A Block

Our Game So Far

- We have a ball that can move and bounce
- Our balls move at a constant speed, even if the frame-rate changes
- We can play sounds – bounce bounce bounce
- Images
- We can control things by dragging or tapping on them
- We have learned how one component can control other components and components can communicate with their controller
- And yet...

Our Block

- Another Flame game component class
- Rectangular
- It will have lives like our ball
- It will have a color
- It will display the number of lives it has left in its center

As an exercise, see how much of the Block component you can create in your own

- File will be block.dart, class name is Block
- Use Ball as a template
- Use a rectangle variable to hold the block's location on the screen
- Initialize the rectangle in the constructor

Here is our component so far.

```
import 'dart:ui';
import 'package:flutter/material.dart';
import 'package:flame/components/component.dart';
import 'package:flutter_ball/flutterball_game.dart';

class Block extends Component {

  // instance variables
  final FlutterballGame game;
  int lives;  // how many hits until the block dies
  Paint paint = Paint();  // paint the rectangle
  Rect position;  // position of block

  // create a block
  Block(this.game, {this.position, Color color=Colors.white, this.lives=10,}) : super() {
    paint.color = color;
    paint.style = PaintingStyle.fill;
  }

  void resize(Size size) { }

  void render(Canvas c) {
    // draw the block
    c.drawRect(position, paint);
  }

  void update(double t) { }

  bool destroy() => lives <= 0;

}
```

A Basic Component Revisited

- We have a class named Block that is based off of a regular Flame Component class
- We are not using images, so we don't need SpriteComponent
- Our game instance variable lets us access instance variables in our flutterball game if we need to. We always define it and set it in the constructor just in case we need it later on
- We are using a Rect class to hold the location of the block. A rect consists of 4 x,y points (Offsets)
- A Paint class will hold the color and fill style of our block
- We set our instance variables directly in our constructor – this.position
- We use named parameters to make the constructor easier to use { }
- We set our color and style to our paint object

No resize() ?

We don't need to know our screen size, we have the position we draw on

No update ?

The block stays still (for now). We don't need to calculate anything new on every frame

A canvas has a handy method for drawing rectangles

We pass it our block's position and paint

Add some blocks to the game and try it out. You can add as many as you want

```
// make a new block game component
var block = Block(this, position: Rect.fromLTWH(100, 200, 50, 50));

// tell the game about this component
add(block);
```

Before We Go On...

We will put text on our block to show how many lives it has

Before we do that, we will learn more about painting text on a canvas

Text is used all over games

- Title screen
- Help text
- Displaying current score and other information
- Various messages between rounds of the game

Painting text is more complex than painting basic shapes

We will make our own text box component that we will use use throughout our game