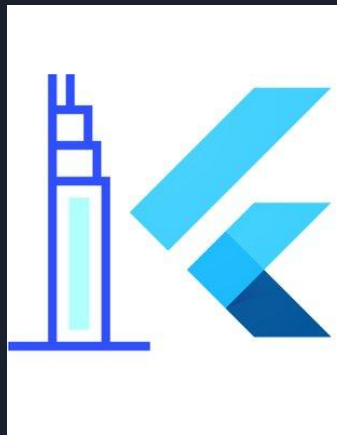


# First off, Big Thanks!

Jorge & the Chicago Flutter Group!

Everyone for tuning in!



# My Super Fast Intro



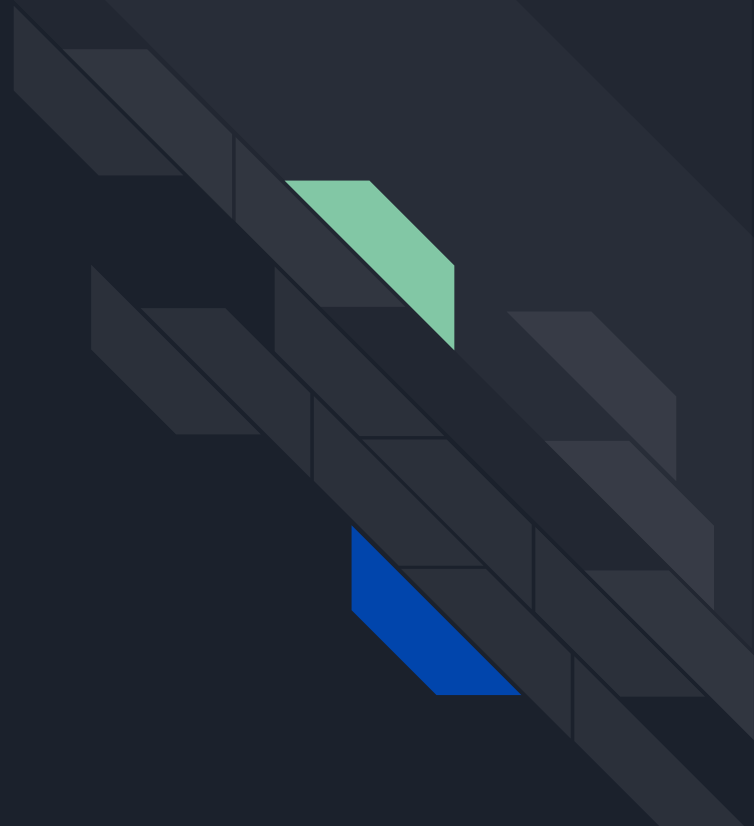
@she\_who\_codes  
Mandy Lowry

# Join the Flutteristas

(for women & non-binary)



@she\_who\_codes



@SHE.WHO.CODES



Extending the

# FlareController

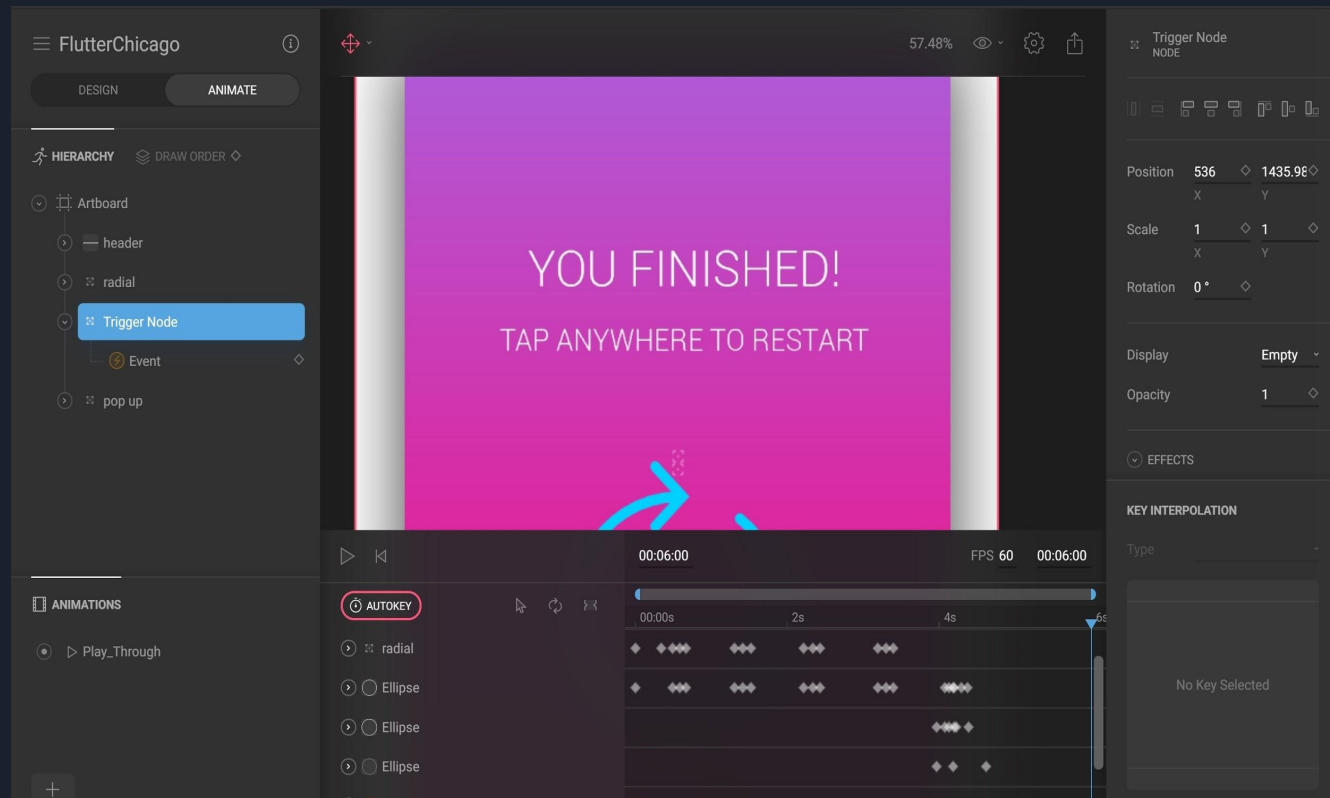
Mandy Lowry

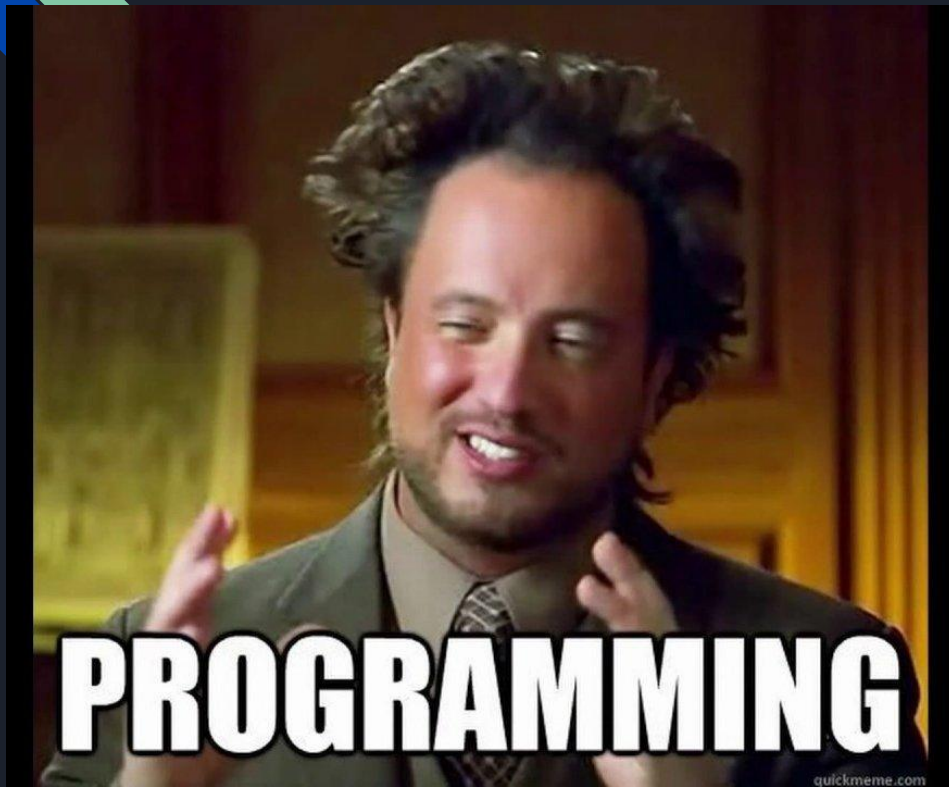
Our Progress Radial:



Art:

- Artboard
- Nodes
- Events
- Animations





Code:

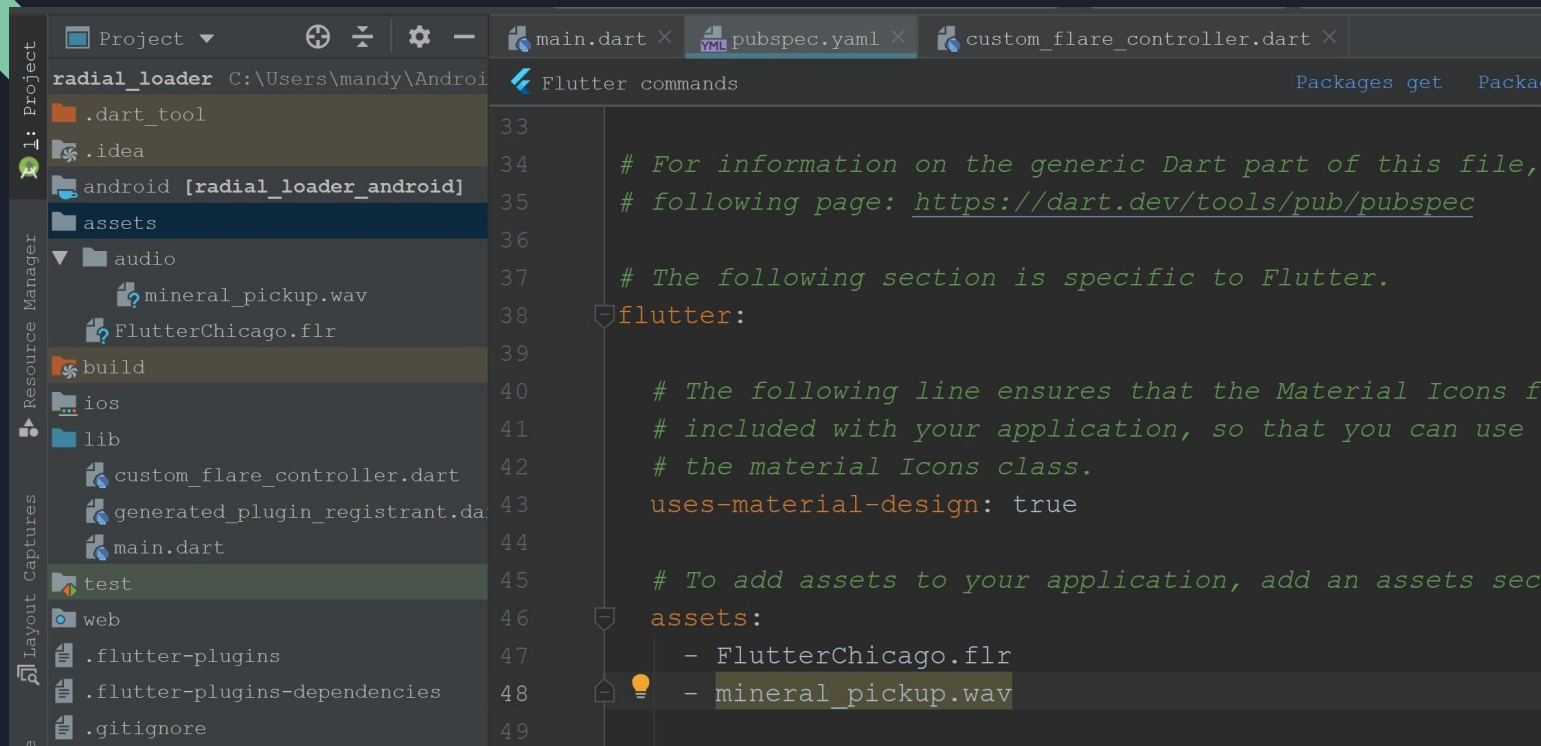
- Widgets
  - Stack, Container, FlareActor, Button
- FlareController
  - Artboard, TriggerEvents, ActorAnimation
- AudioPlayers
  - Play sound on event

# Grab our Plug Ins!

```
13 # https://developer.apple.com/library/archive/docu
14 version: 1.0.0+1
15
16 environment:
17   sdk: ">=2.1.0 <3.0.0"
18
19 dependencies:
20   flutter:
21     sdk: flutter
22   flare_flutter: ^2.0.3
23   audioplayers: ^0.15.1
24
25   # The following adds the Cupertino Icons font to
26   # Use with the CupertinoIcons class for iOS styl
27   cupertino_icons: ^0.1.2
28
```




# Add our assets!



# Clean Up Boiler Plate & Add Headers

```
main.dart x pubspec.yaml x custom_flare_controller.dart x
1 import 'package:flutter/material.dart';
2 import 'package:flare_flutter/flare_actor.dart';
3 import 'custom_flare_controller.dart';
4
5 void main() => runApp(MyApp());
6
7 class MyApp extends StatelessWidget {
8   // This widget is the root of your application.
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Chicago Flutter',
13      theme: ThemeData(
14        primarySwatch: Colors.pink,
15      ), // ThemeData
16      home: MyHomePage(title: 'Chicago Flutter'),
17    ); // MaterialApp
18  }
19 }
20
21 class MyHomePage extends StatefulWidget {
22   MyHomePage({Key key, this.title}) : super(key: key);
23
24   final String title;
25
26   @override
27   _MyHomePageState createState() => _MyHomePageState();
28 }
29
30 class _MyHomePageState extends State<MyHomePage> {
31
```

# Set Up Widgets!




```
main.dart x pubspec.yaml x custom_flare_controller.dart x
});
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ), // AppBar
    body: Center(
      child: Stack(
        children: <Widget>[
          ///Actor must have a sized parent
          Container(
            width: MediaQuery.of(context).size.width,
            height: MediaQuery.of(context).size.height,
            ///Here we set up our Actor with the .flr File
            child: FlareActor("assets/FlutterChicago.flr",
              fit: BoxFit.contain,
              animation: "Play_Through",
            ), // FlareActor, Container
          ], // <Widget>[]
        ), // Stack
      ), // Center
    floatingActionButton: FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: Icon(Icons.add),
    ), // FloatingActionButton
  ); // Scaffold
}
```

# Our Custom Controller!

```
1  import 'package:flare_dart/math/mat2d.dart';
2  import 'package:flare_flutter/flare.dart';
3  import 'package:flare_flutter/flare_controller.dart';
4  import 'package:audioplayers/audioplayers.dart';
5
6  ///this will be a custom controller for our Flare Actor
7  ///It extends the FlareController class, so there will be some required overrides
8  class CustomFlareController extends FlareController {
9
10     ///where we get a ref to our artboard and animation that we'll be translating
11     void initialize(FlutterActorArtboard artboard)
12     {
13
14     }
15
16     ///required since we extended FlareController
17     void setViewTransform(Mat2D viewTransform)
18     {
19
20     }
21
22     ///required since we extended FlareController, also where the magic happens!
23     bool advance(FlutterActorArtboard artboard, double elapsed)
24     {
25         return true;
26     }
27
28 }
29
```

# Add CustomFlareController to Main.dart



```
in.dart x pubspec.yaml x custom_flare_controller.dart x
}

class _MyHomePageState extends State<MyHomePage> {

  ///our class that extends the FlareController
  CustomFlareController controller = CustomFlareController()

  ///simple check to see if we have reached the max number c
  bool _isFull = false;

  void _incrementCounter()
  {
    controller.incrementFill();

    setState(() {
      ///we need to know if it's full so we can reset it
      _isFull = controller.isFull();
    });
  }
}
```

# Add Controller to Widget

```
main.dart x pubspec.yaml x custom_flare_controller.dart x
52 |   title: Text(widget.title),
53 | ), // AppBar
54 | - body: Center(
55 |   | child: Stack(
56 |     | children: <Widget>[
57 |       | ///Actor must have a sized parent
58 |       | - Container(
59 |         | width: MediaQuery.of(context).size.width,
60 |         | height: MediaQuery.of(context).size.height,
61 |         | ///Here we set up our Actor with the .flr File
62 |         | - child: FlareActor("assets/FlutterChicago.flr",
63 |           | fit: BoxFit.contain,
64 |           | /// intro animation
65 |           | animation: "Intro",
66 |           | ///set our custom class to be the controller for [FlareActor]
67 |           | controller: controller,
68 |         | ), // FlareActor
69 |       | ), // Container
70 |     | ], // <Widget>[]
71 |   | ), // Stack
72 | ), // Center
73 | - floatingActionButton: FloatingActionButton(
74 |   | onPressed: _incrementCounter,
75 |   | tooltip: 'Increment',
76 |   | - child: Icon(
77 |     | /// if we are done, then we need to restart everything, so change the icon
78 |     | isFull ? Icons.refresh : Icons.add,
79 |     | size: 40.0,
80 |   | ), // Icon
81 | ), // FloatingActionButton
82 | ); // Scaffold
```

# Add Vars to Controller

```
art x pubspec.yaml x custom_flare_controller.dart x
///this will be a custom controller for our Flare Actor
///It extends the FlareController class, so there will be some required overrides
class CustomFlareController extends FlareController {
  ///artboard
  FlutterActorArtboard _artboard;

  ///can the pop up animate in
  bool _animatePopUp = false;
  ///can the pop up be visible or should the other be visible?
  bool _showPopUp = false;


  ///how much we increment each tap
  double _incrementAmt;
  ///gets increased by increment amount
  double _myFill = 0;
  ///what we'll use to smooth the fill
  double _currentFill = 0;
  ///time used to smooth the fill line movement
  double _smoothTime = 5;
  ///time for playing the pop up animation
  double _animTime = 0;

  ///fill animation
  FlareAnimationLayer _baseAnimation;
  ///pop up animation
  ActorAnimation _popUpAnimation;
  ///for playing audio
  AudioCache audioCache = AudioCache();
}
```

# Initialize all the things!

```
39  ///where we get a ref to our artboard and animations that we'll be translating
40  void initialize(FlutterActorArtboard artboard)
41  {
42      _artboard = artboard;
43
44      _baseAnimation = FlareAnimationLayer()
45      ..animation = _artboard.getAnimation("Fill_Up");
46
47
48      _popUpAnimation = _artboard.getAnimation("End_Pop_Up");
49
50      ///math based on how many taps we want before radial is full.
51      _incrementAmt = _baseAnimation.animation.duration/29;
52
53  }
54
```






```
main.dart x pubspec.yaml x custom_flare_controller.dart x
112
113     ///called from tapping the button
114     void incrementFill()
115     {
116         if(isFull() == false){
117             _myFill += _incrementAmt;
118
119         } else {
120             _myFill = 0;
121             _showPopUp = false;
122
123         }
124     }
125
126     ///checks to see if our radial is full
127     ///also used in button tap to set icon image
128     bool isFull()
129     {
130         if(_myFill > 1){
131             return true;
132         }
133         return false;
134     }
135
136     ///called on animation event
137     void _playSound()
138     {
139         audioCache.play('audio/mineral_pickup.wav', mode: PlayerMode.LOW_LATENCY);
140     }
141 }
```

# Pop Up Code

```
60 void onCompleted(String name) {
61     if(name.compareTo("End_Pop_Up") == 0){
62         _animatePopUp = false;
63     }
64 }
65 ///required since we extended FlareController, also where the magic happens!
66 bool advance(FlutterActorArtboard artboard, double elapsed)
67 {
68     ///pop up animation
69     if(_showPopUp == true) {
70
71         if(_animatePopUp == true){
72             _animTime += elapsed * 1;
73         }
74         _popUpAnimation.apply(_animTime % _popUpAnimation.duration, _artboard, 1);
75
76         if ((_animTime % _popUpAnimation.duration + .04) >
77             _popUpAnimation.duration) {
78             onCompleted(_popUpAnimation.name);
79         }
80
81     } else ///filling animation
```

# Fill Animation



```
main.dart x pubspec.yaml x custom_flare_controller.dart x
81 } else {///filling animation
82
83     _currentFill += (_myFill-_currentFill) * min(1, elapsed *
84         _smoothTime);
85
86     _baseAnimation.animation.apply( _currentFill * _baseAnimation.animation.duration,
87         artboard, 1);
88
89     ///EventTrigger
90     List<AnimationEventArgs> _animationEvents = [];
91
92     double currLayerAnim = _baseAnimation.time;
93     _baseAnimation.time =
94         (_currentFill * _baseAnimation.animation.duration);
95
96     _baseAnimation.animation.triggerEvents(
97         artboard.components, currLayerAnim, _baseAnimation.time, _animationEvents);
98
99     for (var event in _animationEvents) {
100         switch (event.name) {
101             case "Event":
102                 _animatePopUp = true;
103                 _showPopUp = true;
104                 _playSound();
105                 break;
106         }
107     }
108 }
109 return true;
110 }
```

# Can't Touch This

```
37  
38 void _incrementCounter()  
39 {  
40     controller.incrementFill();  
41  
42     setState(() {  
43         //we need to know if it's full so we can reset  
44         _isFull = controller.isFull();  
45     });  
46 }  
47
```

**We did it!**  
**Thank you so much!**

