CMSC131

Basic 2D Graphics: Part I

Raster Graphics

For logistical reasons (such as speed) raster graphics, which is basically taking an image drawn using things like geometric shapes and converting it “down” to a rectangular (two dimension) **pixel**-based representations were (and still are) very popular in some uses.

- Easily displayed on simple computer screens.
- Easy/possible to print on early printers.
- Quick to read from a data file and use because there’s no post-processing, though they cause the files to be larger which adds to physical read or transmission time.

Vector and Other Graphics

While vector graphics and different common graphics formats (such as JPG) and even some less common formats (such as HEIC) are utilized behind the scenes, to be displayed on your screen or edited in many image manipulation programs they effectively might need to become pixel-based.

Scaling, Adding 3D

Some representations (like TrueType fonts or vector graphics) scale well to any size while others (GIF, JPEG) do not (though there are tricks to improve the results.

Adding a third dimension can be done in a wide variety of ways. We might have a cube (three dimensions) of **voxels**, but there are many other ways.

Graphics in Java

We are using Java's **Color** class and some local-made classes to support storing color information in image grids for simplicity in this first course but the concepts apply to any pixel-based library.

- **Grid_3x5** and **SquareGrid** (self-contained classes that uses Java's **Color** class and where individual pixels are drawn in a "zoomed in" view to more easily spot errors)
- **Pixel** and **Photograph** (in P4, classes that will allow us to easily create our own **RGB** colors and allows us to easily modify a pixel-based image that can then be used by existing Java graphics libraries behind the scenes).

Grid_3x5 and **SquareGrid** classes

When you create a new **`Grid_3x5`** or **`SquareGrid`** object you will pass a logical size or scale to the constructor. The grid's dimensions cannot be altered once the object is created.

- NOTE: Once you create a **`Grid_3x5`** or **`SquareGrid`** object it will automatically display itself on-screen and any changes to it will be automatically displayed as well.

Grid_3x5 and **SquareGrid** methods

The following instance methods are available:

- **`getHt()`**: returns an **`int`** representing its logical height
- **`getWd()`**: returns an **`int`** representing its logical width
- **`getColor(int row, int col)`**: returns a reference to the **`Color`** in the specified logical row/col position
- **`setColor(int row, int col, Color c)`**: sets the **`Color`** in the specified logical row/col position

Simple **SquareGrid** Pattern: Border

```
for (int col=0; col<grid.getWd(); col++) { //top line
   grid.setColor(0, col, Color.RED);
}
for (int col=0; col<grid.getWd(); col++) { //bottom line
  grid.setColor(grid.getHt()-1, col, Color.RED);
}
for (int row=0; row<grid.getHt(); row++) {//left side
  grid.setColor(row, 0, Color.RED);
}
for (int row=0; row<grid.getHt(); row++) {//right side
  grid.setColor(row, grid.getWd()-1, Color.RED);
}
```