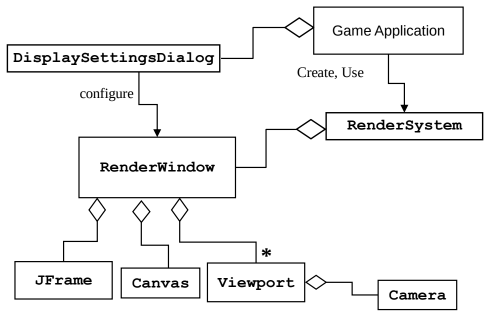
CSc 165
Computer Game Architecture

5 - Displays & Rendering

Render System Organization



RenderSystem Interface

```
//  Every DisplaySystem includes physical parameters such as width and height.
//  DisplaySystems also have a Renderer which knows how to draw SceneNodes
//  on the display.

interface DisplaySystem
{   createGpuShaderProgram();
    createRenderQueue();
    createRenderWindow();
    processRenderQueue();
    setActiveLights();
    setHUD();

    ... // etc.

    including some private details, such as swapBuffers();
}
```

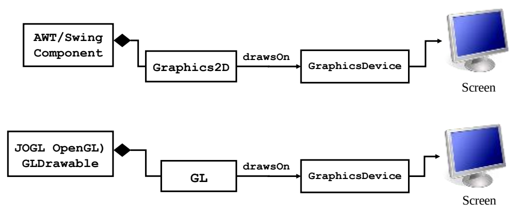
RenderWindow Interface

```
interface RenderWindow
{   ...
    createViewport();
    removeViewport();
    setTitle();
    setVisible;

    ... // etc.

    including some accessors, such as getHeight();
}


// constructor in GL4RenderWindow only:
    GL4RenderWindow(canvas, DisplayMode, fullScreen);
```

a Java class

Graphics Devices

Output devices are managed by objects of (Java) type **GraphicsDevice**



Graphics Devices (cont.)

GraphicsEnvironment holds the collection of current **GraphicsDevice** objects

Graphics-Configuration:
- *image capabilities,*
- *buffer capabilities,*
- *color models supported,*

etc.

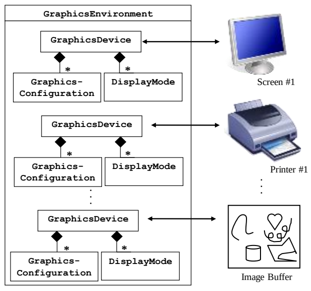
Display Mode:
- *display size,*
- *bit depth,*
- *refresh rate,*

etc.

Graphics environment:
- *Graphics Devices,*
- *fonts,*

etc.

see:
java.awt.GraphicsDevice



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 `DisplayMode`

- Obtaining current mode:

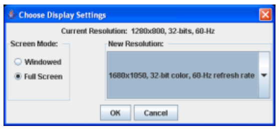
```
DisplayMode curMode = device.getDisplayMode();
```

- Obtaining all supported modes:

```
DisplayMode [] modes = device.getDisplayModes();
```

- User-selection tool available on homework page:

DisplaySettingsDialog:


Choose Display Settings
Current Resolution: 1280x800, 32-bits, 60-Hz
Screen Mode:
Windowed
[✓ Full Screen]
New Resolution:
1680x1050, 32-bit color, 60-Hz refresh rate
OK
Cancel

in setupWindow

```
GraphicsDevice gd = ge.getDefaultScreenDevice();
DisplaySettingsDialog dsd = new DisplaySettingsDialog(ge.getDefaultScreenDevice());
dsd.showIt();
RenderWindow rw = rs.createRenderWindow(dsd.getSelectedDisplayMode(),
                                        dsd.isFullScreenModeSelected());
```

Full-Screen Exclusive Mode

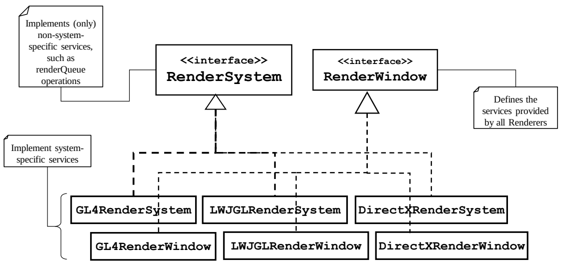
- "*FSEM*": special mode of Window Managers
  - Gives program direct, exclusive control of screen
  - Allows program to change DisplayMode
    (if change is supported by OS/hardware)
- Java AWT FSEM applications should:
  - `setResizable(false);`
  - `setUndecorated(true);`
  - `setIgnoreRepaint(true);`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

Screen Initialization

```
private void tryFullScreenMode(GraphicsDevice gd, DisplayMode dispMode)
{   if (gd.isFullScreenSupported())
    {  frame.setUndecorated(true);
       frame.setResizable(false);

       // AWT repaint events unecessary – we manage render loop
       frame.setIgnoreRepaint(true);
       gd.setFullScreenWindow(frame);

    if (gd.isDisplayChangeSupported())
    { try
      {  gd.setDisplayMode(dispMode);
         frame.setSize(dispMode.getWidth(), dispMode.getHeight());
         isInFullScreenMode = true;
      } catch (IllegalArgumentException e)
      {  frame.setUndecorated(false);
         frame.setResizable(true);
    }} else {
         logger.fine("FSEM not supported");
    }} else {
         frame.setUndecorated(false);
         frame.setResizable(true);
         frame.setSize(dispMode.getWidth(), dispMode.getHeight());
         frame.setLocationRelativeTo(null);
    }
}
```

Isolating Graphical Operations

Graphics functions are encapsulated inside system-specific implementations


Implements (only) non-system-specific services, such as renderQueue operations
<<interface>> RenderSystem
<<interface>> RenderWindow
Defines the services provided by all Renderers
Implement system-specific services
GL4RenderSystem
LWJGLRenderSystem
DirectXRenderSystem
GL4RenderWindow
LWJGLRenderWindow
DirectXRenderWindow

Rendering

in RenderSystem:

```
public void display(GLAutoDrawable glad)
{ . . .
  for (Renderable r : renderQueue)
  {  GpuShaderProgram program = r.getGpuShaderProgram();
     setRenderStates(r);
     GpuShaderProgram.Context ctx = program.createContext();
     ctx.setRenderable(r);
     tx.setViewMatrix(viewMatrix);
     ctx.setProjectionMatrix(projMatrix);
     ctx.setLightsList(lightsList);
     ctx.setAmbientLight(ambientLight);
     program.bind();
     drawRenderable(gl, r);
     program.unbind();
  } . . .
}
```

GPU

Vertex shader:

```
void main()
{  out_position = projMatrix * viewMatrix * vertex;
   out.texcoord = in_texcoord;
   out_normal = invTrnsp(viewMatrix) * in_normal;
}
```

Fragment shader:

```
void main()
{  for (int i = 0; i < lights.length(); ++i)
        effect += get_light_effect(lights[i], material);
   fragment = texture2D(texture_sampler, in.position) * effect;
}
```