Wireless Game Development – Now and Future

Simon Ritter
Technology Evangelist
simon.ritter@sun.com

Sun™ Tech Days

Some Statistics

- Global video game market:
  - $28 billion in 2001 (wireless games $.0007b)
  - $30 billion in 2006 (wireless games $3.6b)
    (Source: Informa: Global Videogame Market)
- 7 million wireless gamers in 2002
  71.2 million wireless gamers in 2007
  (Source: IDC)

Agenda

- Challenges
- J2ME™ Platform for Game Development
- Summary and Resources

Device Resource Challenges

- CPU Power
- Screen Size
- Memory
  - Both static and dynamic
- Latency
  - For networked games

User Experience Challenges

- Rule #1: Games must be fun!
- Factors influencing ease of use
  - Screen size
  - Keypad size
  - Sound capabilities
  - Look and feel consistency
  - Latency
    - For network games

On-device Debugging Challenges

- On-device debugging is usually painful for small devices
- One can use emulators like the J2ME Wireless Toolkit
- Tip: Override the toString() method to print debugging information

MIDP 2.0 Game APIs

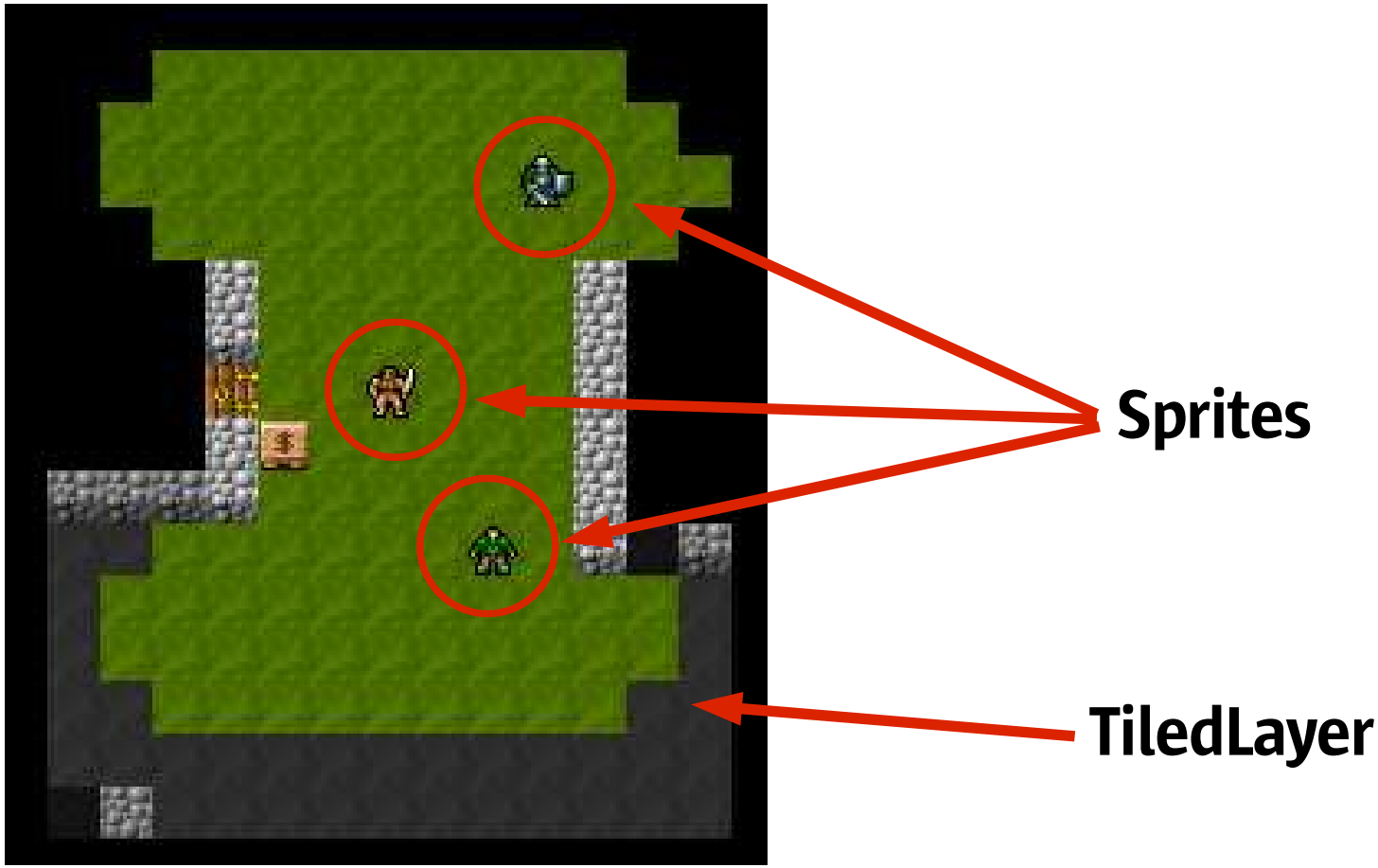
Sun™ Tech Days

Sun microsystems

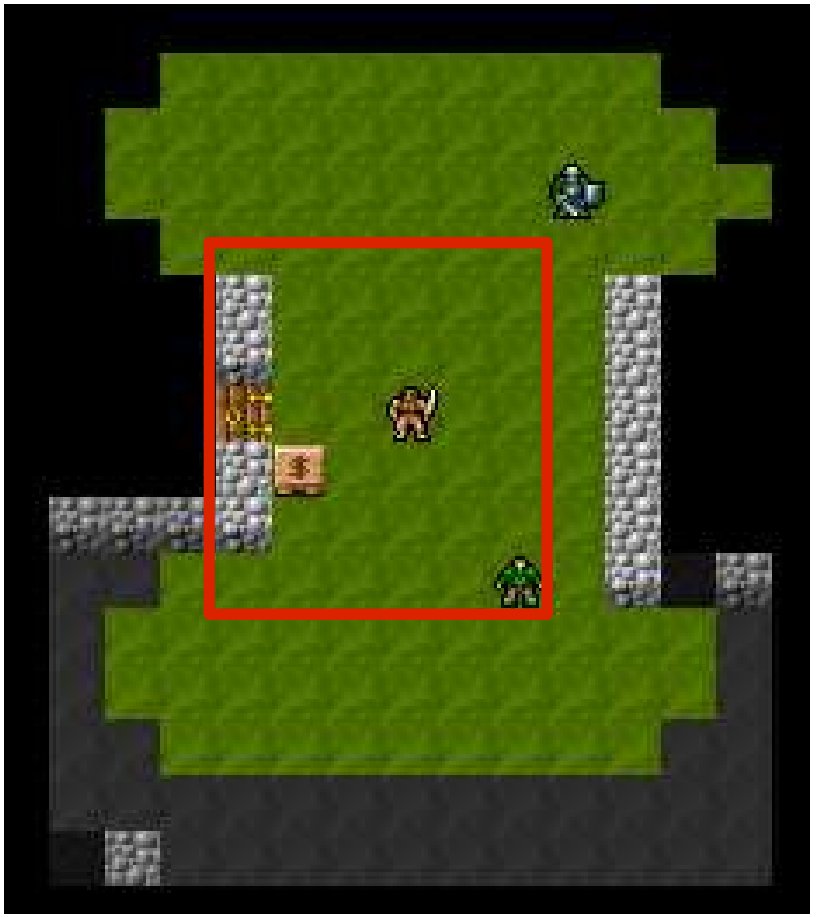
`javax.microedition.lcdui.game`

- This package provides a series of classes for the development of rich gaming content
- APIs are optimized by device manufacturers using native code, hardware acceleration, etc.
- Game APIs consist of:
  - Layer, Sprite, TiledLayer, LayerManager, GameCanvas

Step 1: Set Up a Game Screen



Step 2: Define a View

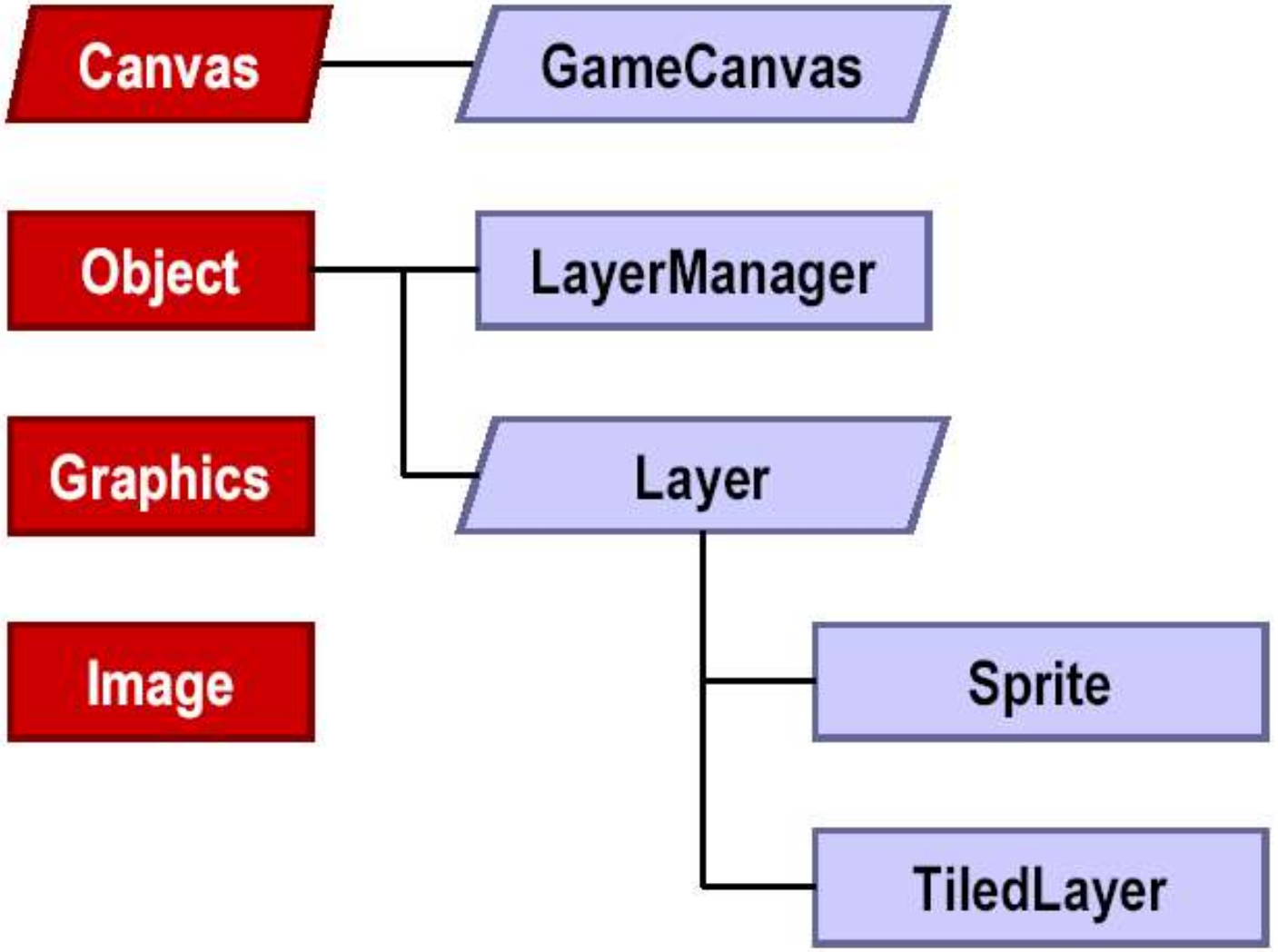


1. Add game world screen (TiledLayer) and sprites to LayerManager

2. Define a view port into the game world

Step 3: Render to the Screen

Game API Class Hierarchy



Game API Classes: Layer

Sun™ Tech Days

Sun microsystems

javax.microedition.lcdui.game.Layer

- Abstract class that represents a visual element of a game
- Layer subclasses must implement `paint`
- Layer methods:

```
int getHeight()
int getWidth()
int getX(), int getY()
boolean isVisible()
void move( int x, int y)
abstract void paint( Graphics g )
void setPosition( int x, int y )
void setVisible( boolean visible )
```

Game API Classes: Sprite

Sun™ Tech Days

Sun microsystems

javax.microedition.lcdui.game.Sprite

- Basic animated Layer that can display one of several graphical frames
- The frames are all of equal size and are provided by a single Image object
- In addition to animating the frames sequentially, a custom sequence can also be set in an arbitrary manner
- Also provides methods for transformation and collision detection

Game API Classes: TiledLayer

javax.microedition.lcdui.game.TiledLayer

- A Layer comprised of a grid of cells
- Each cell can display one of several tiles that are provided by a single Image object
- Cells can be filled with animated tiles, whose pixel data can be changed very rapidly
  - Very useful for animating large groups of cells such as areas of water

Game API Classes: LayerManager

Javax.microedition.lcdui.game.LayerManager

- Manages a series of Layers
- Automates rendering process for a game that employs many Layers
- Allows a developer to set a view window that represents a user's view of the game
- Provides APIs that control how the game's Layers are rendered on the Screen

Game API Classes: GameCanvas

- In MIDP 1.0, you needed three different thread contexts (application, input events, repainting)
- In MIDP 2.0, GameCanvas handles game loop:

```
public class MyGameCanvas extends GameCanvas
                                  implements Runnable {
  public void run() {
    Graphics g = getGraphics();

    for (;;) {
      updateGameState(getKeyStates());
      redrawScreenContents(g);
      flushGraphics();
      try {Thread.sleep(50);}
      catch (InterruptedException e) {/*do nothing*/}
    }
  }
}
```

Other MIDP 2.0 Features Useful for Game Development

- Painting polygons very quickly: Graphics.fillTriangle()
- Arbitrary images galore: Image.createRGBImage()

Graphics.fillTriangle() (1)


```
Public class Point {
  public int x, y, z, screen_x, screen_y;
  public void rotate(int yaw, int pitch, int roll) {/* ... */}
  public void translate(int xt, int yt, int zt) {/* ... */}
  public void scale(int n) {/* ... */}
}

public class Camera extends Point {
  public int yaw, pitch, roll;
}

public class Polygon {
  public Point[] plist;
  public int base_color, color, luminance;

  public Polygon(Point[] plist) {
    if (plist.length != 3 && plist.length != 4) {
      throw new IllegalArgumentException();
    }
    this.plist = plist;
  }

  public void shade(Point light) {/* ... */}
}
```

Graphics.fillTriangle() (2)

```
public class ThreeDObject {
  public Polygon[] polygon_list;
  public Point origin;
  public int radius;

  public ThreeDObject(DataInputStream in) {/* ... */}

  public void rotate(int yaw, int pitch, int roll) {/* ... */}
  public void translate(int xt, int yt, int zt) {/* ... */}
  public void scale(int n) {/* ... */}
  public void transfromToScreenCoordinates(Camera camera)
  {/* ... */}
}
```

Graphics.fillTriangle() (3)

```
public void render(Graphics g) {
  for (int i=0; i<object_list.size(); i++) {
    ThreeDObject object = (ThreeDObject)object_list.elementAt(i);
    object.transfromToScreenCoordinates(camera);
    for (int j=0; j<object.polygon_list.length; j++) {
      Polygon poly = object.polygon_list[j];
      poly.shade(light);
      g.setColor(poly.color);
      g.fillTriangle(
        poly.plist[0].screen_x, poly.plist[0].screen_y,
        poly.plist[1].screen_x, poly.plist[1].screen_y,
        poly.plist[2].screen_x, poly.plist[2].screen_y
      );

      if (poly.point_list.length == 4) {
        g.fillTriangle(
          poly.plist[2].screen_x, poly.plist[2].screen_y,
          poly.plist[3].screen_x, poly.plist[3].screen_y,
          poly.plist[0].screen_x, poly.plist[0].screen_y
        );
      }
    }
  }
}
```

Image.createRGBImage()

```
Image.createRGBImage( int[] data,
                      int width,
                      int height,
                      boolean alpha)
```

- Lets you do cool stuff:
  - Texture Mapping
  - Ray Tracing
- Watch out for garbage

Mobile Media APIs (JSR-135)

- Mobile Media APIs (MMA) specifies a small footprint multimedia API for J2ME
- Allows for both audio and video multimedia resources
  - Addresses scalability and support for more advanced features
- Can be used to enhance the game experience with rich multimedia content
- Built-in support in the J2ME™ Wireless Toolkit 2.0 (WTK)

Media Types Supported in WTK

- WTK 2.0 supports the following media formats for MMA development:
  - Audio: PCM and WAV audio
  - MIDI: MIDI (Type 0 and Type 1), SP-MIDI
  - Video : MPEG-1
  - Audio Capture: Typical audio capture from Solaris, Windows, and Linux platforms

Example: MIME Types Supported via MMA in Nokia 3650

Sun™ Tech Days

Sun microsystems

- video/3gpp (3G video standard)
- video/vnd.nokia.interleaved-multimedia (Nokia NIM video)
- audio/x-wav (WAV, various encodings)
- audio/midi and audio/sp-midi
- audio/amr (speech encoding)
- audio/x-nokia-rng (Nokia ringing tones)
- audio/x-tone-seq (Content type returned by tone player)

Summary

- Wireless applications, games included, face challenges beyond desktop and server apps
  - Network latency, device power, form factor
- Think simple, user-friendly, and extensible
- Design for tomorrow, implement today
- Use proven technologies and design patterns
- Test, test, and test...

Coming Soon to a Device Near You

- JCP developed specifications will continue to expose additional capabilities in the near future:
  - Java APIs for Bluetooth (JSR 82)
  - CLDC 1.1 (JSR 139)
  - Location API for J2ME (JSR 179)
  - Mobile 3D Graphics APIs for J2ME (JSR 184)
  - J2EE Client Provisioning (JSR 124)

Resources

- Download the Sun J2ME Wireless Toolkit: java.sun.com/products/j2mewtoolkit
- Sun's wireless blueprints: http://java.sun.com/blueprints/wireless/
- Wireless gaming blueprints: http://java.sun.com/blueprints/code/index.html#games
- Kay Neunhofen's article discussing the details of the wireless gaming blueprint applications: http://wireless.java.sun.com/blueprints/articles/game/
- Java™ Games in the java.net community: http://community.java.net/games

Additional Resources

- kvm-interest mailing list archive: archives.java.sun.com/kvm-interest.html
- J2ME Platform and Wireless Webcasts: java.sun.com/jdc/onlineTraining/webcasts
- “The Nokia 3650 Mobile Media API” and “Designing Single-Player Games” papers: forum.nokia.com

Simon Ritter

Technology Evangelist

simon.ritter@sun.com

Sun™ Tech Days