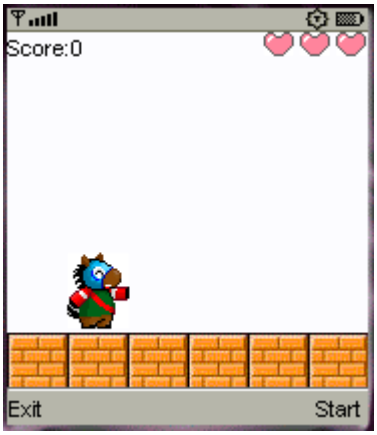
```
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas {
  private Image imgBrick;                    // Graphic for Brick
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  public void paint(Graphics g) {
    try {
        // Clear the screen
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, 180, 180);

        // Draw the screen
        for (i=0; i<7; i++) {
             g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
        }

        // Display the number of life
        for (i=1; i<=CurrentLife; i++) {
          g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
        }

        // Draw the score
        g.setColor(0, 0, 0);
        g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);
```

```
        // Draw the Hero image
        g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
    } catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

4. Compile and execute the MIDlet, you can see the draft screen for game.



5. Add the Opening

1. Open the project "***Action***" and put the opening image (PNG format) into the resource folder (C:\WTK22\apps\Action\res).

2. Modify the source file "**MyClass.java**" and "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/**************************************************/
/* Sample Shooting Game.                          */
/**************************************************/
public class MyClass extends MIDlet implements CommandListener {
  // Define the GUI components
  private ImageCanvas MyCanvas;     // Canvas
  private Command cmdExit;          // Exit Button
  private Command cmdStart;         // Start Button
  private Thread MyThread;          // Thread

  // Define the no-argument constructor
  public MyClass() {
    // Define the Exit Button
    cmdExit = new Command("Exit", Command.EXIT, 0);

    // Define the Start Button
    cmdStart = new Command("Start", Command.SCREEN, 1);

    // Define the canvas
    MyCanvas = new ImageCanvas();
  }

  // Called by application manager to start the MIDlet
  public void startApp() {
    // Add the Command button and the Command Listener
    MyCanvas.addCommand(cmdExit);
    MyCanvas.addCommand(cmdStart);
    MyCanvas.setCommandListener(this);

    // Set the current display of the midlet to the Canvas
    Display.getDisplay(this).setCurrent(MyCanvas);
```

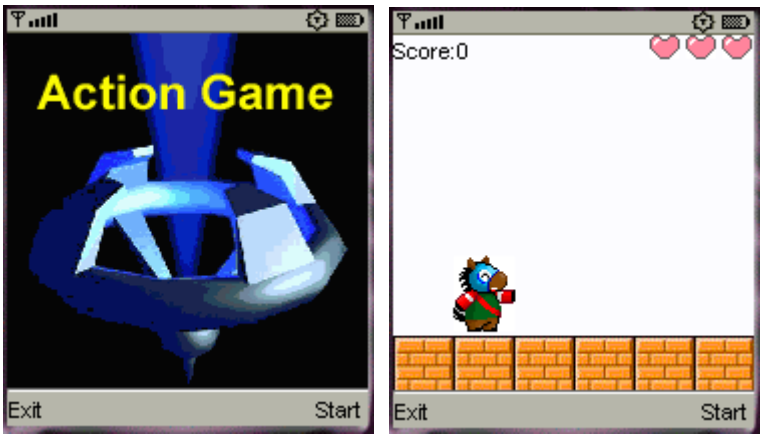
```
  }

  // PauseApp is used to suspend background activities and release resources
  // on the device when the midlet is not active.
  public void pauseApp() {
  }

  // DestroyApp is used to stop background activities and release
  // resources on the device when the midlet is at the end of its life cycle.
  public void destroyApp(boolean unconditional) {
  }

  // Implement the event handling method defined in the CommandListener interface.
  public void commandAction(Command c, Displayable s) {
    if (c == cmdStart) {
      // Start the thread only one time
      if (MyThread == null) {
        // Define the thread
        MyThread = new Thread(MyCanvas);

        // Start the thread
        MyThread.start();
      }
    } else if (c == cmdExit) {
      destroyApp(true);
      notifyDestroyed();
    }
  }
}
```

ImageCanvas.java

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over

  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgOpening = Image.createImage("/opening.png");  // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
```

```
    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;

      // Continuous to repaint the screen until the ExitFlag = True
      while (!ExitFlag) {
          repaint();                // Call paint() to repaint the screen
          Thread.sleep(100);        // Sleep for 0.1 second
      }
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }

  public void paint(Graphics g) {
    try {
      if (ScreenMode == GAME_OPEN) {
        // Clear the screen
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, 180, 180);

        // Display the Opening
        g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
      } else if (ScreenMode == GAME_PLAY) {
        // Clear the screen
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, 180, 180);

        // Draw the screen
        for (i=0; i<7; i++) {
             g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
        }

        // Display the number of life
        for (i=1; i<=CurrentLife; i++) {
          g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
        }

        // Draw the score
        g.setColor(0, 0, 0);
        g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

        // Draw the Hero image
        g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
      }
    } catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

3. Compile and execute the program, then you can see the opening screen, you can press **[Start]** to enter the game play screen.



6. Movement of the Character

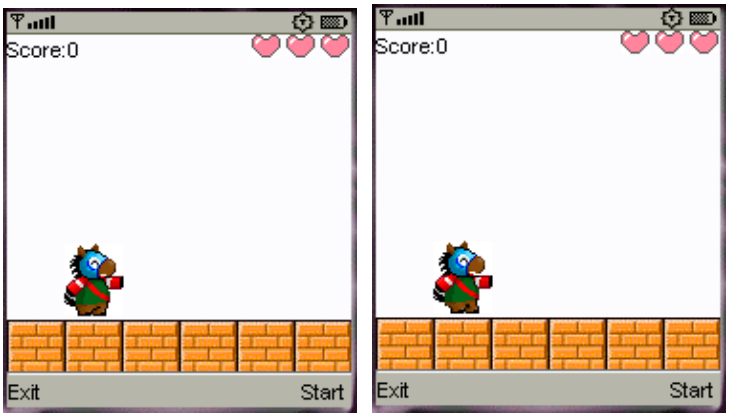
1. Open the project "***Action***" and put the hero-walking image (PNG format) into the resource folder (C:\WTK22\apps\Action\res).

2. Modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over

  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;

      // Continuous to repaint the screen until the ExitFlag = True
      while (!ExitFlag) {
```

```
            repaint();                  // Call paint() to repaint the screen
            Thread.sleep(100);          // Sleep for 0.1 second
        }
      } catch (Exception e) {
        System.out.println(e.getMessage());
      }
    }

    public void paint(Graphics g) {
      try {
        if (ScreenMode == GAME_OPEN) {
          // Clear the screen
          g.setColor(255, 255, 255);
          g.fillRect(0, 0, 180, 180);

          // Display the Opening
          g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
        } else if (ScreenMode == GAME_PLAY) {
          // Clear the screen
          g.setColor(255, 255, 255);
          g.fillRect(0, 0, 180, 180);

          // Draw the screen
          for (i=0; i<7; i++) {
               g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
          }

          // Display the number of life
          for (i=1; i<=CurrentLife; i++) {
            g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
          }

          // Draw the score
          g.setColor(0, 0, 0);
          g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

          // Set the Hero to Walk if he is stand
          if (HeroPicture == 0) {
            HeroPicture = 1;
          } else {
            HeroPicture = 0;
          }

          // Draw the Hero image
          g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
        }
      } catch(Exception e) {
        System.out.println(e.getMessage());
      }
    }
}
```

3. Compile and execute the MIDlet, can you see your character walking on the ground?



7. Handle Jump Action

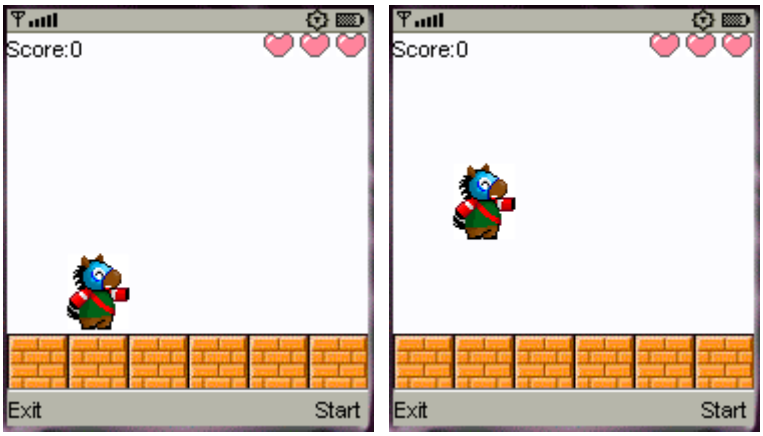
1. Open the project "**Action**" and modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;

/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread
  private int JumpAction = 0;                // Jump Action (Up/Down/Drop)

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over

  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;

      // Continuous to repaint the screen until the ExitFlag = True
      while (!ExitFlag) {
          repaint();                // Call paint() to repaint the screen
          Thread.sleep(100);        // Sleep for 0.1 second
      }
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
```

```
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Draw the screen
      for (i=0; i<7; i++) {
           g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
      }

     // Handle the Jump up action
     if (JumpAction == 1) {
       if (HeroY > 60){
         HeroY = HeroY - 5;    // Jump up until a defined high
       } else {
         JumpAction = -1;      // Reset the Jump Directions to Down
       }
     // Handle the Jump Down action
     } else if (JumpAction == -1) {
       if (HeroY < 110) {
         HeroY = HeroY + 5;    // Free Fall to the ground
       } else {
         JumpAction = 0;       // Reset to walking mode
       }
     }

      // Display the number of life
      for (i=1; i<=CurrentLife; i++) {
        g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
      }

      // Draw the score
      g.setColor(0, 0, 0);
      g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

      // Set the Hero to Walk if he is stand
      if (JumpAction == 0) {
        if (HeroPicture == 0) {
          HeroPicture = 1;
        } else {
          HeroPicture = 0;
        }
      }

      // Draw the Hero image
      g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
    }
  } catch(Exception e) {
```

```
            System.out.println(e.getMessage());
        }
    }
}
```

2. Compile and execute the MIDlet, then press **[2]** key on your mobile to control the character to jump.



8. Movement of the Character on the ground

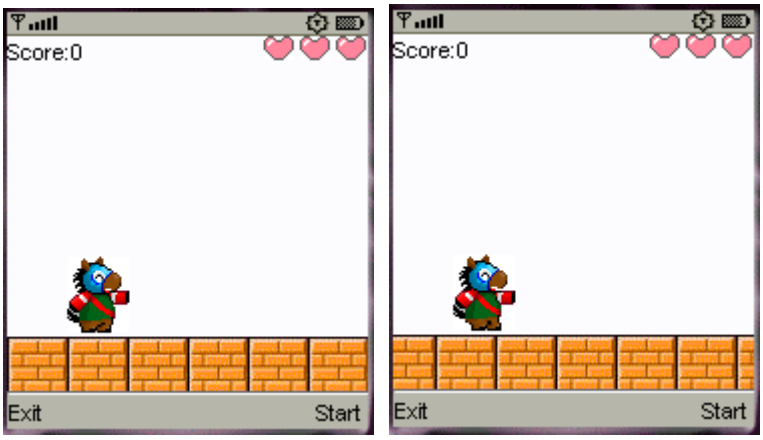
1. Open the project "**Action**" and modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;


/*-----------------------------------------------------*/
/* Canvas for graphic display                          */
/*-----------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Random rand = new Random();          // Initialize Random Seek
  private Image imgOpening;                    // Graphic for Opening
  private Image imgBrick;                      // Graphic for Brick
  private Image imgDollar;                     // Graphic for Dollar
  private Image imgLife;                       // Graphic for Life remain
  private Image imgHero[] = new Image[2];      // Graphic for Hero moving
  private int HeroY = 110;                     // Y-coordinate for Hero
  private int HeroPicture = 0;                 // Index for Hero picture
  private int Map[] = new int[8];              // Map
  private int MapX = 0;                        // X coordinate for Map
  private int i;                               // Looping variables
  private int CurrentLife = 3;                 // Number of Life leave
  private int score = 0;                       // Display the Score
  private int ScreenMode = 0;                  // Game Screen Status
  private boolean ExitFlag = false;            // The Flag to Exit the Thread
  private int JumpAction = 0;                  // Jump Action (Up/Down/Drop)
  private int HeroSpeed = 3;                   // The Walking speed (1 - 9)

  // Define the game status
  final int   GAME_OPEN = 0,              // Opening
              GAME_PLAY = 1,              // Playing
              GAME_DROP = 2,              // Drop to Hole
              GAME_OVER = 9;              // Game Over


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;

      // Continuous to repaint the screen until the ExitFlag = True
```

```
    while (!ExitFlag) {
        repaint();                // Call paint() to repaint the screen
        Thread.sleep(100);        // Sleep for 0.1 second
    }
  } catch (Exception e) {
    System.out.println(e.getMessage());
  }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Move the map to left according to Hero speed
      MapX = MapX - HeroSpeed;

      if (MapX < -29) {
         // Shift the Map to left
        MapX = 0;
        for (i=0; i<7; i++) {
          Map[i] = Map[i+1];
        }
        // Generate the next map
        Map[7] = Math.abs(rand.nextInt())%5;
      }

      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Draw the screen
      for (i=0; i<7; i++) {
           g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
      }

     // Handle the Jump up action
      if (JumpAction == 1) {
        if (HeroY > 60){
          HeroY = HeroY - 5;   // Jump up until a defined high
        } else {
          JumpAction = -1;     // Reset the Jump Directions to Down
        }
     // Handle the Jump Down action
      } else if (JumpAction == -1) {
        if (HeroY < 110) {
          HeroY = HeroY + 5;   // Free Fall to the ground
        } else {
          JumpAction = 0;      // Reset to walking mode
        }
      }

      // Display the number of life
      for (i=1; i<=CurrentLife; i++) {
```

```
          g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
        }

        // Draw the score
        g.setColor(0, 0, 0);
        g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

        // Set the Hero to Walk if he is stand
        if (JumpAction == 0) {
          if (HeroPicture == 0) {
            HeroPicture = 1;
          } else {
            HeroPicture = 0;
          }
        }

        // Draw the Hero image
        g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
      }
    } catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

2. Compile and execute the MIDlet, you can see your character walking on the ground now.



9. Create the Map

1. Open the project "***Action***" and put the dollar image (PNG format) into the resource folder (C:\WTK22\apps\Action\res).

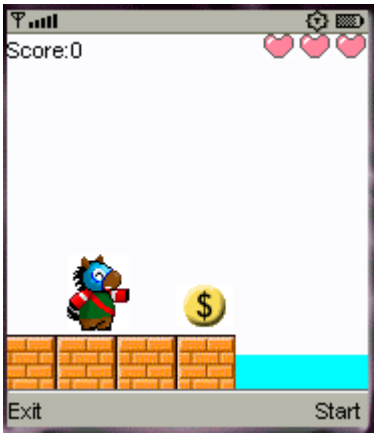
2. Modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Random rand = new Random();        // Initialize Random Seek
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgDollar;                   // Graphic for Dollar
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread
  private int JumpAction = 0;                // Jump Action (Up/Down/Drop)
  private int HeroSpeed = 3;                 // The Walking speed (1 - 9)

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgDollar = Image.createImage("/dollar.png");    // Dollar
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }
```

```
// Execute the Thread
public void run() {
  try {
    // Start the game
    ScreenMode = GAME_PLAY;

    // Continuous to repaint the screen until the ExitFlag = True
    while (!ExitFlag) {
        repaint();                // Call paint() to repaint the screen
        Thread.sleep(100);        // Sleep for 0.1 second
    }
  } catch (Exception e) {
    System.out.println(e.getMessage());
  }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Move the map to left according to Hero speed
      MapX = MapX - HeroSpeed;

      if (MapX < -29) {
         // Shift the Map to left
         MapX = 0;
         for (i=0; i<7; i++) {
           Map[i] = Map[i+1];
         }
        // Generate the next map
        Map[7] = Math.abs(rand.nextInt())%5;
      }

      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Draw the screen
      for (i=0; i<7; i++) {
        switch (Map[i]) {
          case 0: // 20% to display a hole
            g.setColor(0, 255, 255);
            g.fillRect(MapX + 30*i, 160, 30, 30);
            break;

          case 1: // 20% to display a dollar
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
            g.drawImage(imgDollar, MapX + 30*i, 125, g.TOP|g.LEFT);
            break;

          default:    // default to display a ground
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
            break;
```

```
        }
      }

      // Handle the Jump up action
      if (JumpAction == 1) {
        if (HeroY > 60){
          HeroY = HeroY - 5;    // Jump up until a defined high
        } else {
          JumpAction = -1;      // Reset the Jump Directions to Down
        }
      // Handle the Jump Down action
      } else if (JumpAction == -1) {
        if (HeroY < 110) {
          HeroY = HeroY + 5;    // Free Fall to the ground
        } else {
          JumpAction = 0;       // Reset to walking mode
        }
      }

      // Display the number of life
      for (i=1; i<=CurrentLife; i++) {
        g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
      }

      // Draw the score
      g.setColor(0, 0, 0);
      g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

      // Set the Hero to Walk if he is stand
      if (JumpAction == 0) {
        if (HeroPicture == 0) {
          HeroPicture = 1;
        } else {
          HeroPicture = 0;
        }
      }

      // Draw the Hero image
      g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
    }
   } catch(Exception e) {
     System.out.println(e.getMessage());
   }
  }
}
```

3. Compile and execute the MIDlet, the map will change when you're walking along it.



10. Increase the Score after Get Coins

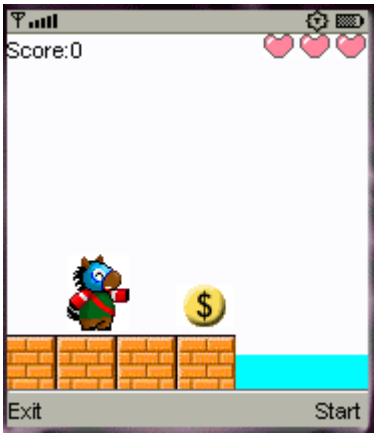
1. Open the project "**Action**" and modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Random rand = new Random();        // Initialize Random Seek
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgDollar;                   // Graphic for Dollar
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread
  private int JumpAction = 0;                // Jump Action (Up/Down/Drop)
  private int HeroSpeed = 3;                 // The Walking speed (1 - 9)

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgDollar = Image.createImage("/dollar.png");    // Dollar
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;
```

```
    // Continuous to repaint the screen until the ExitFlag = True
    while (!ExitFlag) {
        repaint();                // Call paint() to repaint the screen
        Thread.sleep(100);        // Sleep for 0.1 second
    }
  } catch (Exception e) {
    System.out.println(e.getMessage());
  }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Move the map to left according to Hero speed
      MapX = MapX - HeroSpeed;

      if (MapX < -29) {
         // Shift the Map to left
        MapX = 0;
        for (i=0; i<7; i++) {
          Map[i] = Map[i+1];
        }
       // Generate the next map
        Map[7] = Math.abs(rand.nextInt())%5;
      }

      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Draw the screen
      for (i=0; i<7; i++) {
        switch (Map[i]) {
          case 0: // 20% to display a hole
            g.setColor(0, 255, 255);
            g.fillRect(MapX + 30*i, 160, 30, 30);
            break;

          case 1: // 20% to display a dollar
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
            g.drawImage(imgDollar, MapX + 30*i, 125, g.TOP|g.LEFT);
            break;

          default:    // default to display a ground
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
            break;
        }
      }

     // Handle the Jump up action
      if (JumpAction == 1) {
        if (HeroY > 60){
```

```
            HeroY = HeroY - 5;    // Jump up until a defined high
          } else {
            JumpAction = -1;      // Reset the Jump Directions to Down
          }
        // Handle the Jump Down action
        } else if (JumpAction == -1) {
          if (HeroY < 110) {
            HeroY = HeroY + 5;    // Free Fall to the ground
          } else {
            JumpAction = 0;       // Reset to walking mode
          }
        }

        // Check for Get Coin action
        if (MapX < 0) {
          if (Map[2] == 1 && HeroY > 90) {
            // Increase the score if get a coin
            score = score + 10;
            // Reset the map to ground
            Map[2] = 2;
          }
        }

        // Display the number of life
        for (i=1; i<=CurrentLife; i++) {
          g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
        }

        // Draw the score
        g.setColor(0, 0, 0);
        g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

        // Set the Hero to Walk if he is stand
        if (JumpAction == 0) {
          if (HeroPicture == 0) {
            HeroPicture = 1;
          } else {
            HeroPicture = 0;
          }
        }

        // Draw the Hero image
        g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
      }
    } catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

2. Compile and Execute the MIDlet, you can walk along to get coin and increase the score.



11. Control the Speed of the Character

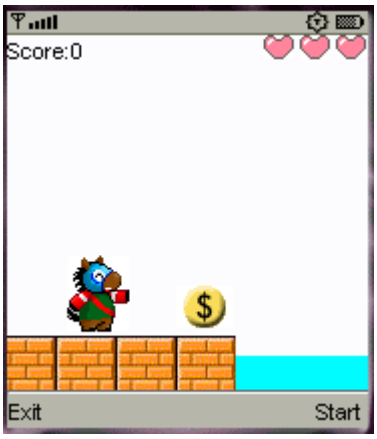
1. Open the project "**Action**" and modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Random rand = new Random();        // Initialize Random Seek
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgDollar;                   // Graphic for Dollar
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread
  private int JumpAction = 0;                // Jump Action (Up/Down/Drop)
  private int HeroSpeed = 3;                 // The Walking speed (1 - 9)

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgDollar = Image.createImage("/dollar.png");    // Dollar
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;
```

```
    // Continuous to repaint the screen until the ExitFlag = True
    while (!ExitFlag) {
        repaint();                // Call paint() to repaint the screen
        Thread.sleep(100);        // Sleep for 0.1 second
    }
  } catch (Exception e) {
    System.out.println(e.getMessage());
  }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  // Walk faster if press [6]
  } else if (keyCode == KEY_NUM6) {
    // Only allow to walk faster if the speed is less than 9
    if (HeroSpeed < 9) {
      HeroSpeed = HeroSpeed + 1;
    }
  // Walk slower if press [4]
  } else if (keyCode == KEY_NUM4) {
    // Only allow to walk faster if the speed is greater than 1
    if (HeroSpeed > 1) {
      HeroSpeed = HeroSpeed - 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Move the map to left according to Hero speed
      MapX = MapX - HeroSpeed;

      if (MapX < -29) {
         // Shift the Map to left
        MapX = 0;
        for (i=0; i<7; i++) {
          Map[i] = Map[i+1];
        }
       // Generate the next map
        Map[7] = Math.abs(rand.nextInt())%5;
      }

      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Draw the screen
      for (i=0; i<7; i++) {
        switch (Map[i]) {
          case 0: // 20% to display a hole
            g.setColor(0, 255, 255);
            g.fillRect(MapX + 30*i, 160, 30, 30);
            break;

          case 1: // 20% to display a dollar
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
```

```
            g.drawImage(imgDollar, MapX + 30*i, 125, g.TOP|g.LEFT);
            break;

          default:    // default to display a ground
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
            break;
        }
      }

      // Handle the Jump up action
      if (JumpAction == 1) {
        if (HeroY > 60){
          HeroY = HeroY - 5;   // Jump up until a defined high
        } else {
          JumpAction = -1;     // Reset the Jump Directions to Down
        }
      // Handle the Jump Down action
      } else if (JumpAction == -1) {
        if (HeroY < 110) {
          HeroY = HeroY + 5;   // Free Fall to the ground
        } else {
          JumpAction = 0;      // Reset to walking mode
        }
      }

      // Check for Get Coin action
      if (MapX < 0) {
        if (Map[2] == 1 && HeroY > 90) {
          // Increase the score if get a coin
          score = score + 10;
          // Reset the map to ground
          Map[2] = 2;
        }
      }

      // Display the number of life
      for (i=1; i<=CurrentLife; i++) {
        g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
      }

      // Draw the score
      g.setColor(0, 0, 0);
      g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

      // Set the Hero to Walk if he is stand
      if (JumpAction == 0) {
        if (HeroPicture == 0) {
          HeroPicture = 1;
        } else {
          HeroPicture = 0;
        }
      }

      // Draw the Hero image
      g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
    }
   } catch(Exception e) {
     System.out.println(e.getMessage());
   }
  }
}
```

2. Compile and Execute the MIDlet, press **[4]** and **[6]** on your mobile to control the speed of your character.



12. Drop to the hole if can't jump over it

1. Open the project "**Action**" and modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

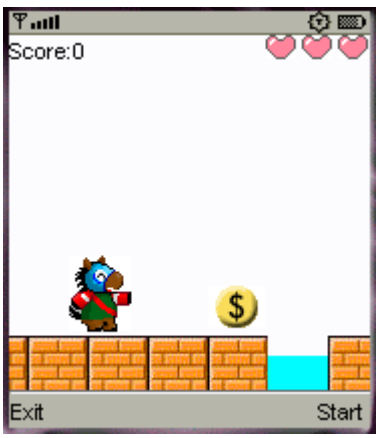
```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Random rand = new Random();        // Initialize Random Seek
  private Image imgOpening;                  // Graphic for Opening
  private Image imgBrick;                    // Graphic for Brick
  private Image imgDollar;                   // Graphic for Dollar
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread
  private int JumpAction = 0;                // Jump Action (Up/Down/Drop)
  private int HeroSpeed = 3;                 // The Walking speed (1 - 9)

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgDollar = Image.createImage("/dollar.png");    // Dollar
      imgBrick = Image.createImage("/brick.png");      // Brick
      imgLife = Image.createImage("/life.png");        // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");    // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");    // Hero Walk
      imgOpening = Image.createImage("/opening.png"); // Opening
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }

    // Initial the map (set it as ground first)
    for (i=0; i<8; i++) {
      Map[i] = 2;
    }
  }

  // Execute the Thread
  public void run() {
    try {
      // Start the game
      ScreenMode = GAME_PLAY;
```

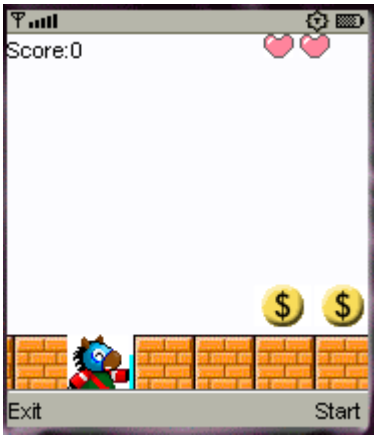
```
    // Continuous to repaint the screen until the ExitFlag = True
    while (!ExitFlag) {
        repaint();                // Call paint() to repaint the screen
        Thread.sleep(100);        // Sleep for 0.1 second
    }
  } catch (Exception e) {
    System.out.println(e.getMessage());
  }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  // Walk faster if press [6]
  } else if (keyCode == KEY_NUM6) {
    // Only allow to walk faster if the speed is less than 9
    if (HeroSpeed < 9) {
      HeroSpeed = HeroSpeed + 1;
    }
  // Walk slower if press [4]
  } else if (keyCode == KEY_NUM4) {
    // Only allow to walk faster if the speed is greater than 1
    if (HeroSpeed > 1) {
      HeroSpeed = HeroSpeed - 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Move the map to left according to Hero speed
      MapX = MapX - HeroSpeed;

      if (MapX < -29) {
         // Shift the Map to left
        MapX = 0;
        for (i=0; i<7; i++) {
          Map[i] = Map[i+1];
        }
       // Generate the next map
        Map[7] = Math.abs(rand.nextInt())%5;
      }

      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Draw the screen
      for (i=0; i<7; i++) {
        switch (Map[i]) {
          case 0: // 20% to display a hole
            g.setColor(0, 255, 255);
            g.fillRect(MapX + 30*i, 160, 30, 30);
            break;

          case 1: // 20% to display a dollar
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
```

```
            g.drawImage(imgDollar, MapX + 30*i, 125, g.TOP|g.LEFT);
            break;

          default:    // default to display a ground
            g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
            break;
        }
      }

      // Handle the Jump up action
      if (JumpAction == 1) {
        if (HeroY > 60){
          HeroY = HeroY - 5;   // Jump up until a defined high
        } else {
          JumpAction = -1;     // Reset the Jump Directions to Down
        }
      // Handle the Jump Down action
      } else if (JumpAction == -1) {
        if (HeroY < 110) {
          HeroY = HeroY + 5;   // Free Fall to the ground
        } else {
          JumpAction = 0;      // Reset to walking mode
        }
      }

      // Check for Get Coin action
      if (MapX < 0) {
        if (Map[2] == 1 && HeroY > 90) {
          // Increase the score if get a coin
          score = score + 10;
          // Reset the map to ground
          Map[2] = 2;
        }
      }

      // Check for Drop to a hole
      if (MapX < -25 && Map[2] == 0 && HeroY == 110) {
        JumpAction = -99;                    // Set the Flag to Drop
        HeroY = 150;                         // Draw the hero in the hole
        CurrentLife = CurrentLife - 1;        // Reduce one life
        ScreenMode = GAME_DROP;              // Change the flag to GAME_DROP
      }

      // Display the number of life
      for (i=1; i<=CurrentLife; i++) {
        g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
      }

      // Draw the score
      g.setColor(0, 0, 0);
      g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

      // Set the Hero to Walk if he is stand
      if (JumpAction == 0) {
        if (HeroPicture == 0) {
          HeroPicture = 1;
        } else {
          HeroPicture = 0;
        }
      }

      // Draw the Hero image
      g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_DROP) {
      // Pause the screen for 2 seconds
      Thread.sleep(2000);

      // Reset the Screen
      for (i=0; i<8; i++) {    // Reset the map to ground
        Map[i] = 2;
```

```
        }
        HeroY = 110;              // Reset Hero location
        HeroSpeed = 3;            // Reset the hero speed
        JumpAction = 0;           // Set the Hero to Walk (No Jump)

        // Reset the Game Screen Flag
        ScreenMode = GAME_PLAY;
      }
    } catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

2. Compile and Execute the MIDlet, you must pay attention to the hole or you'll drop to it. However, can you see the bugs here?





13. Handle the Game Over

1. Open the project "***Action***" and put the game over image (PNG format) into the resource folder (C:\WTK22\apps\Action\res).

2. Modify the source file "**ImageCanvas.java**" in the "**src**" folder (C:\WTK22\apps\Action\src).

```
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// import the random library
import java.util.Random;


/*-------------------------------------------------*/
/* Canvas for graphic display                      */
/*-------------------------------------------------*/
public class ImageCanvas extends Canvas implements Runnable {
  private Random rand = new Random();        // Initialize Random Seek
  private Image imgOpening;                  // Graphic for Opening
  private Image imgGameover;                 // Graphic for Game over
  private Image imgBrick;                    // Graphic for Brick
  private Image imgDollar;                   // Graphic for Dollar
  private Image imgLife;                     // Graphic for Life remain
  private Image imgHero[] = new Image[2];    // Graphic for Hero moving
  private int HeroY = 110;                   // Y-coordinate for Hero
  private int HeroPicture = 0;               // Index for Hero picture
  private int Map[] = new int[8];            // Map
  private int MapX = 0;                      // X coordinate for Map
  private int i;                             // Looping variables
  private int CurrentLife = 3;               // Number of Life leave
  private int score = 0;                     // Display the Score
  private int ScreenMode = 0;                // Game Screen Status
  private boolean ExitFlag = false;          // The Flag to Exit the Thread
  private int JumpAction = 0;                // Jump Action (Up/Down/Drop)
  private int HeroSpeed = 3;                 // The Walking speed (1 - 9)

  // Define the game status
  final int   GAME_OPEN = 0,             // Opening
              GAME_PLAY = 1,             // Playing
              GAME_DROP = 2,             // Drop to Hole
              GAME_OVER = 9;             // Game Over


  public ImageCanvas() {
    // Load the image from resource folder
    try {
      imgDollar = Image.createImage("/dollar.png");        // Dollar
      imgBrick = Image.createImage("/brick.png");          // Brick
      imgLife = Image.createImage("/life.png");            // Life Remain
      imgHero[0] = Image.createImage("/hero0.png");        // Hero stand
      imgHero[1] = Image.createImage("/hero1.png");        // Hero Walk
      imgOpening = Image.createImage("/opening.png");      // Opening
      imgGameover = Image.createImage("/gameover.png");    // Game Over
    } catch (Exception e) {
```

```
    System.out.println(e.getMessage());
  }

  // Initial the map (set it as ground first)
  for (i=0; i<8; i++) {
    Map[i] = 2;
  }
}

// Execute the Thread
public void run() {
  try {
    // Start the game
    ScreenMode = GAME_PLAY;

    // Continuous to repaint the screen until the ExitFlag = True
    while (!ExitFlag) {
        repaint();                // Call paint() to repaint the screen
        Thread.sleep(100);        // Sleep for 0.1 second
    }
  } catch (Exception e) {
    System.out.println(e.getMessage());
  }
}

// KeyCode Event Handler to detecting game actions
public void keyPressed(int keyCode) {
  // Move the Hero up when press [2]
  if (keyCode == KEY_NUM2) {
    // Only allow Hero jump if he is walking on the ground
    if (JumpAction == 0) {
      JumpAction = 1;
    }
  // Walk faster if press [6]
  } else if (keyCode == KEY_NUM6) {
    // Only allow to walk faster if the speed is less than 9
    if (HeroSpeed < 9) {
      HeroSpeed = HeroSpeed + 1;
    }
  // Walk slower if press [4]
  } else if (keyCode == KEY_NUM4) {
    // Only allow to walk faster if the speed is greater than 1
    if (HeroSpeed > 1) {
      HeroSpeed = HeroSpeed - 1;
    }
  }
}

public void paint(Graphics g) {
  try {
    if (ScreenMode == GAME_OPEN) {
      // Clear the screen
      g.setColor(255, 255, 255);
      g.fillRect(0, 0, 180, 180);

      // Display the Opening
      g.drawImage(imgOpening, 0, 0, g.TOP|g.LEFT);
    } else if (ScreenMode == GAME_PLAY) {
      // Move the map to left according to Hero speed
      MapX = MapX - HeroSpeed;

      if (MapX < -29) {
        // Shift the Map to left
        MapX = 0;
        for (i=0; i<7; i++) {
          Map[i] = Map[i+1];
        }
        // Generate the next map
        Map[7] = Math.abs(rand.nextInt())%5;
      }
```

```
        // Clear the screen
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, 180, 180);

        // Draw the screen
        for (i=0; i<7; i++) {
          switch (Map[i]) {
            case 0: // 20% to display a hole
              g.setColor(0, 255, 255);
              g.fillRect(MapX + 30*i, 160, 30, 30);
              break;

            case 1: // 20% to display a dollar
              g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
              g.drawImage(imgDollar, MapX + 30*i, 125, g.TOP|g.LEFT);
              break;

            default:    // default to display a ground
              g.drawImage(imgBrick, MapX + 30*i, 150, g.TOP|g.LEFT);
              break;
          }
        }

       // Handle the Jump up action
        if (JumpAction == 1) {
          if (HeroY > 60){
            HeroY = HeroY - 5;   // Jump up until a defined high
          } else {
            JumpAction = -1;     // Reset the Jump Directions to Down
          }
       // Handle the Jump Down action
        } else if (JumpAction == -1) {
          if (HeroY < 110) {
            HeroY = HeroY + 5;   // Free Fall to the ground
          } else {
            JumpAction = 0;      // Reset to walking mode
          }
        }

        // Check for Get Coin action
        if (MapX < 0) {
          if (Map[2] == 1 && HeroY > 90) {
            // Increase the score if get a coin
            score = score + 10;
            // Reset the map to ground
            Map[2] = 2;
          }
        }

        // Check for Drop to a hole
        if (MapX < -25 && Map[2] == 0 && HeroY == 110) {
          JumpAction = -99;                    // Set the Flag to Drop
          HeroY = 150;                         // Draw the hero in the hole
          CurrentLife = CurrentLife - 1;        // Reduce one life
          ScreenMode = GAME_DROP;              // Change the flag to GAME_DROP

         // Change to Game Over if no life remain
         if (CurrentLife == 0) {
            ScreenMode = GAME_OVER;
          }
        }

        // Display the number of life
        for (i=1; i<=CurrentLife; i++) {
          g.drawImage(imgLife, 110+i*18, 0, g.TOP|g.LEFT);
        }

        // Draw the score
        g.setColor(0, 0, 0);
```

```
        g.drawString("Score:" + score, 0, 0, g.TOP|g.LEFT);

        // Set the Hero to Walk if he is stand
        if (JumpAction == 0) {
          if (HeroPicture == 0) {
            HeroPicture = 1;
          } else {
            HeroPicture = 0;
          }
        }

        // Draw the Hero image
        g.drawImage(imgHero[HeroPicture], 30, HeroY, g.TOP|g.LEFT);
      } else if (ScreenMode == GAME_DROP) {
        // Pause the screen for 2 seconds
        Thread.sleep(2000);

        // Reset the Screen
        for (i=0; i<8; i++) {    // Reset the map to ground
          Map[i] = 2;
        }
        HeroY = 110;             // Reset Hero location
        HeroSpeed = 3;           // Reset the hero speed
        JumpAction = 0;          // Set the Hero to Walk (No Jump)

        // Reset the Game Screen Flag
        ScreenMode = GAME_PLAY;
      } else if (ScreenMode == GAME_OVER) {
        // Display the GameOver
        g.drawImage(imgGameover, 0, 0, g.TOP|g.LEFT);
        ExitFlag = true;
      }
    } catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

3. Compile and execute the MIDlet, how long can you survive?