Text is Boring

Every Game Needs Text

- Game Engine Not Optimized for Creating Text
- Calculate exactly where you want the text
- Describe the text with various objects and methods
- “Compile” the text so it can be painted very quickly
- Paint the compiled text somewhere on the canvas during update

- Changing text means stop updating with one piece of text and start updating with another
- We will write a useful game component that lets you add text boxes to the game
- Erase them by setting their lives to 0, just like our ball

Steps to Creating Text

- Define a rectangle where you want text to go

  You really only need the top, left, and right. Text grows down
- Create a TextStyle object where you define basic information about the text
  - Color
  - Font size
  - Other font information
- Create a TextSpan object where you specify the text string and style object
- Create a TextPainter, passing it the text span and other formatting options
- Compile the text by calling layout() on the text painter
- Use the paint() method on the text painter to paint the text somewhere on the canvas

Our Text Component

To make things easy for a game designer to add text, our component will do most of the work. To add text to a game,

- You still need to create a text span object

  A simple one with white text at the default size

  ```
  TextSpan span = TextSpan(text: 'Hello world!');
  ```

- Create a TextDraw component, passing it the Rect object for a position and the span.
- Add it to the game
- You can add an optional border around the text box and color the box

Complex Text

A TextSpan object can define a complex piece of text

- Text is like a paragraph – it flows as a single line and wraps if necessary
- Define multiple pieces of text
- Allows you to bold, underline, change parts of the text
- Example

  Can you <u>find the</u> secret?

```
TextSpan(text: 'Can you ', style: TextStyle(color: Colors.yellow),
  children: <InlineSpan>[
    TextSpan(text: 'find the', style: TextStyle(
        color: Colors.green, decoration: TextDecoration.underline,
      ),
    ),
    TextSpan(text: ' secret?'),
  ],
)
```

More Information

- Google flutter textspan
- Look at the flutter painting library

  all the things you can do with a canvas
- Try out examples you find online
- Keep text simple
- If you want more complex text, you want flutter widgets, not text spans and a canvas

Our TextDraw Component

Time for a new component

You know the drill

File text.dart in components folder

Similar to the Block component

We do not need the game

These imports

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

Make the basic blank component, and we will then discuss all of the code

Instance Variables

These first ones should be self explanatory

```
// instance variables
int lives = 1;   // used to destroy the component
Paint paint;   // paint the rectangle
Paint border;   // paint the border
Rect position;   // position of block
```

They allow us to paint in the text box if desired, paint a border, position the box, and destroy the component by setting lives to 0 to make the text disappear.

Let's do a simple method while we're doing the easy stuff

```
bool destroy() => lives <= 0;
```

More Easy Stuff

We don't need to know the size of the screen, and we don't need to update anything between frames.

```
void resize(Size size) {
}
void update(double t) {
}
```

We know we will need a text span and text painter, so we'll add them to our instance variables.

```
// show text inside box
TextSpan textSpan;  // used to paint text
TextPainter tp = TextPainter(
  textDirection: TextDirection.ltr,
);
```

For some odd reason, a text painter doesn't default to a text direction, so we tell it one, left to right.

The Constructor

We must pass in 2 required parameters when creating our TextDraw component

- A rectangle position
- A text span object

Everything else is optional

- Color of text box
- Color of border,
- Thickness of border
- How text is aligned (centered, left, etc) default is centered
- Overall text scale (default is 2) for text size

Let's define the parameters of the constructor.

```
TextDraw(this.position, this.textSpan, {
  Color boxColor, Color borderColor,
  textAlign:TextAlign.center,
  borderThickness:1.0, scale:2.0,
  }) : super() {

}
```

The two required parameters can be set right to the instance variables (using this)

The other parameters get put inside other instance variable objects

```
tp.textAlign = textAlign;
tp.textScaleFactor = scale;
tp.text = textSpan;   // text to draw
```

We have our text painter all set up now, so we compile the text, giving a minimum and maximum width of the text. We just specify the entire width of our component.

```
tp.layout(minWidth: position.width, maxWidth: position.width, );
```

Remember that we defined our position rectangle in the constructor.

We need to specify a height, because it's a rectangle.

Our text could go past the height we specify, so we are really specifying a minimum height.

We adjust the height if our text is too tall

Once our text painter is compiled, we can get how high the text will be in pixels. We use this to bump up the height of the rectangle if necessary.

```
// see if box is too short
if (tp.height > position.height) {
  position = Rect.fromLTRB(position.left, position.top, position.right, position.top + tp.height);
}
```

If the text is taller than the box's height, we make a new rectangle

Keep all the sides from the old rectangle except the height

One last thing our constructor has to do

Set up paints for the box color and the border color

Only if they are specified

If not specified, we don't want to paint anything for them

```
if (boxColor != null) {
  paint = Paint();
  paint.color = boxColor;
  paint.style = PaintingStyle.fill;
}
if (borderColor != null) {
  border = Paint();
  border.color = borderColor;
  border.style = PaintingStyle.stroke;
  border.strokeWidth = borderThickness;
}
```

If we don't set our instance variable paints, they will have the value null and we won't use them to paint.

Last Thing – Paint the Text

Just like our other components, we put something on the screen by adding it to the render() method.

```
void render(Canvas c) {
  // draw the box, the border, and then the text on top
  if (paint != null) c.drawRect(position, paint);
  if (border != null) c.drawRect(position, border);
  tp.paint(c, Offset(position.left, position.top));
}
```

If there's a box paint defined, we draw it. It will be a filled-in color

If there's a border paint defined, we draw it. It will be an outline of the rectangle

We paint our text. We give paint() the canvas, and the point at which we start the text

Left hand top of our box

Try it Out

Let's add some blocks to the game. Here is some sample code, but you can experiment with the component and try your hand at making more complicated text.

```
Rect pos = Rect.fromLTRB(100, 100, 200, 200);
TextStyle style = TextStyle(color: Colors.black, fontSize: 10, );
TextSpan span = TextSpan(text: "By default, text is centered.  Notice how the box
grows to hold the text.", style: style);
var text = TextDraw(pos, span, borderColor: Colors.green, boxColor:
Colors.lightBlueAccent);
add(text);
```

Let's add some blocks to the game. Here is some sample code.

```
Rect pos = Rect.fromLTRB(100, 100, 200, 200);
TextStyle style = TextStyle(color: Colors.black, fontSize: 10, );
TextSpan span = TextSpan(text: "By default, text is centered.  Notice how the box
grows to hold the text.", style: style);
var text = TextDraw(pos, span, borderColor: Colors.green, boxColor:
Colors.lightBlueAccent);
add(text);
```

Here is some more complex text.

```
// add complex text to game
pos = Rect.fromLTRB(  0, 500, 500, 600);
span = TextSpan(
  text: 'Can you ',
  style: TextStyle(color: Colors.yellow),
  children: <InlineSpan>[
    TextSpan(
      text: 'find the',
      style: TextStyle(
        color: Colors.green,
        decoration: TextDecoration.underline,
        decorationStyle: TextDecorationStyle.wavy,
      ),
    ),
    TextSpan(
      text: ' secret?\n',
    ),
  ],
);
text = TextDraw(pos, span, textAlign: TextAlign.left);
add(text);
```

Next Time

The code for this section is here

https://github.com/shawnlg/flutter_ball/tree/131_text

Try making other text boxes. Try some complicated text to better understand text spans.

Next time we add more functionality to our block component.