Beginning Java for Android

Session 3: Drawing—a high tech Pre-K class

Izzy Johnston

izzycjohnston@gmail.com

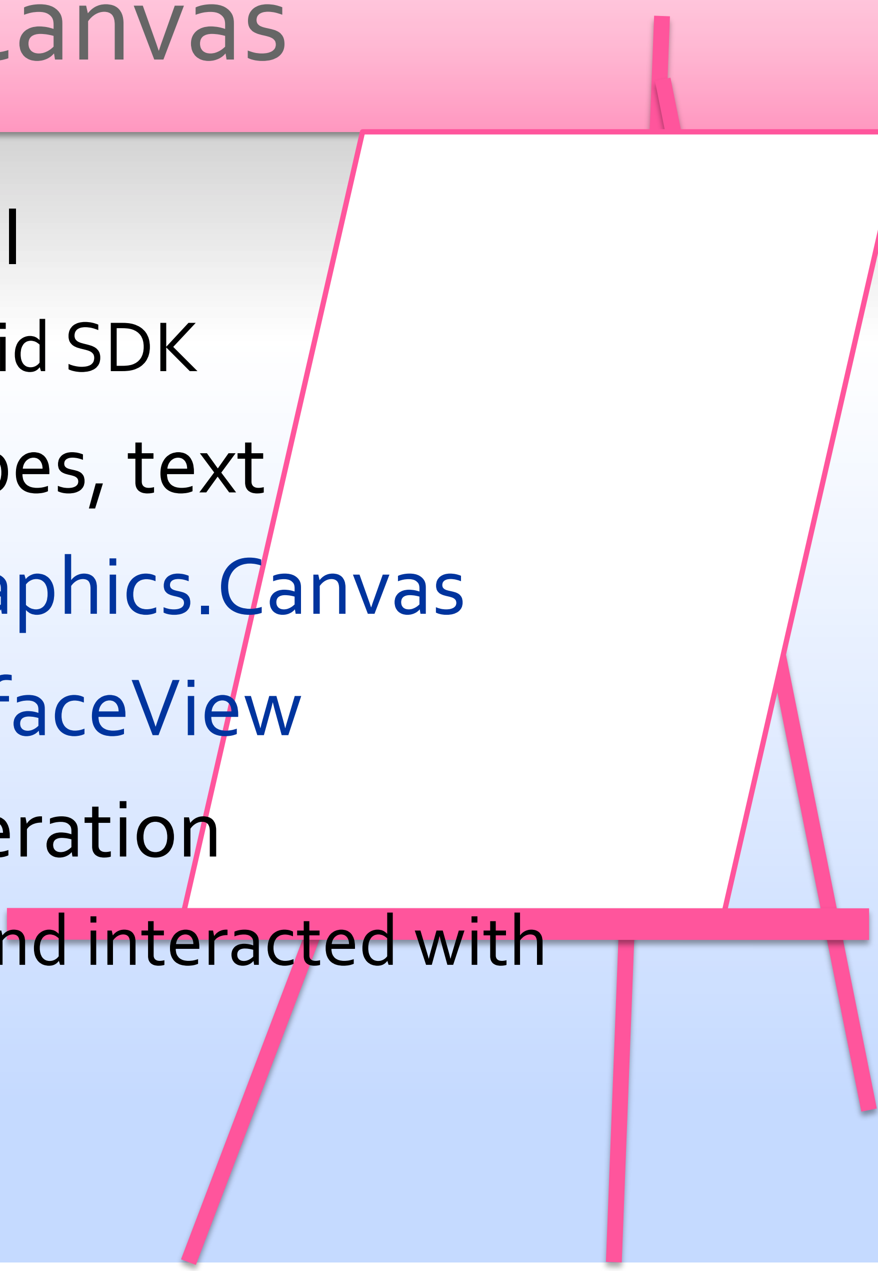
@izzy_johnston

Quick Review

- Object Oriented Programming
- Classes, Methods, States
- Android SDK
- Getting User Data
- Giving User Data
- Math class
- String class

Canvas

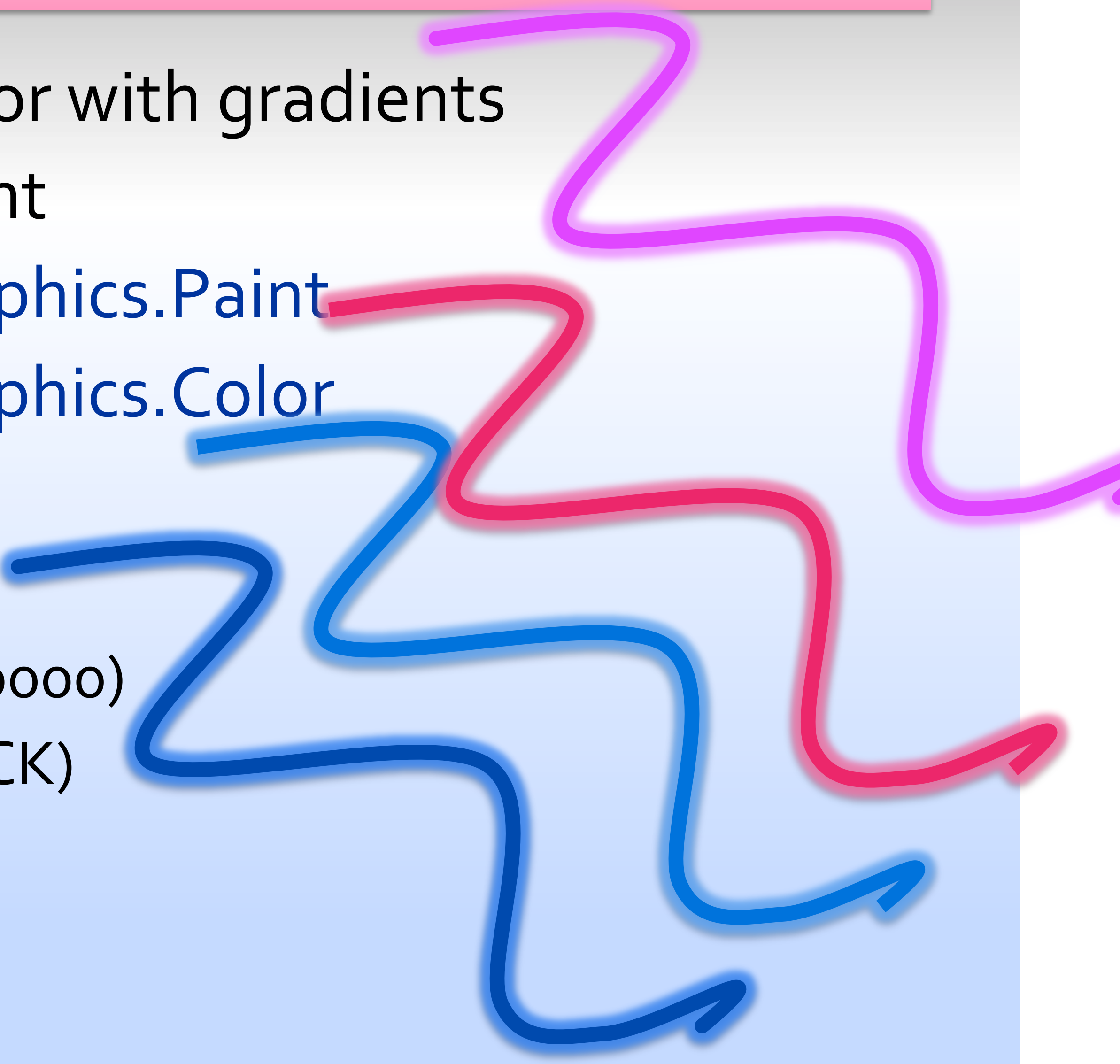
- Class from Java API
  - updated for Android SDK
- Draw images, shapes, text
- Import android.graphics.Canvas
- On a View or a SurfaceView
- Allows constant iteration
  - can be animated and interacted with



Paint

- Color objects solid or with gradients
- Style stroke and font
- import android.graphics.Paint
- import android.graphics.Color

- Color Types
  – Hexadecimal (#000000)
  – Integer (Color.BLACK)
  – RGB (0, 0, 0)



What shapes can we draw?

Circle---

```
drawCircle(int xPosition, int yPosition, int radius, Paint paint);
```

Color---

```
drawColor(int color);
```

Line---

```
drawLine(float startX, float startY, float stopX, float stopY, Paint paint);
```

Point---

```
drawPoint (float x, float y, Paint paint);
```

Rectangle---

```
drawRect (float xLeft, float yTop, float xRight, float yBottom, Paint paint);
```

Bitmap---

```
drawBitmap(Bitmap bitmap, float xLeft, float yTop, Paint paint);
```

Draw Shapes on Canvas

- Create new Android application, called ShapesCanvas
  - com.gdi.shapescanvas
  - Default activity ShapeCanvas

Creating a new class

```
private class Rectangle extends View{
   private final float x;
   private final float y;

  private Paint rectPaint = new Paint(Paint.ANTI_ALIAS_FLAG);

public Rectangle(Context context, float x, float y) {
         super(context);

         this.x = x;
         this.y = y;
}
```

Implementing Canvas

```
@Override
   protected void onDraw(Canvas canvas) {
      super.onDraw(canvas);

      float centerX=canvas.getWidth()/2;
      float centerY=canvas.getWidth()/2;
       int topX=(int)(centerX-(x/2));
       int topY=(int)(centerY-(y/2));
       int botX=(int)(centerX+(x/2));
       int botY=(int)(centerY+(y/2));

rectPaint.setColor(Color.GREEN);

canvas.drawRect(topX, topY, botX, botY, rectPaint);


     }
}
```

Because drawRect needs the coordinates of the upper left and the bottom right corners, we have to generate them.

Drawing on our screen

```
Rectangle myRectangle = new
   Rectangle(this, 20,40);

setContentView(myRectangle);
```

To think about:
Fair warning, your homework will be to do everything we did today for a circle

What Styles can we add?

One Color---

setColor(int color);

A two-color, linear gradient---

setShader(new LinearGradient(int xStart, int yStart, int xEnd, int yEnd,
    int color1, int color2, Shader.TileMode.MIRROR);

A two-color, linear gradient---

setShader(new RadialGradient(int xCenter, int yCenter, int radius, int
    color1, int color2, Shader.TileMode.MIRROR);

Dressing up our rectangle

- Add 2 new parameters to Rectangle method
  - color1 and color2

Don't forget to add colors when we call our rectangle in the main method!

```
rectPaint.setColor(color1);
```

OR

```
rectPaint.setShader(new LinearGradient
   (topX,botY,topX,botY, color1, color2,
   Shader.TileMode.MIRROR));
```

OR

```
rectPaint.setShader(new RadialGradient(centerX,
   centerY, 7, color1, color2,
   Shader.TileMode.MIRROR));
```

Drawables

- More static objects
- Can be placed on Canvas or ImageView
- More easily created, less easily manipulated
  - Angry Birds vs. background images
- Allows for some animation
  - Not meant to be iterated constantly

Drawing with Drawables

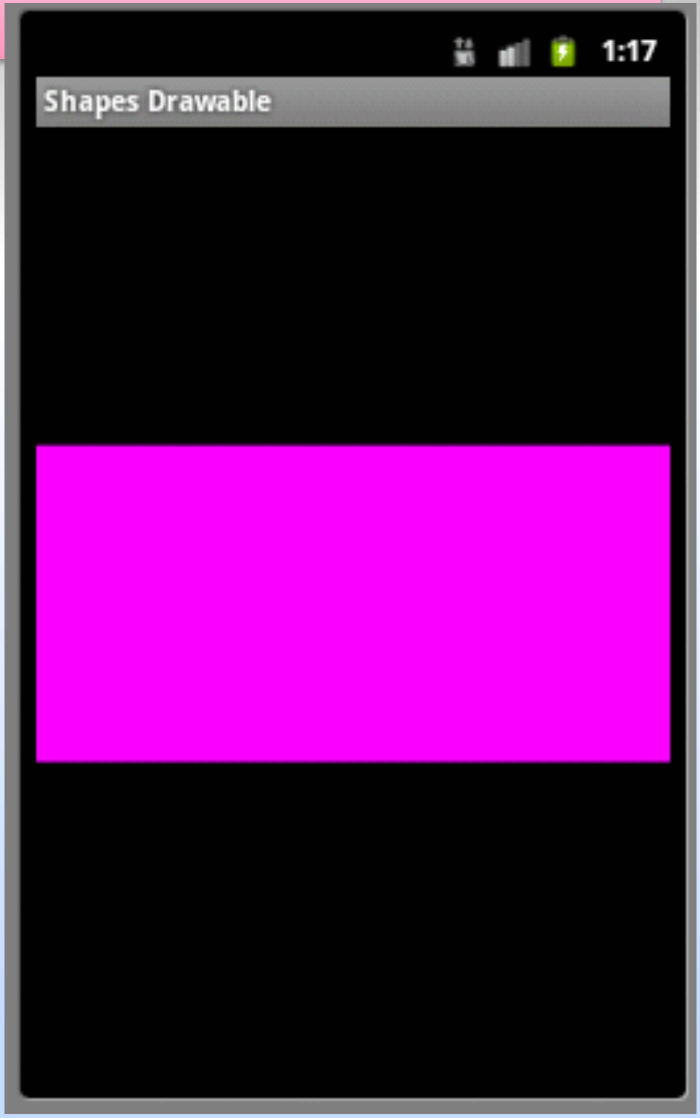
- Create new Android application, called ShapesCanvas
  - com.gdi.shapescanvas
  - Default activity ShapeCanvas
- Add an ImageView to main.xml

Creating a ShapeDrawable

```
ShapeDrawable rectangle = new
    ShapeDrawable ();

    rectangle.setShape(new RectShape());
    rectangle.setIntrinsicHeight(100);
    rectangle.setIntrinsicWidth(200);
    rectangle.getPaint().setColor
    (color.MAGENTA);

    ImageView iView = (ImageView)
    findViewById(R.id.imageView1);
     iView.setImageDrawable(rectangle);
```

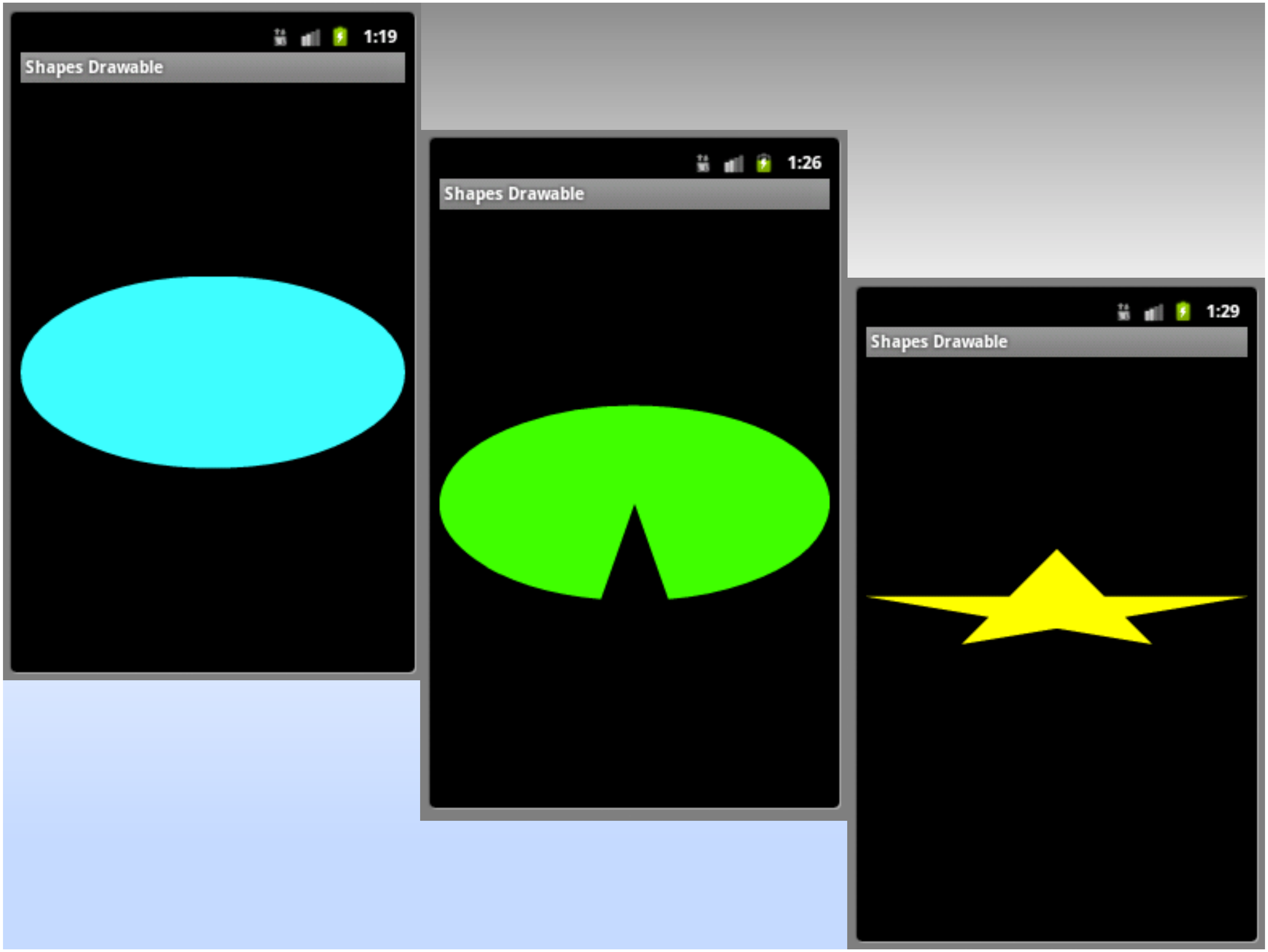


What else can we draw?

OvalShape()—oval width and height defined like RectShape

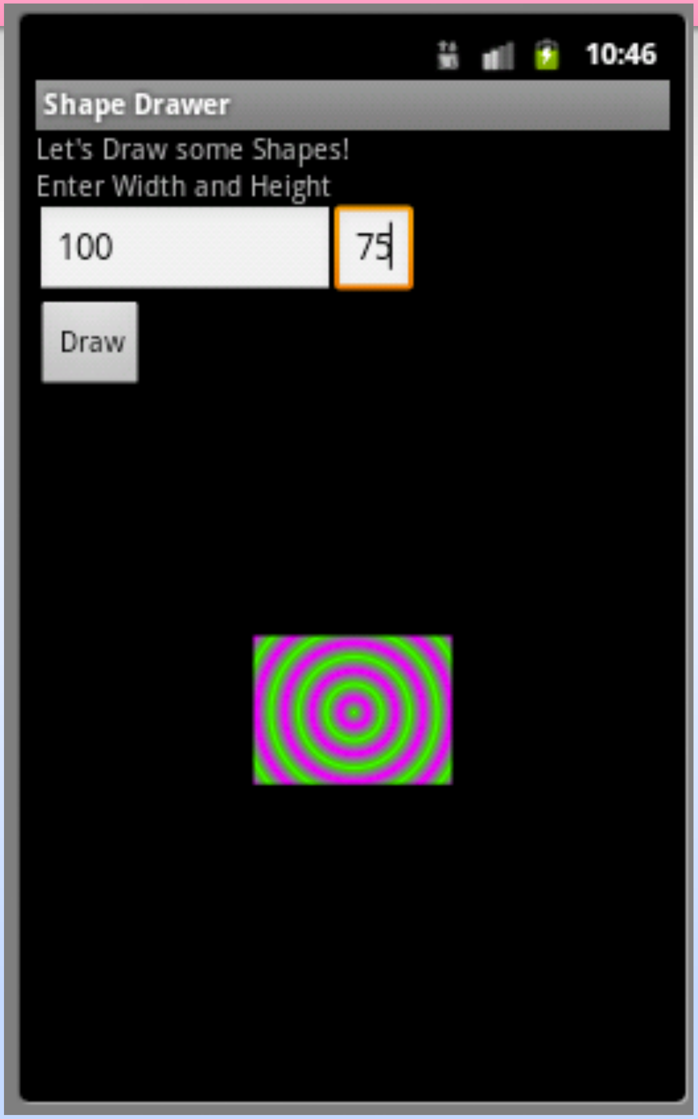
ArcShape(int startDegree, int endDegree)

```
Path p =new Path();
p.moveTo(50, 0);
p.lineTo(25, 100);
p.lineTo(100, 50);
p.lineTo(0, 50);
p.lineTo(75, 100);
p.lineTo(50, 0);
PathShape(Path path, int middleX, int middleY);
```


1:19
Shapes Drawable
1:26
Shapes Drawable
1:29
Shapes Drawable

Letting users pick the size

- Add two EditText widgets to the main.xml
- Add one button
- Create an onClick method that gets the height, width
- Create myRectangle with those values



Let's Draw some Shapes!
Enter Width and Height
100
75

Letting users pick the size

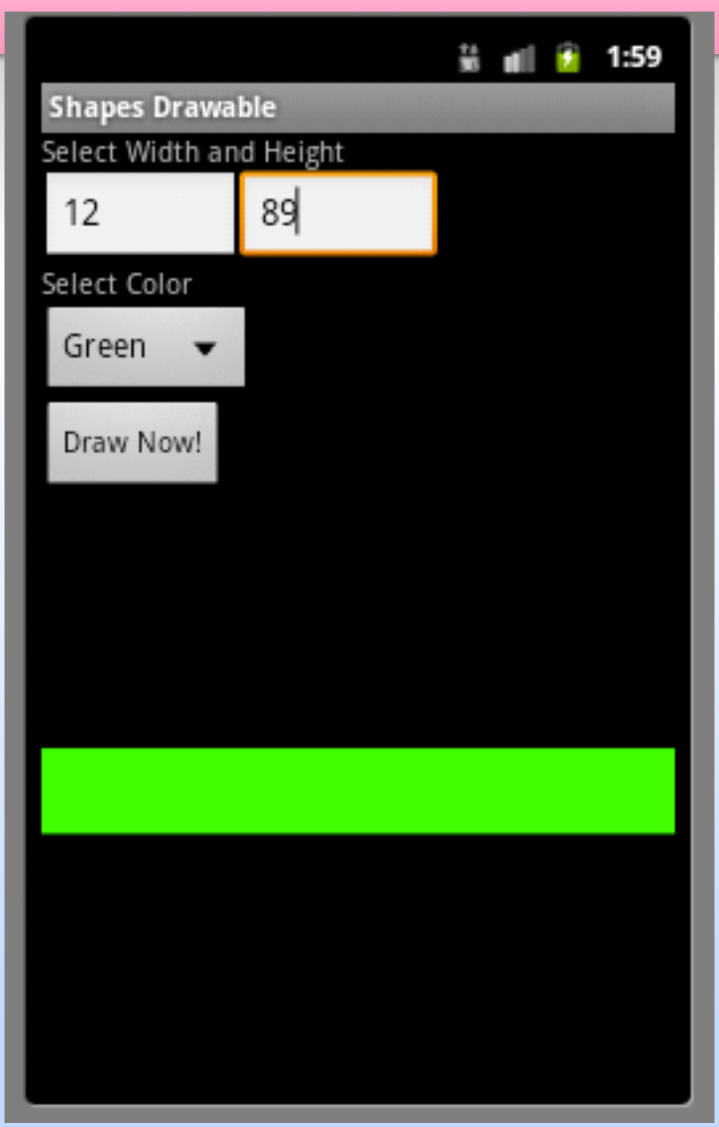
```
button = (Button) findViewById(R.id.button1);
widthText = (EditText) findViewById(R.id.editText1);
heightText = (EditText) findViewById(R.id.editText2);

button.setOnClickListener(new OnClickListener() {
        @Override
         public void onClick(View v) {
        String widthString=widthText.getText().toString();
        width=Integer.parseInt(widthString);
        String heightString=heightText.getText().toString();
        height=Integer.parseInt(heightString);
[...]
         rectangle.setIntrinsicHeight(width);
        rectangle.setIntrinsicWidth(height);

[...]
         }
 });
```

Letting Users Pick the Color

- Add a spinners in main.xml
- Populate them with an array of strings
- Get the user choice onClick
- Use the user's choice to add color



Letting Users Pick the Color

Spinner—

```
android:entries="@array/colors"
```

Strings—

```
<string-array name="colors">
    <item name="green">Green</item>
    <item name="blue">Blue</item>
    <item name="magenta">Magenta</item>
    <item name="cyan">Cyan</item>
</string-array>
```

Letting Users Pick the Color

```
colorSpinner= (Spinner)findViewById(R.id.spinner1);
int colorPos=colorSpinner.getSelectedItemPosition();
        if (colorPos==0){
              color=Color.GREEN;
       }
        else if(colorPos==1){
                  color=Color.BLUE;
        }
       else if(colorPos==2){
             color=Color.MAGENTA;
       }
       else{
             color=Color.CYAN;
       }
[...]
rectangle.getPaint().setColor(color);
[...]
```

In Class and Homework

- Create new applications (or add to your existing one) to create circles, ovals, or paths in Canvas and with ShapeDrawable
- Experiment with Paint Color and Gradient
- Think of one other widget we could use to get user data and apply it to the shapes we are drawing.

Questions?