Overview

- Display System Architecture
- Graphics Devices, Environments & Display Modes
- Full-Screen Exclusive Mode (FSEM)
- Rendering System Architecture

Display System Organization

```
Game Application

Create, Use

<IDisplaySystem>

DisplaySystem

Canvas

JFrame

Renderer

Camera
```

Creating a DisplaySystem

// Creates a DisplaySystem for use by the game. Since display creation may involve separate
// threads whose timing is unknown, this method waits until either display creation
// is complete or a timeout occurs.

private IDisplaySystem createDisplaySystem() {
    IDisplaySystem display = new MyDisplaySystem();
    display.createDisplaySystem();
    return display;
}
```

DisplaySystem Interface

```
interface IDisplaySystem
{
    int getWidth();
    int getHeight();
    int getBitDepth();
    int getRefreshRate();
    void setTitle(String title);
    IRenderer getRenderer();
    boolean isCreated();
    boolean isFullScreen();
    void addKeyListener(KeyListener k);
    void addMouseListener(MouseListener m);
    void setPredefinedCursor(int cursor);
    void setCustomCursor(String fileName);
    ...
}
```

SAGE InitSystem() Functions

```
protected void initSystem() {
    initDisplaySystem();
    initInputManager();
    initGameWorld();
   ...
}
```
Graphics Devices

Output devices are managed by objects of type GraphicsDevice

- AWT/Swing
  - Component
  - Graphics2D - draws on Screen

- JOGL (OpenGL)
  - GLDrawable - draws on Screen

Graphics Environment:
- Holds the collection of current GraphicsDevice objects
- GraphicsDevice
- DisplayMode - Configuration

Display Mode:
- Characteristics of devices:
  - Width, Height, Depth (bits per pixel), Refresh Rate
- Encapsulated by Java class DisplayMode
- Display Mode normally controlled by the Window Manager (WM)

Managing Display Mode

- Obtaining current mode:
  DisplayMode curMode = device.getDisplayMode();
- Obtaining all supported modes:
  DisplayMode[] modes = device.getDisplayModes();
- SAGE has a user-selection component:
  sage.display.DisplaySettingsDialog

Full-Screen Exclusive Mode

- “FSEM” - special mode of Window Managers
  - Gives program direct, exclusive control of screen
  - Allows program to change DisplayMode (if change is supported by OS/hardware)
- Java AWT FSEM applications should:
  - setResizable(false);
  - setUndecorated(true);
  - setIgnoreRepaint(true);
- Windows JOGL applications:
  - Pass -Dsun.java2d.d3d=false to JVM

DisplaySystem Construction

// This class defines a DisplaySystem that can be run in windowed or FSEM public class MyDisplaySystem implements IDisplaySystem {
  private JFrame  myFrame; 
  private GraphicsDevice device; 
  private IRenderer myRenderer; 
  private int width, height, bitDepth, refreshRate; 
  private Canvas rendererCanvas; 
  private boolean isCreated; 
  private boolean isFullScreen; 

  // constructor
  public MyDisplaySystem(int w, int h, int depth, int rate, boolean isFS, String rName) {
    //save the input parameters for accessor queries
    width = w;  height = h;  bitDepth = depth; refreshRate = rate;
    this.isFullScreen = isFS;
    //get a renderer from the RendererFactory
    myRenderer = RendererFactory.createRenderer(rName);
    if (myRenderer == null) {
      throw new RuntimeException("Unable to find renderer "+ rendererClassName);
    }
    rendererCanvas = myRenderer.getCanvas();
    myFrame = new JFrame("Default Title");
    myFrame.add(rendererCanvas);
    //initialize the screen with the specified parameters
    DisplayMode displayMode = new DisplayMode(width, height, bitDepth, refreshRate);
    initScreen(displayMode, isFullScreen);
    //save DisplaySystem, show the frame and indicate DisplaySystem is created
    DisplaySystem.setCurrentDisplaySystem(this);
    myFrame.setVisible(true);
    isCreated = true;
  }
  //continued...
Screen Initialization

```java
private void initScreen (DisplayMode dispMode, boolean fullScreenRequested) {
    // open default screen device out of the local graphics environment:
    DisplaySystem localGraphicsEnvironment = GraphicsEnvironment.getLocalGraphicsEnvironment();
    device = localGraphicsEnvironment.getDefaultScreenDevice();

    // queue manipulation methods
    RenderQueue queue = device.getRenderQueue();
    Canvas canvas = device.getCanvas();

    // renderQueue operations, such as renderObject operations
    queue.addRenderObject (new Triangle (0, 0, 0, 10, 0, 0));
    queue.addRenderObject (new Cylinder (0, 0, 0, 10, 0, 0));
    queue.addRenderObject (new Cube (0, 0, 0, 10, 0, 0));
    queue.addRenderObject (new Sphere (0, 0, 0, 10));
    queue.addRenderObject (new HUDstring ("Hello World"));
    queue.addRenderObject (new Polyhedron3D (0, 0, 0, 10, 0, 0));
    queue.addRenderObject (new Rectangle (0, 0, 10, 10));

    // render draw() point methods
    (int) queue.addRenderObject (new Polyhedron3D (0, 0, 0, 10, 0, 0));
    queue.addRenderObject (new Rectangle (0, 0, 10, 10));

    // render callback methods
    RenderQueue.addRenderObject (new Polyhedron3D (0, 0, 0, 10, 0, 0));
    RenderQueue.addRenderObject (new Rectangle (0, 0, 10, 10));

    // render graphics calls are encapsulated inside a
    SetFrame.setFrame (canvas, queue, fullScreenRequested);
    setDefaultScreenDevice (device); // queue set characteristics
    DisplayMode dispMode = device.getDefaultDisplayMode();
    dispMode = GraphicsEnvironment.
    getDisplayMode (device); // queue set characteristics
    device.setDisplayMode (dispMode); // queue set characteristics
    device.setFullScreenWindow (myFrame); // queue set characteristics
    myFrame.setIgnoreRepaint (true); // queue set characteristics
    myFrame.setResizable (false); // queue set characteristics

    try {
        device.setFullScreenWindow (myFrame); // queue set characteristics
    } catch (Exception ex) {
        System.err.println ("Cannot set display mode");
    } else {
        device.setFullScreenWindow (myFrame); // queue set characteristics
    }
}
```

Exiting FSEM

```java
// overriding BaseGame.shutdown() to accomplish orderly exit
protected void shutdown () {
    // exit the display system to shutdown
    display.close ();
}
```

Rendering System

Create Display, Get Renderer

```java
RenderingSystem

DisplaySystem

<abstract> Renderer

Canvas

Camera

Renderer Implementation

<abstract> IRenderer

// Game Application

Fetch objects, Manipulates cameras

Calls draw() on objects.

Isolating Graphical Operations

ALL graphics calls are encapsulated inside a

System-specific renderer

Renderer draw() Callbacks

Problem:
Renderer should be able to make polymorphic draw() calls on objects.
draw() in objects cannot contain graphics-specific methods
Solution:
- Renderer supports `draw()`
  - For each engine-defined drawable-type
  - In a graphics-specific way
- Renderer makes polymorphic `draw()` calls passing itself to the object
- `draw()` in each object calls back to the Renderer specifying its TYPE

```java
Renderer
GameObject
  draw(Renderer r) {...}
  draw(Poly3D p) {...}
  draw(Rectangle r) {...}
  draw(Teapot t) {...}
...```

Camera System

```
Game Application
  DisplaySystem
    Renderer
      DisplaySystem
        IRenderer
          AbstractCamera
            JOGLCamera
...```

Provides functions common to all cameras
Provides graphics API-specific camera functions

`# location` `# U,V,N` `# fovY, aspect, near, far`