Useless Fact of the Day

- You could stuff 1,300 Earths into Jupiter, and about 1,000,000 Earths into the Sun.
- At the equator, the Earth’s surface is rotating at 1,532 feet per second (about 1,000 miles per hour). Earth is also zooming in its orbit around the sun at over 18 miles per second.

source: www.space.com

Topics

- Graphics Hardware
- Coordinate Systems
- Shapes
- GeneralPath and CAG

Hardware

- Monitor
- **Pixels** (little squares)
- Common **resolutions**: 640x480 (in the 1990s), 800x600, 1024x768, 1280x1024
- Ratio of 1 and 1/3
- Widescreen Resolutions normally have a ratio of 1.6
  - “Golden Ratio” is about 1.618033989
  - The “most aesthetically pleasing rectangle” is the golden rectangle
Hardware

- Monitor
  - **Bit depth**: how many bits are used to code for the color of each pixel?
    - Common: 8, 15, 16, 24, 32
    - # of colors possible: \(2^m\)
      - 8-bit: \(2^8 = 256\)
      - 16-bit: \(65536\)
      - 24-bit: \(16.7\) million
  - **Refresh rate**: how many times per second does the monitor redraw the screen?
    - Common: 75-85 Hz

Coordinate Systems

- Cartesian - real-valued
- Polar - real-valued
- Screen - integer-valued
- FANG Engine - real-valued

FANG Engine Coordinates

- The FANG Engine has a frame rate of about 24 frames per second (same as a movie projector)
- FANG Engine stores location and size in fractions of the canvas width/height

**Note**: The smaller dimension ranges from 0 at one extreme of the visible screen to 1.0 at the other, while the larger dimension ranges from 0 to something greater than 1.0. For example, if the visible portion of the canvas is twice as wide as it is tall, the coordinate system looks like this:

- We can resize the game window and not have to worry about changing the locations of Sprites (no matter what size the window -- as long as it was square -- if we want the sprite in the middle of the screen, we could put it at location 0.5, 0.5)
- The coordinate system allows for sub-pixel accuracy in position (unlike screen coordinates)
- Why is that important?
  - If we stored position with only integer pixels (no fractions of pixels), the slowest an object could move is 1 pixel per frame (which means the object would take 20 seconds to traverse the height of a 480-pixel canvas)
  - But what if you wanted an Unnecessarily Slow Moving Dipping Mechanism?
  - Also reduces round-off error, and it is simpler to do calculations (scaling/rotation/movement) with floating-point numbers than integers
Shape Sprites

- EllipseSprite
  - constructed with (width, height)
- PieSprite
  - constructed with (width, height, start, end)
- RectangleSprite
  - constructed with (width, height)
- LineSprite
  - constructed with (startX, startY, endX, endY)
- ArcSprite
  - constructed with (width, height, start, end)

Shapes

- User-defined Shape-making classes (not part of the FANG engine; in the java.awt.geom package):
  - GeneralPath
  - Lines
  - Curves (Quadratic and Cubic)
  - Area
    - Constructive Area Geometry (CAG)

GeneralPath

- This is for drawing out shapes using lines.
- “moveTo” means “pick up the pen and move it to this location, without drawing anything”
- “lineTo” means “draw a line from where we are now to this new location”
- “closePath” means “draw a line from where we are now to where we started drawing”

- Making a path representing a triangle:
  ```java
  GeneralPath path = new GeneralPath();
  path.moveTo((float) 0.0, (float) 0.0);
  path.lineTo((float) 1.0, (float) Math.sqrt(3));
  path.lineTo((float) -1.0, (float) Math.sqrt(3));
  path.closePath();
  ```
- Making a Sprite with the triangle path:
  ```java
  Sprite triangleSprite;
  triangleSprite = new Sprite(path);
  ```

Constructive Area Geometry (CAG)

- Used for creating a shape out of adding and subtracting other shapes
- For example, you could create a rectangle with three holes in it (a traffic light type thing):

```
add subtract subtract subtract subtract
```
Constructive Area Geometry (CAG) Example

• Making a rectangle with three holes in it:

```java
Area area = new Area();
Rectangle box = new Rectangle(0, 0, 20, 60);
area.add(new Area(box));
Ellipse2D.Double circle = new Ellipse2D.Double(2, 2, 16, 16);
area.subtract(new Area(circle));
circle = new Ellipse2D.Double(2, 22, 16, 16);
area.subtract(new Area(circle));
circle = new Ellipse2D.Double(2, 42, 16, 16);
area.subtract(new Area(circle));
```

• Making a Sprite with the Area:

```java
Sprite trafficLightSprite;
trafficLightSprite = new Sprite(area);
```