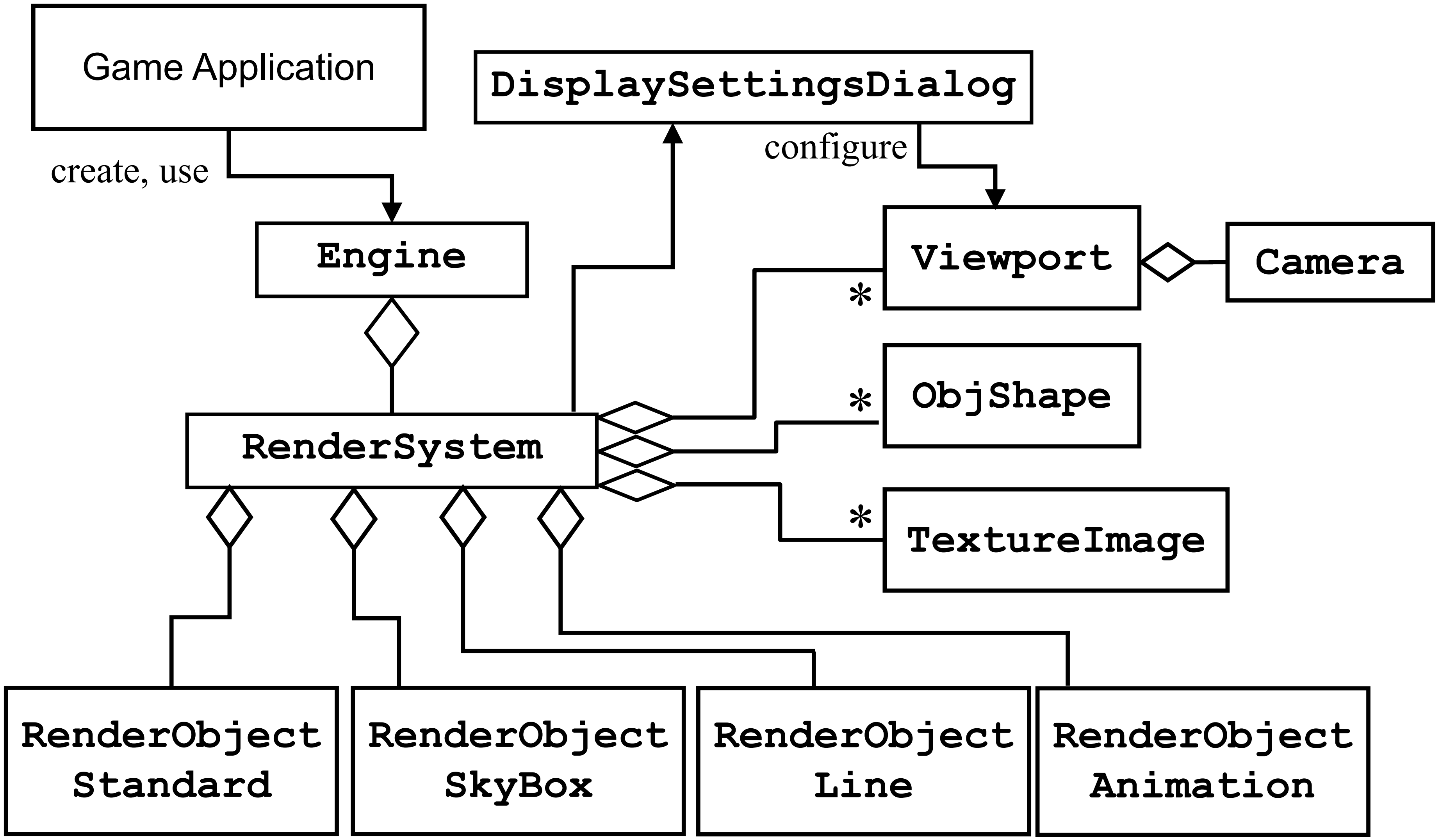
CSc 165

Computer Game Architecture

05 - Displays & Rendering

TAGE Render System Organization



RenderSystem public methods:

```
getGLCanvas() {...}
toggleFullScreenMode() {...}
addViewport() {...}
startGameLoop() {...}
getHeightAt(texture, x, z) {...}
```

methods used by JOGL (OpenGL):

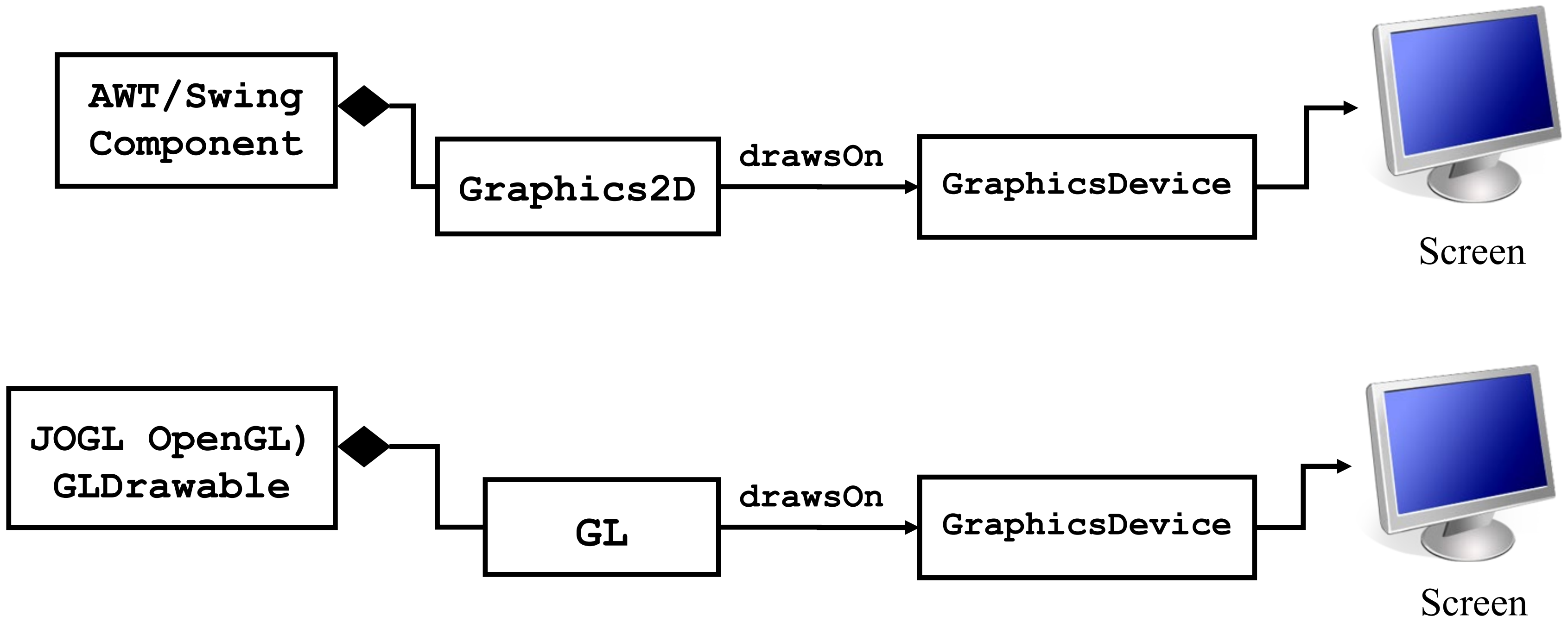
```
init() {...}
display() {...}
```

methods used internally:

```
setUpCanvas() {...}
loadVBOs() {...}
loadTextures() {...}
```

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.

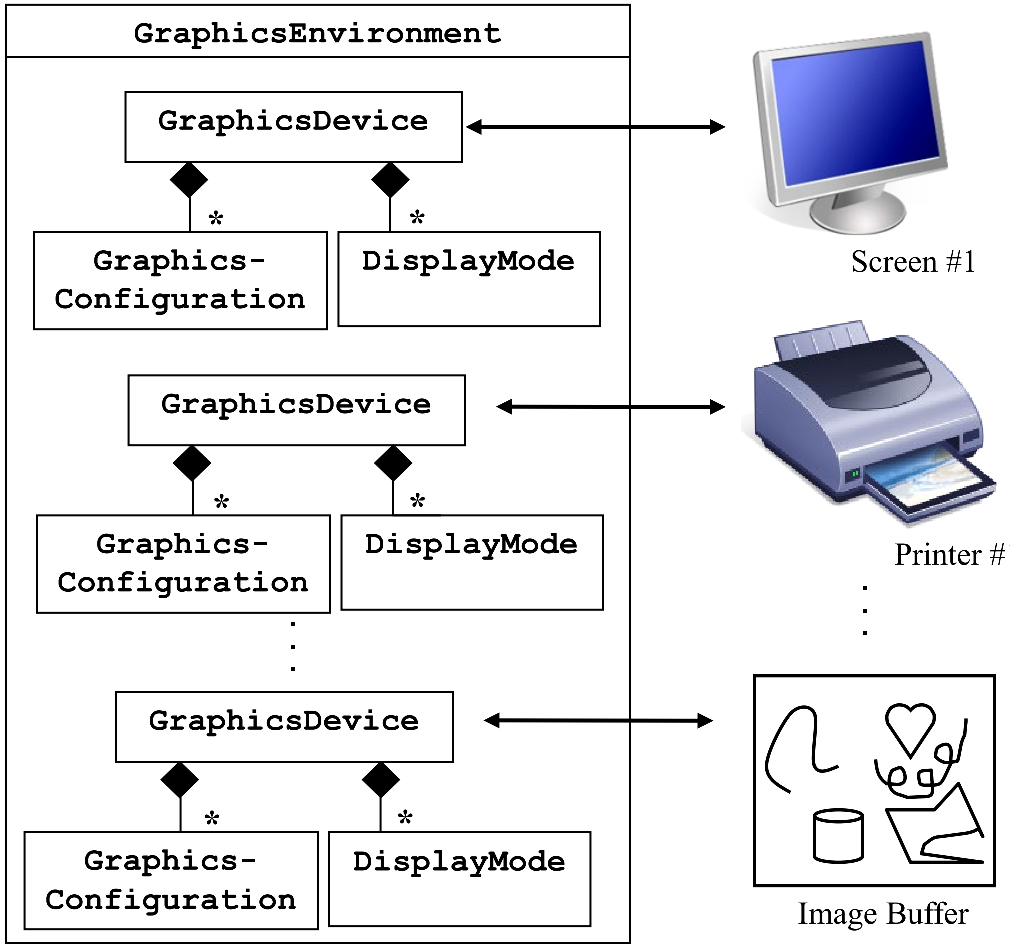
Display Mode:

- *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();
```

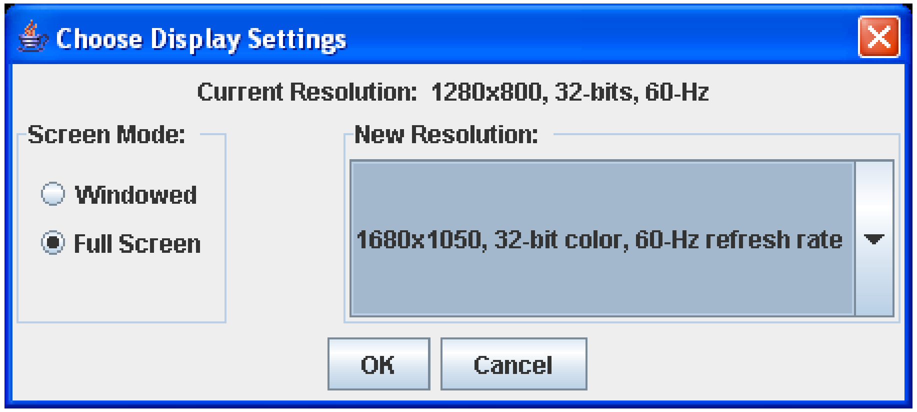
- Obtaining all supported modes:

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

- User-selection tool available on homework page:

***DisplaySettingsDialog*:**

in ***RenderSystem*** *constructor*



Current Resolution: 1280x800, 32-bits, 60-Hz

[✓ Full Screen]

1680x1050, 32-bit color, 60-Hz refresh rate

```
GraphicsDevice gd = ge.getDefaultScreenDevice();
DisplaySettingsDialog dsd = new DisplaySettingsDialog(ge.getDefaultScreenDevice());
dsd.showIt();
```

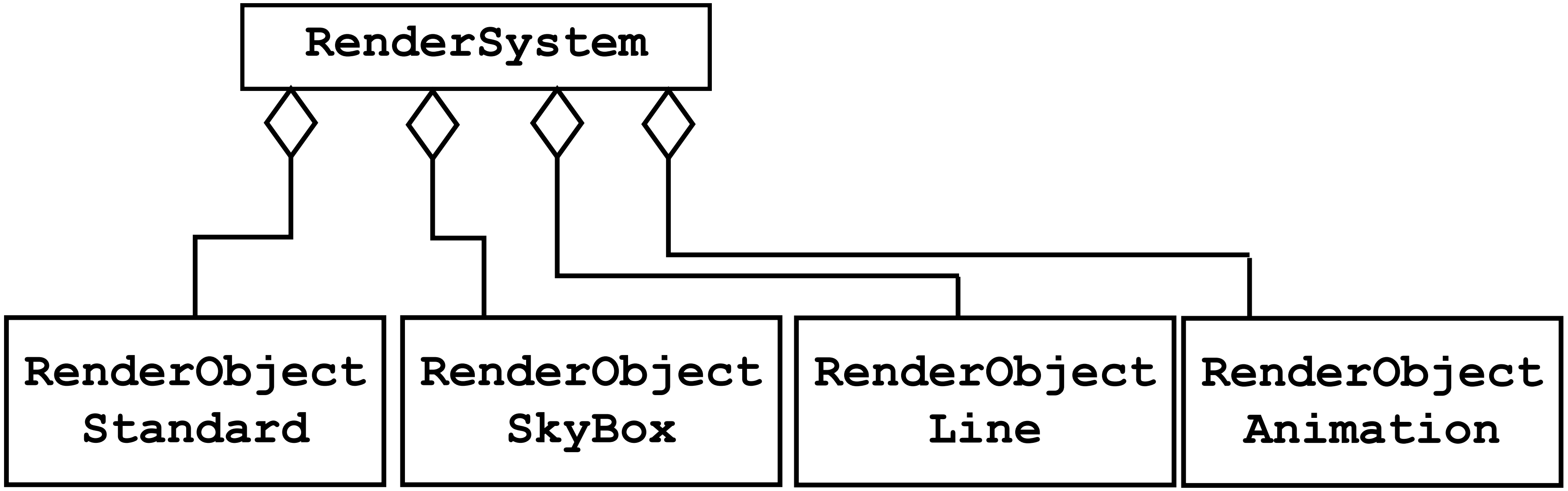
Full-Screen Exclusive Mode

- "*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`**
    `-Dsun.java2d.uiScale=1` to JVM

Screen Initialization

```
private void tryFullScreenMode(GraphicsDevice gd, DisplayMode dispMode)
{   isInFullScreenMode = false;
    if (gd.isFullScreenSupported())
    {   this.setUndecorated(true);
        this.setResizable(false);
        this.setIgnoreRepaint(true); // AWT repaint events ignored
        gd.setFullScreenWindow(this);
        if (gd.isDisplayChangeSupported())
        {   try
            {   gd.setDisplayMode(dispMode);
                this.setSize(dispMode.getWidth(), dispMode.getHeight());
                isInFullScreenMode = true;
            }
            catch (IllegalArgumentException e)
            {   System.out.println(e.getLocalizedMessage());
                this.setUndecorated(false);
                this.setResizable(true);
        }   }
        else
        {   System.out.println("FSEM not supported");
    }   }
    else
    {   this.setUndecorated(false);
        this.setResizable(true);
        this.setSize(dispMode.getWidth(), dispMode.getHeight());
        this.setLocationRelativeTo(null);
}   }
```

Specialized object renderers handle direct communication with shaders



Cube, Sphere, Plane, Torus, imported models, manual objects, roomBox, terrain. (with or without lighting)

Skybox only. There is never more than one skybox to render. Lighting not used.

Line only. Lighting not used.

Animated objects. Handles rendering an object with one or more animation, (with or without lighting)

Rendering

in RenderSystem:

```
public void display(GLAutoDrawable glad)
{ . . .
  for (Viewport vp : viewportList.values())
  { ● get view matrix from viewport's camera
    ● build perspective matrix based on viewport dimensions
    ● render skybox by calling render() in RenderObjectSkyBox
    ● draw HUD(s) using HUDmanager
    ● build render queue vector of objects in the scenegraph
    for (int i=0; i<q.size(); i++)
    {  GameObject go = q.get(i);
       ● if line, call render() in RenderObjectLine
       ● if animated, call render() in RenderObjectAnimated
       ● else call render() in RenderObjectStandard
} } }
```

For details on the specialized object renderers, and the shaders, take CSc-155!