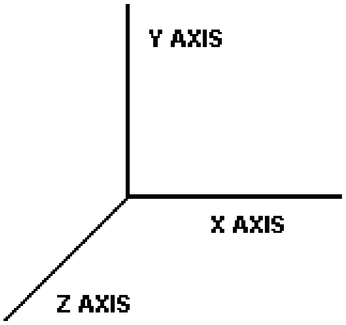
Layer Manager

Back | Tutorial Home | Next

In last section of this chapter we were dealing with one sprite, of course, we can easily put more then one sprite on the screen but what about the scenery. Virtually all games will have some sort of background whether it is animated, still or scrolling there is usually something there to give the game more visual appeal. This is where the LayerManager comes in, the LayerManager does exactly what it is named as, manages graphic layers. It will render all the layers under its control in the correct area and in sequential order. In other words it overlaps images over one another in an easily manageable fashion.

You can think of the ordering of layers as a 3 rd dimension commonly called the z.

Figure: Z Axis Illustration



These layers on z-axis are indexed in a numeric order starting from zero. Index 0 is the layer closest to the user; likewise the layer with the highest index value is furthest from the user. If there are any adjustments to the number of layers in the LayerManager the numbering of indexes with respect to each layer is re-numbered according to keep a continuous sequential order. This would occur if layers are added or removed dynamically throughout the game.

How to use the LayerManager

To add an layer the method append(Layer layer) is available

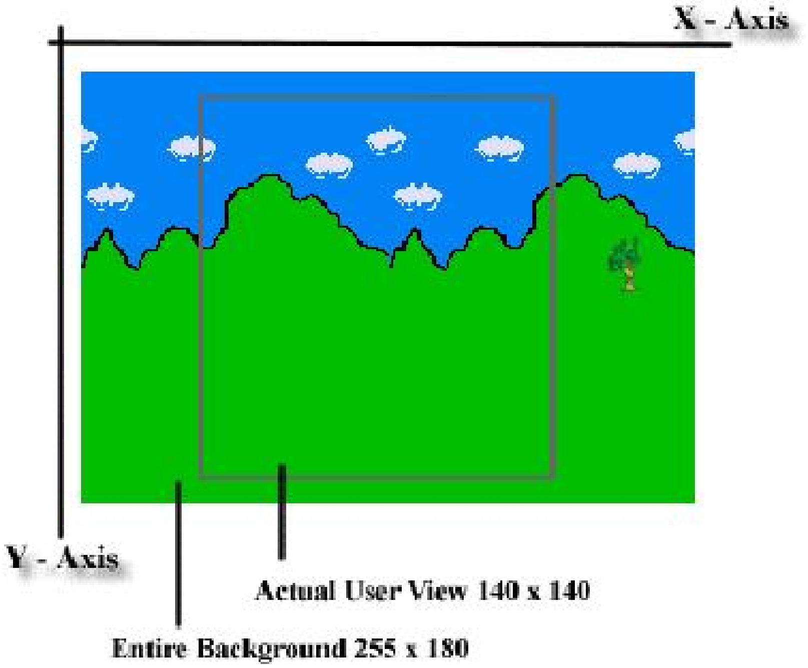
```
LayerManager = new LayerManager();

layerManager.append(layer);
```

To retrieve a layer a specific index use the getLayerAt(int index) method. To obtain total number of layers currently in the LayerManager use the getSize() method. To remove a layer use the remove(Layer layer) method. To add a specific layer at specific index the following method insert(Layer layer, int index) will allow you to do this. To display the layers call the paint(Graphics g, int x, int y) method.

Beyond the Basics

In the last section the mentioned methods are fairly straightforward and really you could easily implement this yourself without using the LayerManager. However, there is one more method that provides a very convenient feature, the setViewWindow(int x, int y, int width, int height) method. This feature allows you to set both the actual size of the window the user can view and what part of the layer to display in the view. Yes this sounds a bit confusing at first but is more easily understood through the following diagram.

Figure: setViewWindowIlustration



The actual method implementation for the above diagram would be setViewWindow(55,20,140,140). Where the first set of numbers is where the top left corner of the User View is located with respect to the entire background, (55,20). The second set of number 140 x 140 is the actual width and height for the User view. The demo code and actual screen shot of this will be demonstrated in the next section.

Basic LayerManager Example

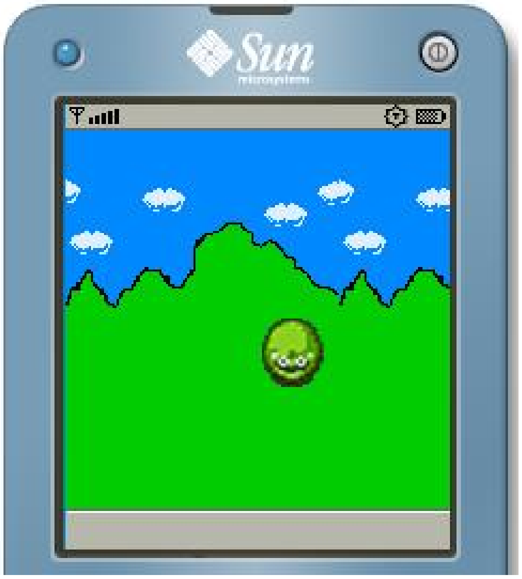
Game Canvas using the LayerManager

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int currentX, currentY;  // To hold current position of the 'X'
  private int width;        // To hold screen width
  private int height;       // To hold screen height

  // Sprites to be used
  private Sprite playerSprite;
  private Sprite backgroundSprite;

  // Layer Manager
  private LayerManager layerManager;

  // Constructor and initialization
  public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();

    currentX = width / 2;
    currentY = height / 2;
    delay = 20;

    // Load Images to Sprites
    Image playerImage = Image.createImage("/transparent.png");
    playerSprite = new Sprite (playerImage,32,32);

    Image backgroundImage = Image.createImage("/background.png");
    backgroundSprite = new Sprite(backgroundImage);

    layerManager = new LayerManager();
    layerManager.append(playerSprite);
    layerManager.append(backgroundSprite);

  }

  // Automatically start thread for game loop
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }

  public void stop() { isPlay = false; }

  // Main Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {

      input();
      drawScreen(g);
      try { Thread.sleep(delay); }
      catch (InterruptedException ie) {}
    }
  }

  // Method to Handle User Inputs
  private void input() {
    int keyStates = getKeyStates();

    playerSprite.setFrame(0);

    // Left
    if ((keyStates & LEFT_PRESSED) != 0) {
      currentX = Math.max(0, currentX - 1);
      playerSprite.setFrame(1);
    }

    // Right
    if ((keyStates & RIGHT_PRESSED) !=0 )
      if ( currentX + 5 < width) {
        currentX = Math.min(width, currentX + 1);
        playerSprite.setFrame(3);
      }

    // Up
    if ((keyStates & UP_PRESSED) != 0) {
      currentY = Math.max(0, currentY - 1);
      playerSprite.setFrame(2);
    }

    // Down
    if ((keyStates & DOWN_PRESSED) !=0)
      if ( currentY + 10 < height) {
        currentY = Math.min(height, currentY + 1);
        playerSprite.setFrame(4);
      }
  }

  // Method to Display Graphics
  private void drawScreen(Graphics g) {

    //g.setColor(0x00C000);
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);

    // updating player sprite position
    playerSprite.setPosition(currentX,currentY);

    // display all layers
    //layerManager.paint(g,0,0);
    layerManager.setViewWindow(55,20,140,140);
    layerManager.paint(g,20,20);

    flushGraphics();
  }

}
```

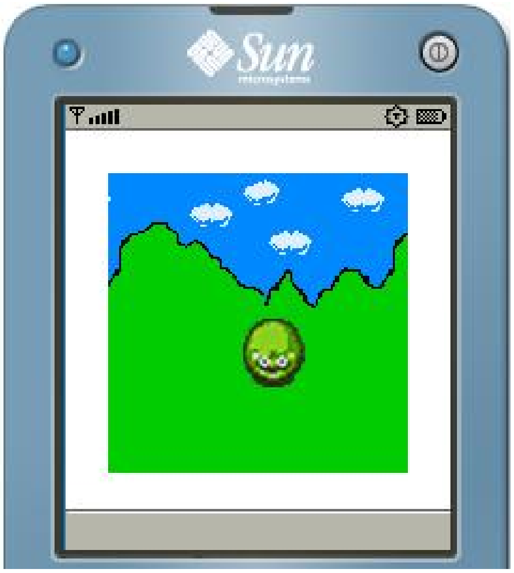
Main Midlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleLayerManagerMidlet extends MIDlet {
  private Display display;

  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }

  public Display getDisplay() {
    return display;
  }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
    exit();
  }

  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
```

Simple LayerManager Screen Shot



LayerManager using setViewWindow method Screen Shot



The figure demonstrates the use of the setViewWindow, the following is the source snippet to do this (replace the current drawScreen method).

```
private void drawScreen(Graphics g) {
        g.setColor(0xffffff);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0x0000ff);
        // updating player sprite position
        playerSprite.setPosition(currentX,currentY);
        // display all layers
        layerManager.setViewWindow(55,20,140,140);
        layerManager.paint(g,20,20);
        flushGraphics();
}
```

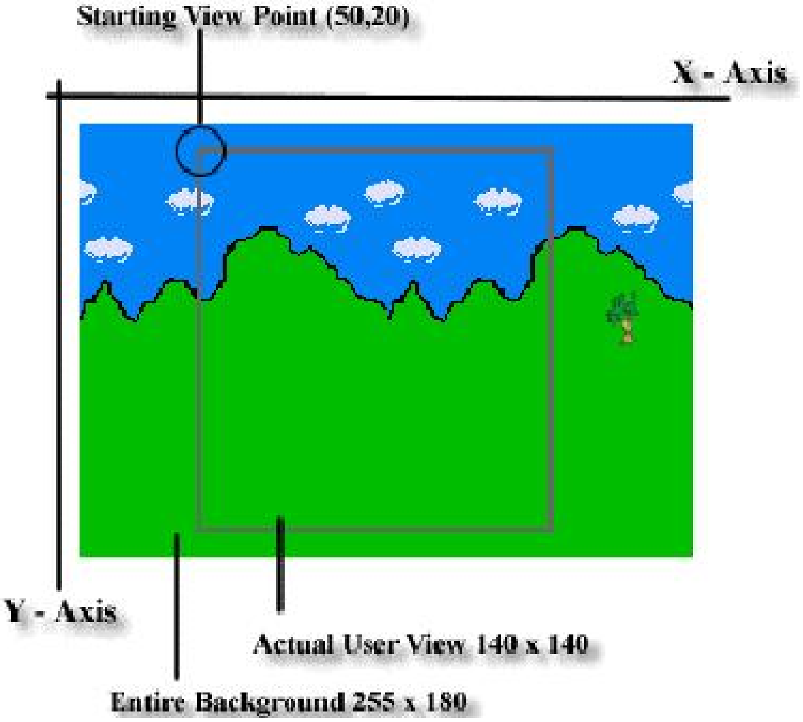
One more note you will notice just using the setViewWindow method the user viewing area will appear offer center. If you want to adjust this you will need to put in the appropriate values in the paint method in this case to center the view the values are 20- pixel a-axis and 20-pixel y-axis. This point is offset from the actual screen itself. Do not confuse this with the first values in the setViewWindow method. Remember these two numbers are offset from the layers being used.

Now that you have some space at the top and bottom of the screen you can now add additional feed back to the user. For example, you can now display the high score, current user score, level and number of lives.

LayerManager and Scrolling Background

You've probably noticed the background image we have been using is bigger then the display screen itself.

Scrolling View


Starting View Point (50,20)
X - Axis
Y - Axis
Actual User View 140 x 140
Entire Background 255 x 180

In this particular example the starting point is (50,20). All you have to do is make the value 50 a dynamic value so that it decreases if you want the background to scroll to the left or increase if you want the background to scroll to the right.

The following code illustrates how this can be done, for simplicity the green smiley sprite has been removed and now if you press the right game key the image will scroll right and if you press the left game key the image will scroll left.

Note in this example, only the horizontal value is made dynamic nothing will happen if you decide to push the down or up key. Which in this scenario makes sense, but of course you can easily change this to scroll up/down or even scroll in all 4 directions.

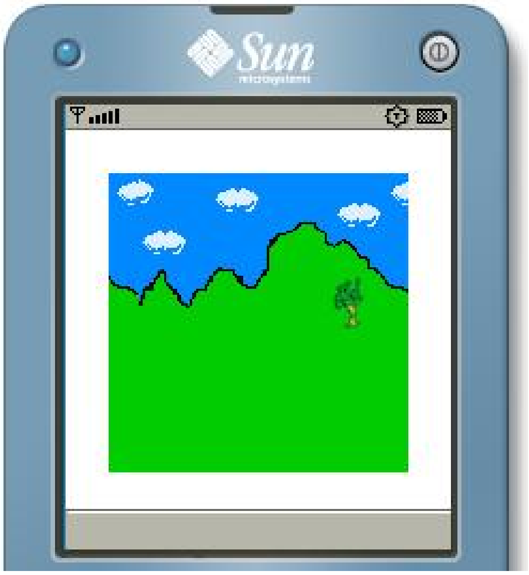
Scrolling Game Canvas

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
  private boolean isPlay;   // Game Loop runs when isPlay is true
  private long delay;       // To give thread consistency
  private int width;        // To hold screen width
  private int height;       // To hold screen height
  private int scnX, scnY;   // To hold screen starting viewpoint


  // Sprites to be used
  Image backgroundImage;
  private Sprite backgroundSprite;

  // Layer Manager
  private LayerManager layerManager;

  // Constructor and initialization
  public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();

    scnX = 55;
    scnY = 20;
    delay = 20;

    // Load Images to Sprites
    backgroundImage = Image.createImage("/background.png");
    backgroundSprite = new Sprite(backgroundImage);

    layerManager = new LayerManager();
    layerManager.append(backgroundSprite);

  }

  // Automatically start thread for game loop
  public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
  }

  public void stop() { isPlay = false; }

  // Main Game Loop
  public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {

      input();
      drawScreen(g);
      try { Thread.sleep(delay); }
      catch (InterruptedException ie) {}
    }
  }

  // Method to Handle User Inputs
  private void input() {
    int keyStates = getKeyStates();


    if ((keyStates & LEFT_PRESSED) != 0) {
      if (scnX - 1 > 0)
        scnX--;
    }
    if ((keyStates & RIGHT_PRESSED) != 0) {
      if (scnX + 1 + 140 < backgroundImage.getWidth())
        scnX++;
    }
  }

  // Method to Display Graphics
  private void drawScreen(Graphics g) {

    //g.setColor(0x00C000);
    g.setColor(0xffffff);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0x0000ff);

    // display all layers
    layerManager.setViewWindow(scnX,scnY,140,140);
    layerManager.paint(g,20,20);

    flushGraphics();
  }

}
```

Main Midlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class SimpleScrollingLayerManger extends MIDlet {
  private Display display;

  public void startApp() {
    try {
      display = Display.getDisplay(this);
      ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
      gameCanvas.start();
      display.setCurrent(gameCanvas);
    } catch (Exception ex) {
      System.out.println(ex);
    }
  }

  public Display getDisplay() {
    return display;
  }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
    exit();
  }

  public void exit() {
    System.gc();
    destroyApp(false);
    notifyDestroyed();
  }
}
```

Simple Scrolling Background using LayerManager Screen Shot



Press the left or right game keys to view the scrolling effect.

j2meSalsa goodies

Execute Sample online	Execute on your browsers left frame.
Download demo code for deploying directly to WTK	Click here to download Zip File Containing WTK compatible file structure.

Back | Tutorial Home | Next

0 comments ★ 3

Leave a message...

Best Community Share ⚙

No one has commented yet.

Subscribe Add Disqus to your site

This page is a part of a frames based web site. If you have landed on this page from a search engine click here to view the complete page.