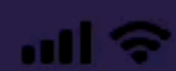




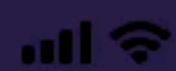
hello!



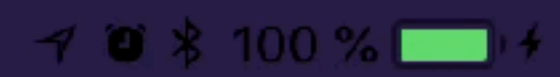
09:41

100 %  





09:41



# Building Structuring Designing

# Building Animations

# CALayer

```
layer.backgroundColor = UIColor.blue.cgColor
```

# Set the Actions Dictionary

```
layer.actions = ["backgroundColor": NSNull()]  
layer.backgroundColor = UIColor.blue.cgColor
```

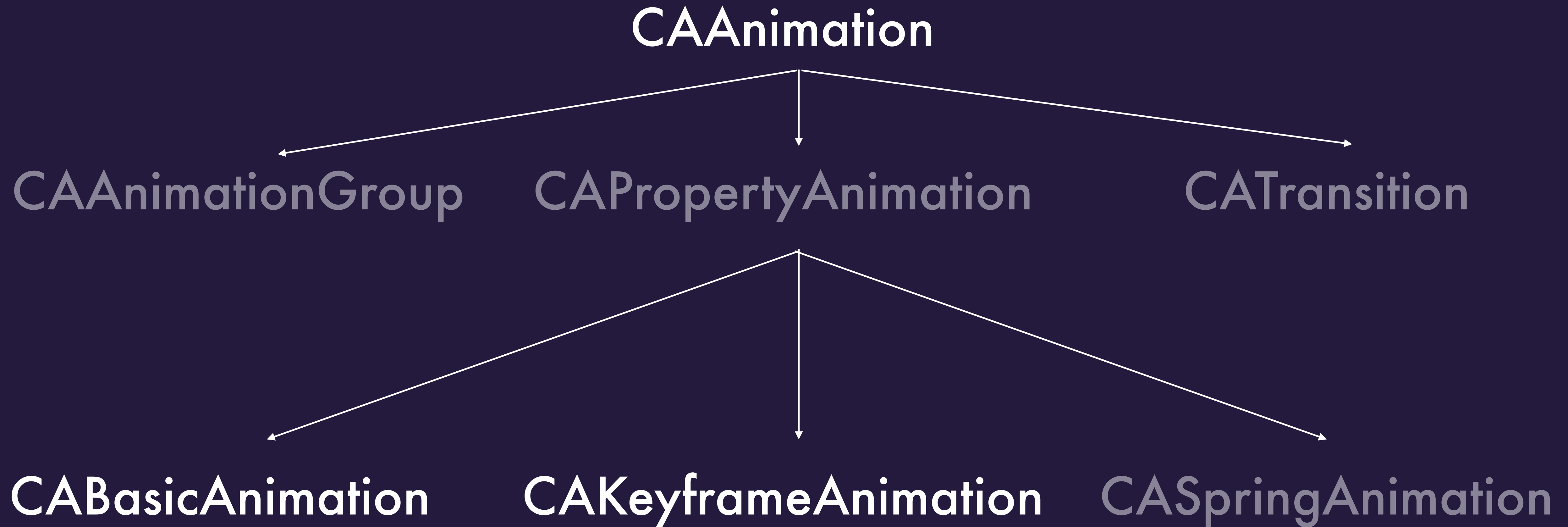
## Custom convenience method

```
layer.removeImplicitAnimations()
```

# Explicit Transactions

```
CATransaction.begin()  
CATransaction.setDisableActions(true) //Disable implicit animations  
//Code  
CATransaction.commit()
```





```
let ani = CABasicAnimation(keyPath: "")
```

## Key Path

transform

transform.translation

transform.translation.x

transform.translation.y

transform.translation.z

transform.scale

transform.scale.x

...

## Value

CATransform3D

[CGFloat, CGFloat]

CGFloat

CGFloat

CGFloat

[CGFloat]

CGFloat

...

# Key Path

# Value

bounds

CGRect

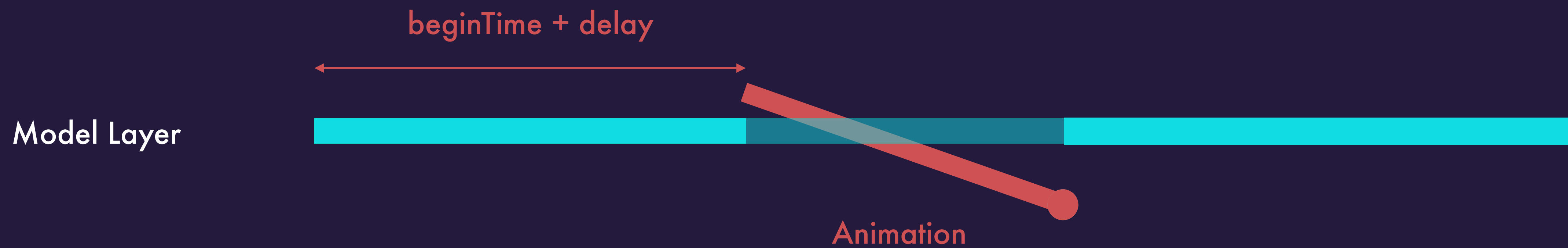
bounds.size.width

CGFloat

...

...

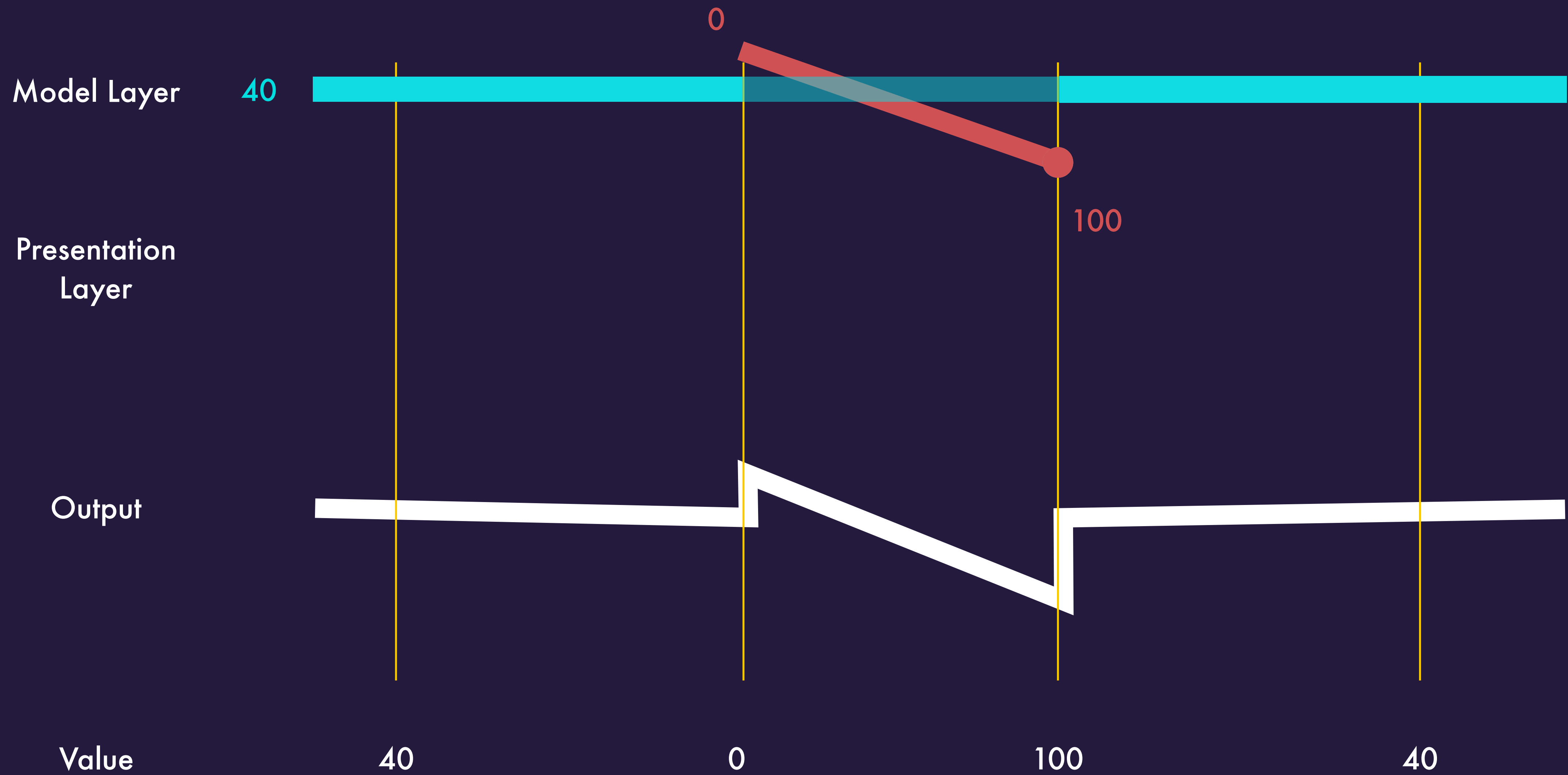
```
let ani = CABasicAnimation(keyPath: "transform.translation.y")
ani.duration = 1.0
ani.fromValue = 0
ani.toValue = 100
ani.beginTime = layer.convertTime(CACurrentMediaTime(), from: nil) + 2.0
layer.add(ani, forKey: nil)
```

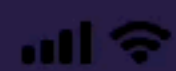


Presentation  
Layer


Output

Value





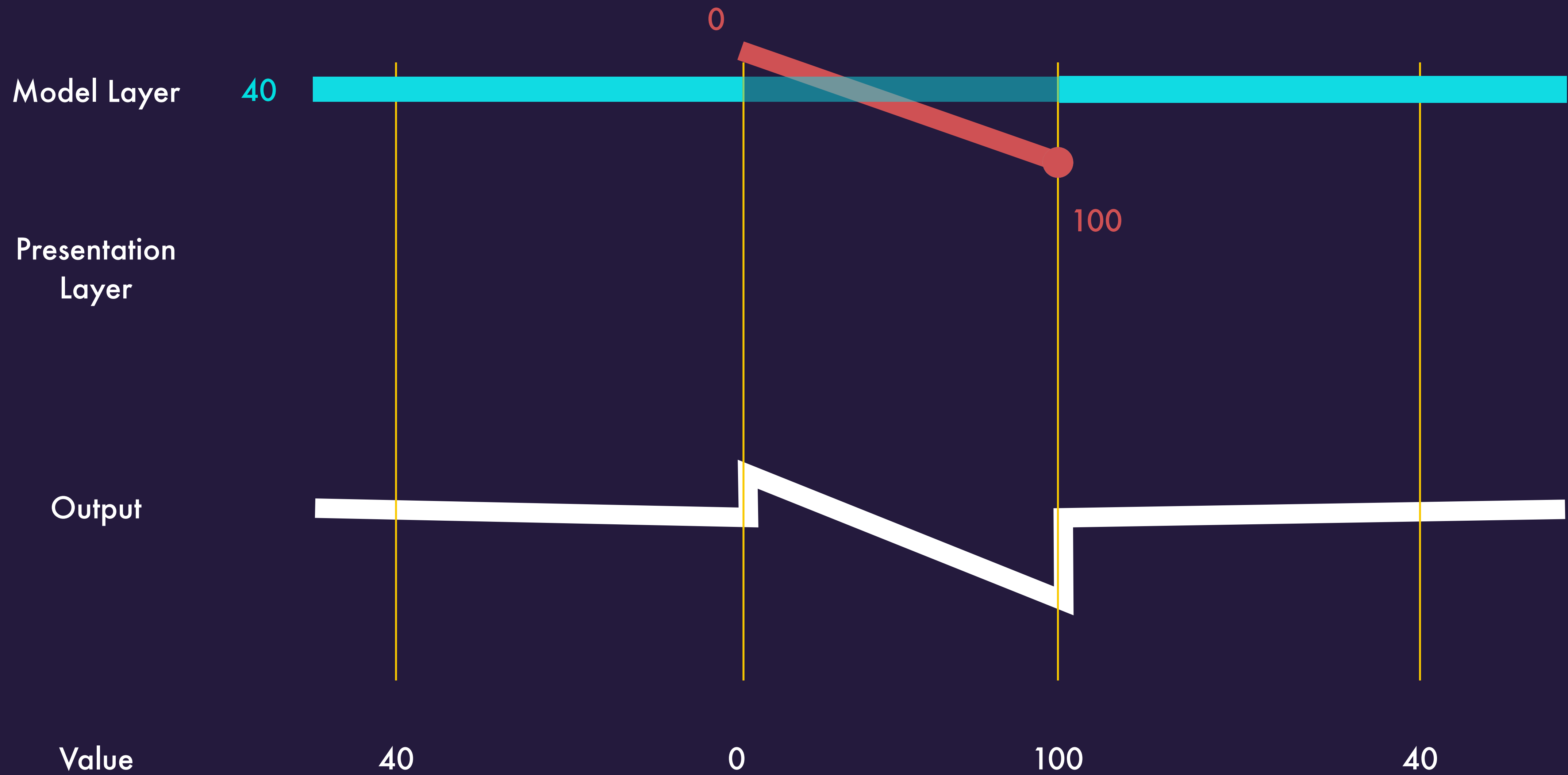
09:41

100 %  

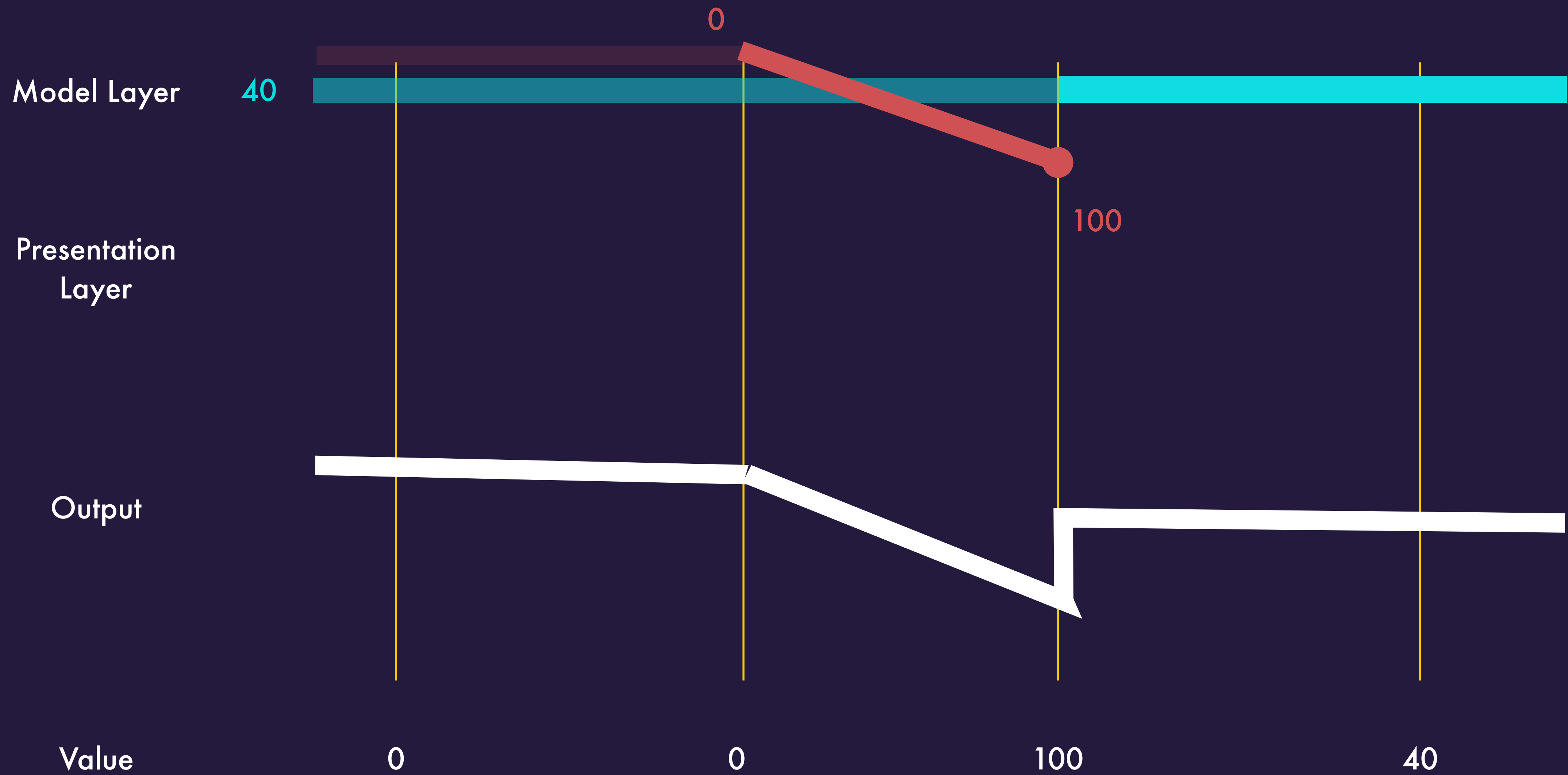




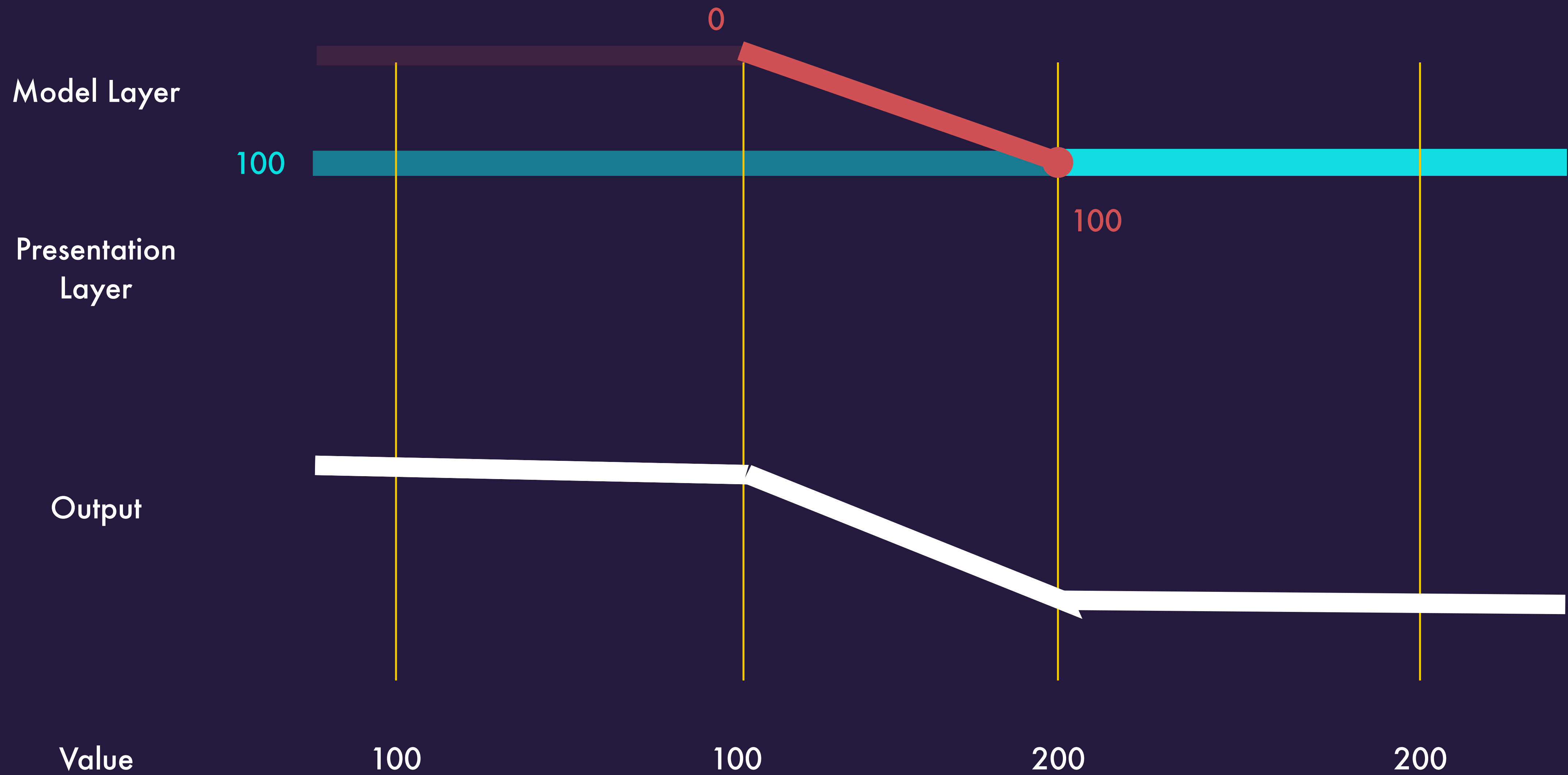
```
ani.fillMode = kCAFillModeRemoved
```

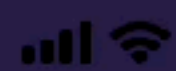


```
ani.fillMode = kCAFillModeBackwards
```



```
ani.fillMode = kCAFillModeBackwards
```





09:41

100 % The battery icon is a green rectangle with a white outline, indicating a full charge.



```
let ani = CABasicAnimation(keyPath: "transform.translation.y")
ani.duration = 1.0
ani.fromValue = 60.0
ani.toValue = 100.0
ani.fillMode = kCAFillModeBackwards
ani.beginTime = layer.convertTime(CACurrentMediaTime(), from: nil) + 2.0
layer.add(ani, forKey: nil)
layer.transform = CATransform3DMakeTranslation(0, 100.0, 0.0)
//l.setValue(ani.toValue, forKeyPath: "transform.translation.y")
```

# CABasicAnimation

fromValue, toValue

0

100



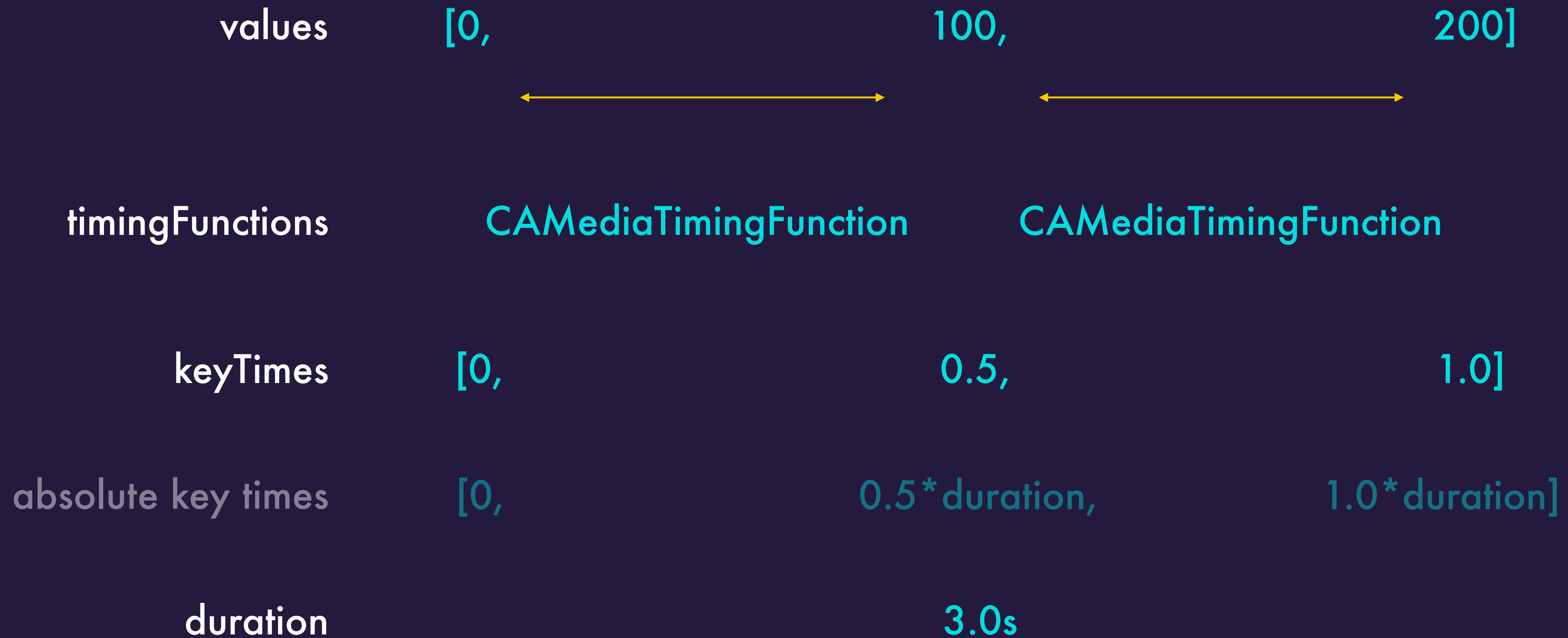
timingFunctions

CAMediaTimingFunction

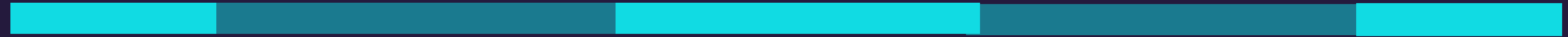
duration

3.0s

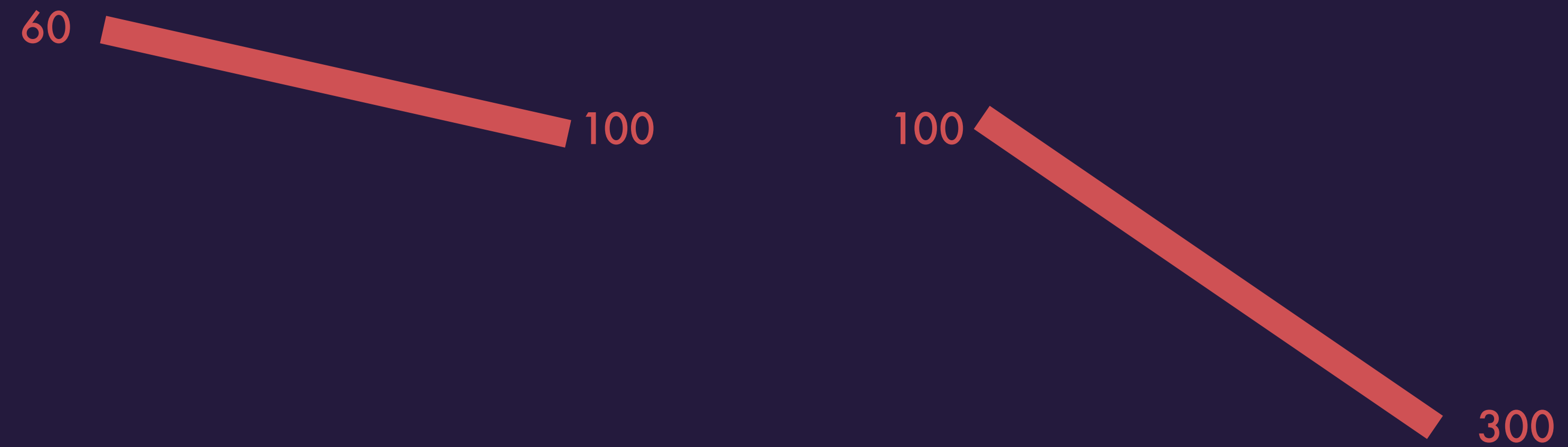
# CAKeyFrameAnimation



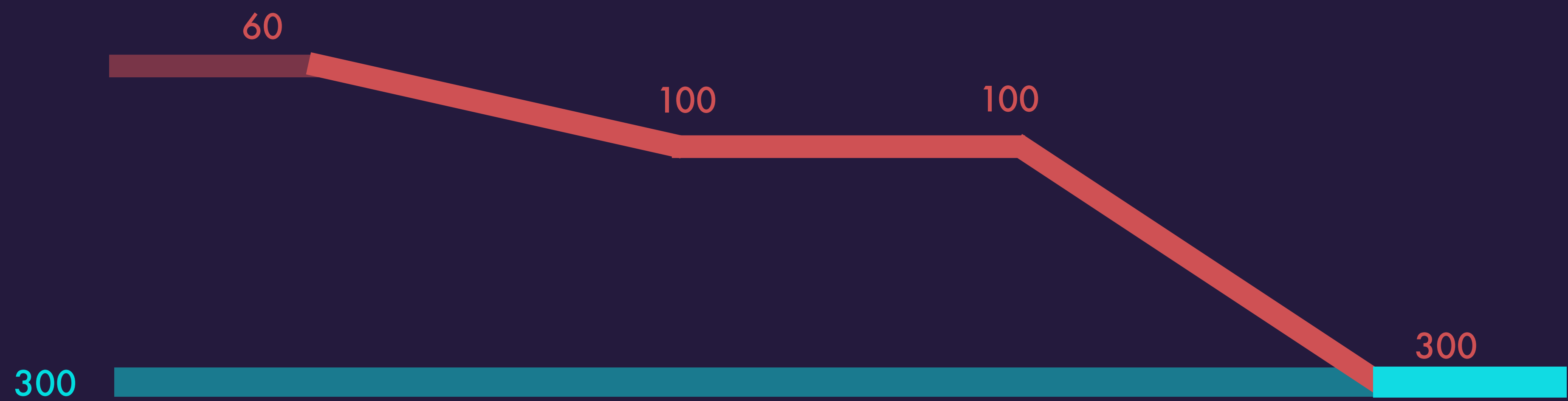
Model Layer



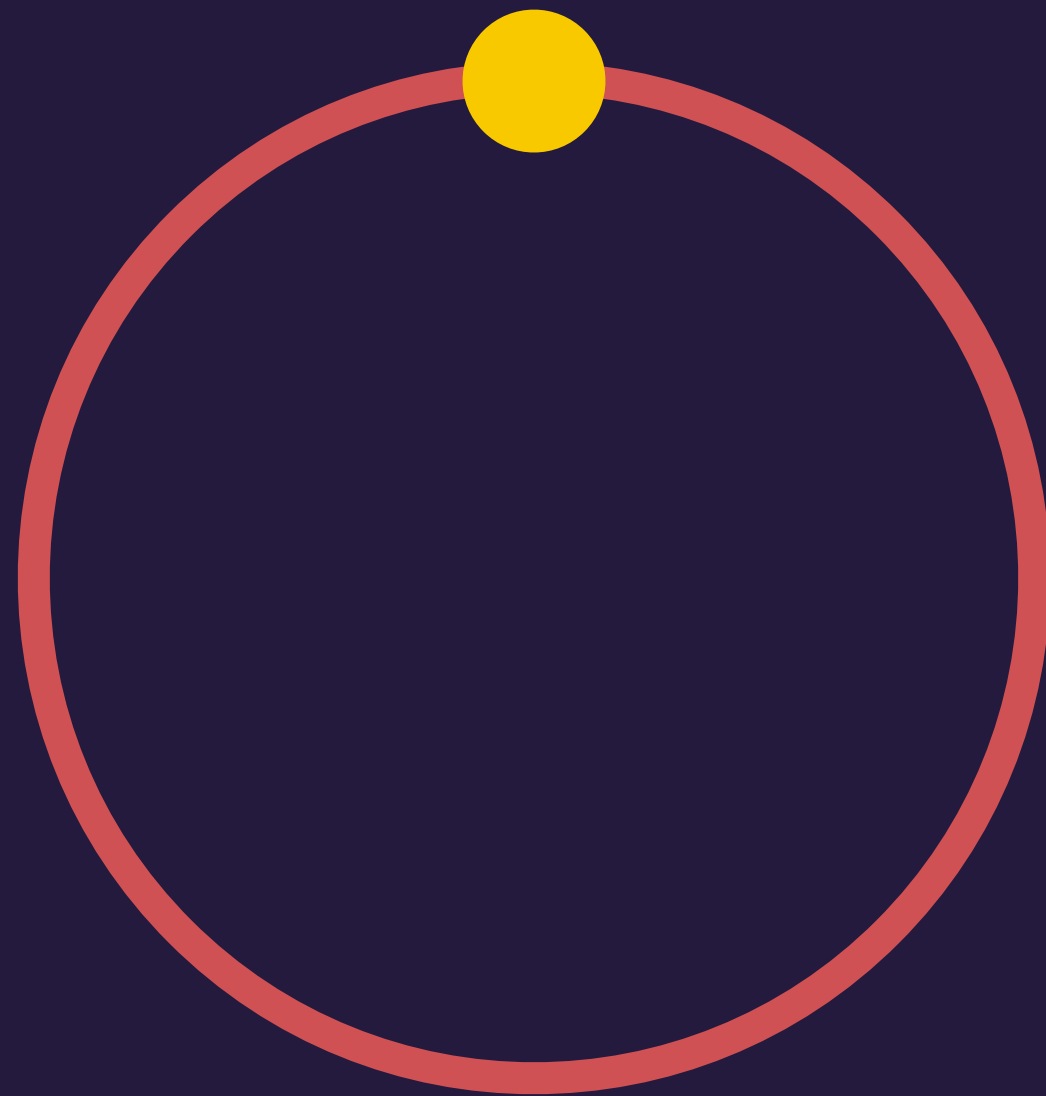
Presentation  
Layer

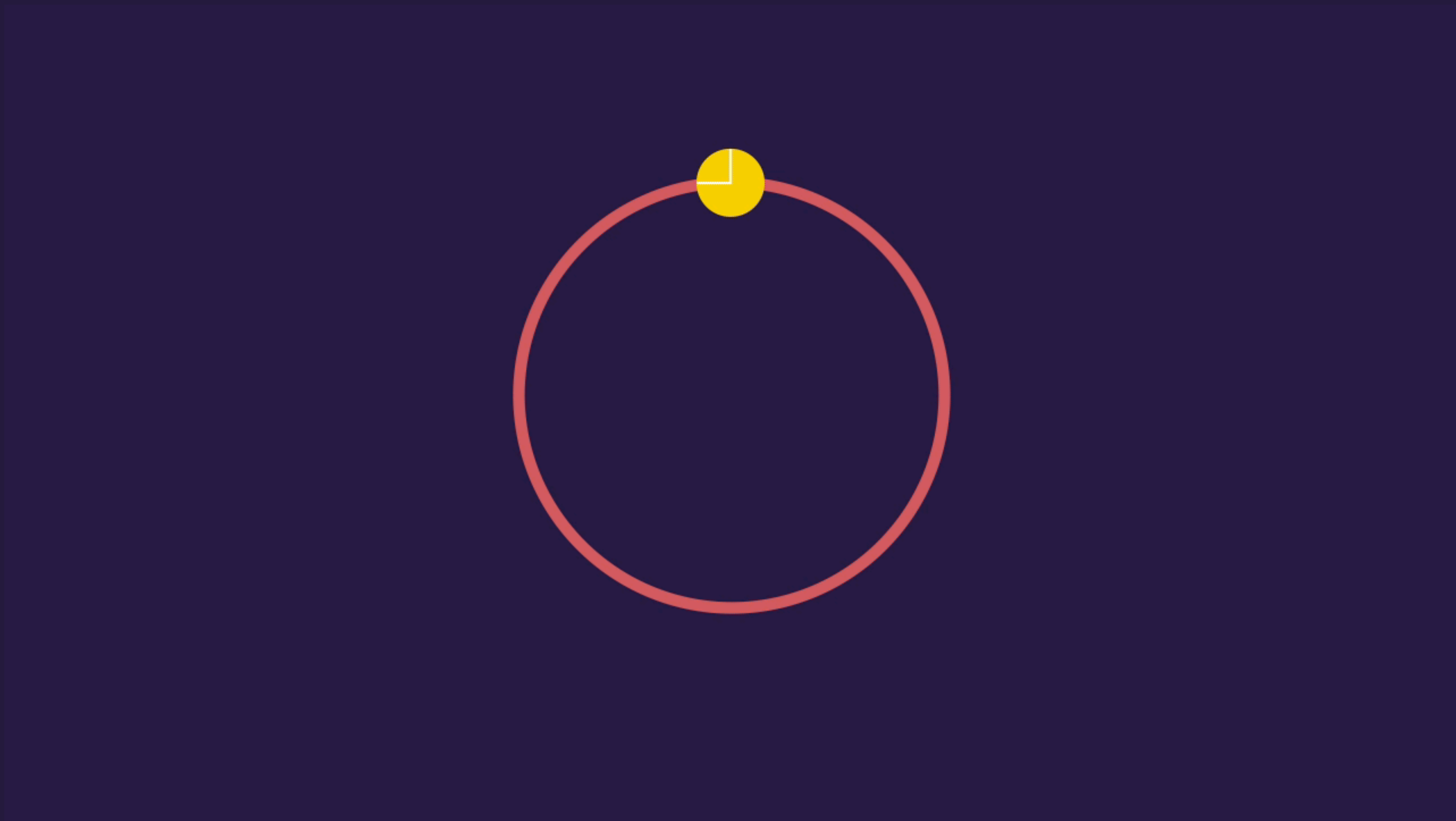


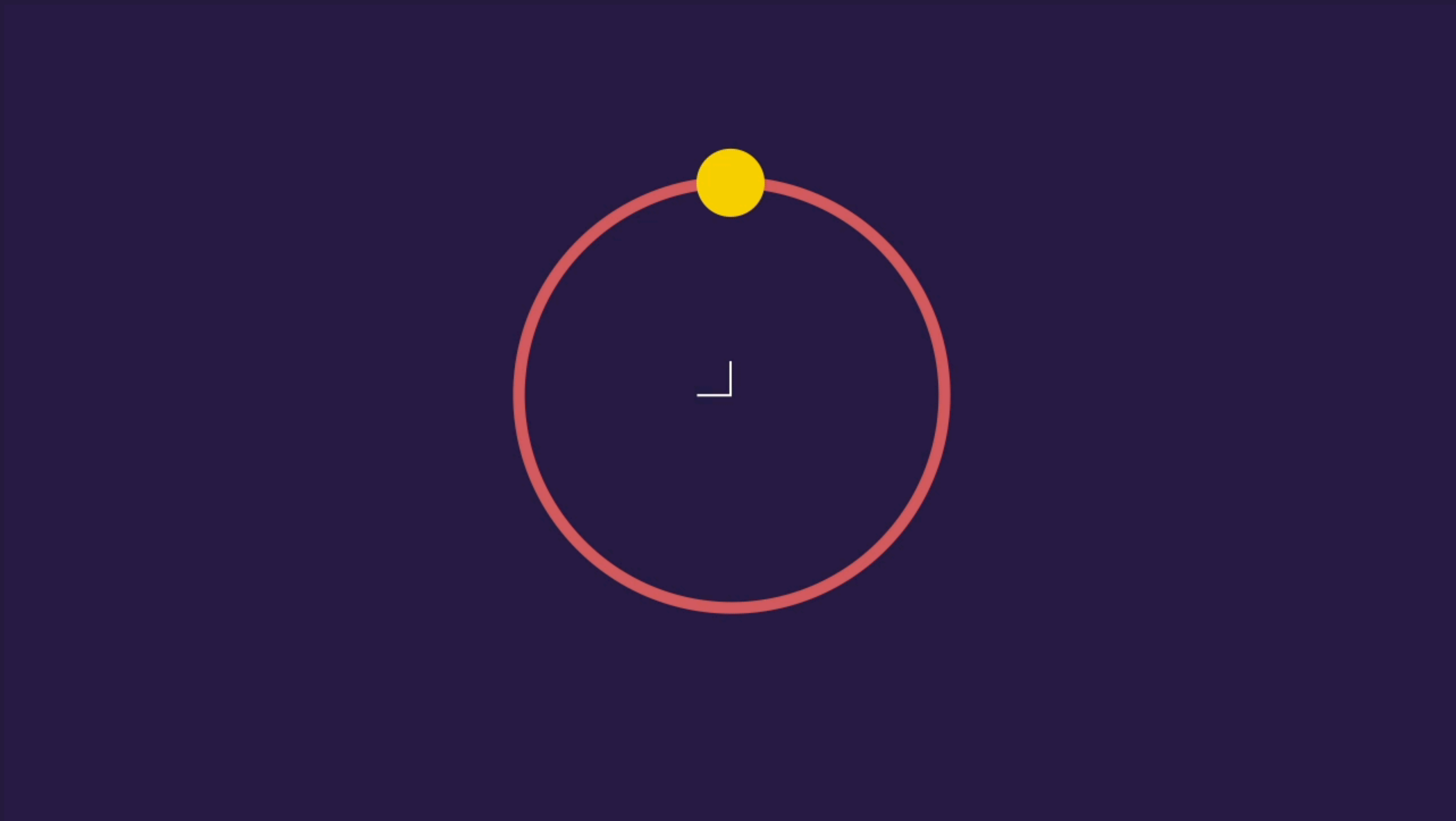


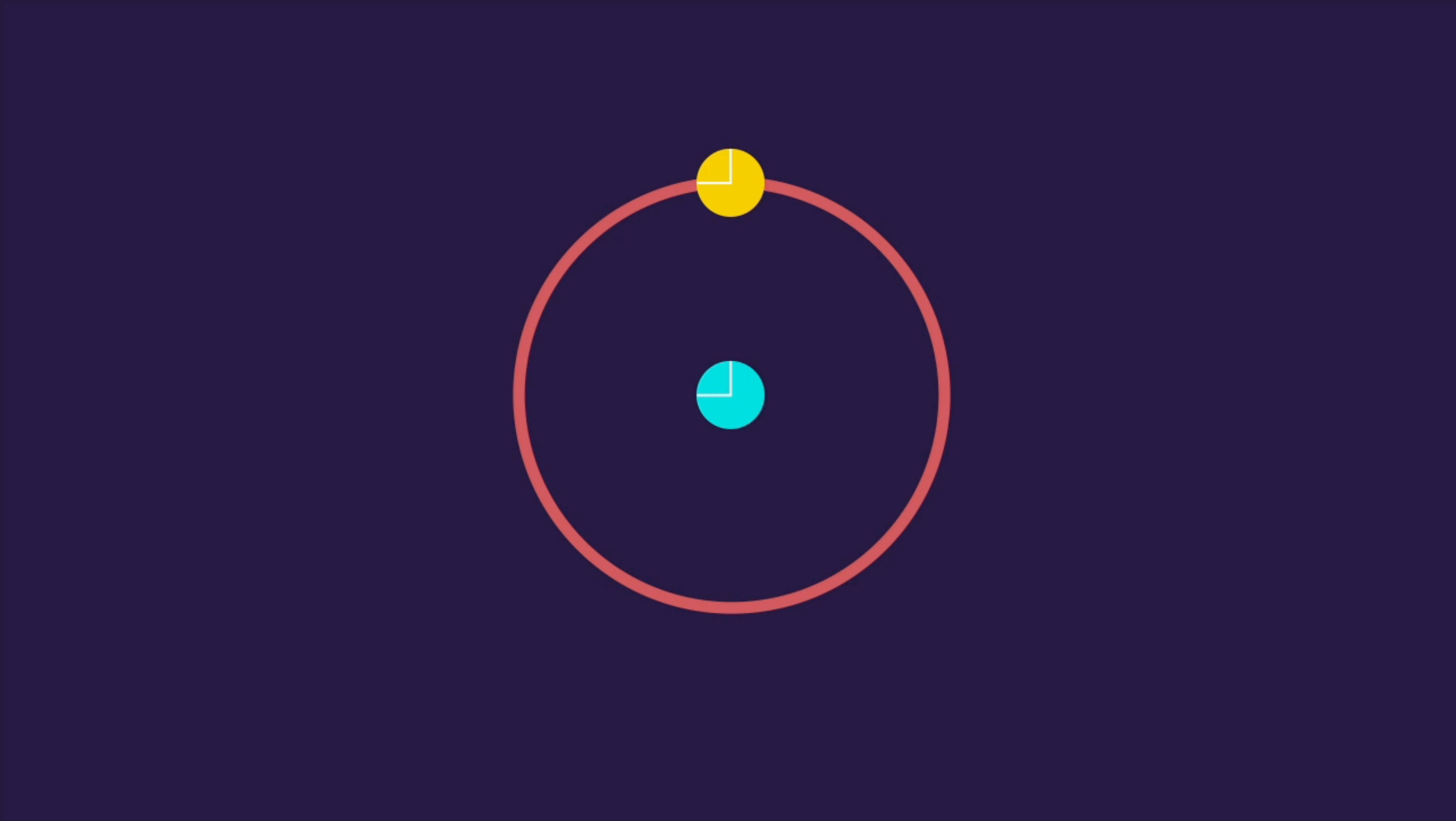


# Structuring Animations







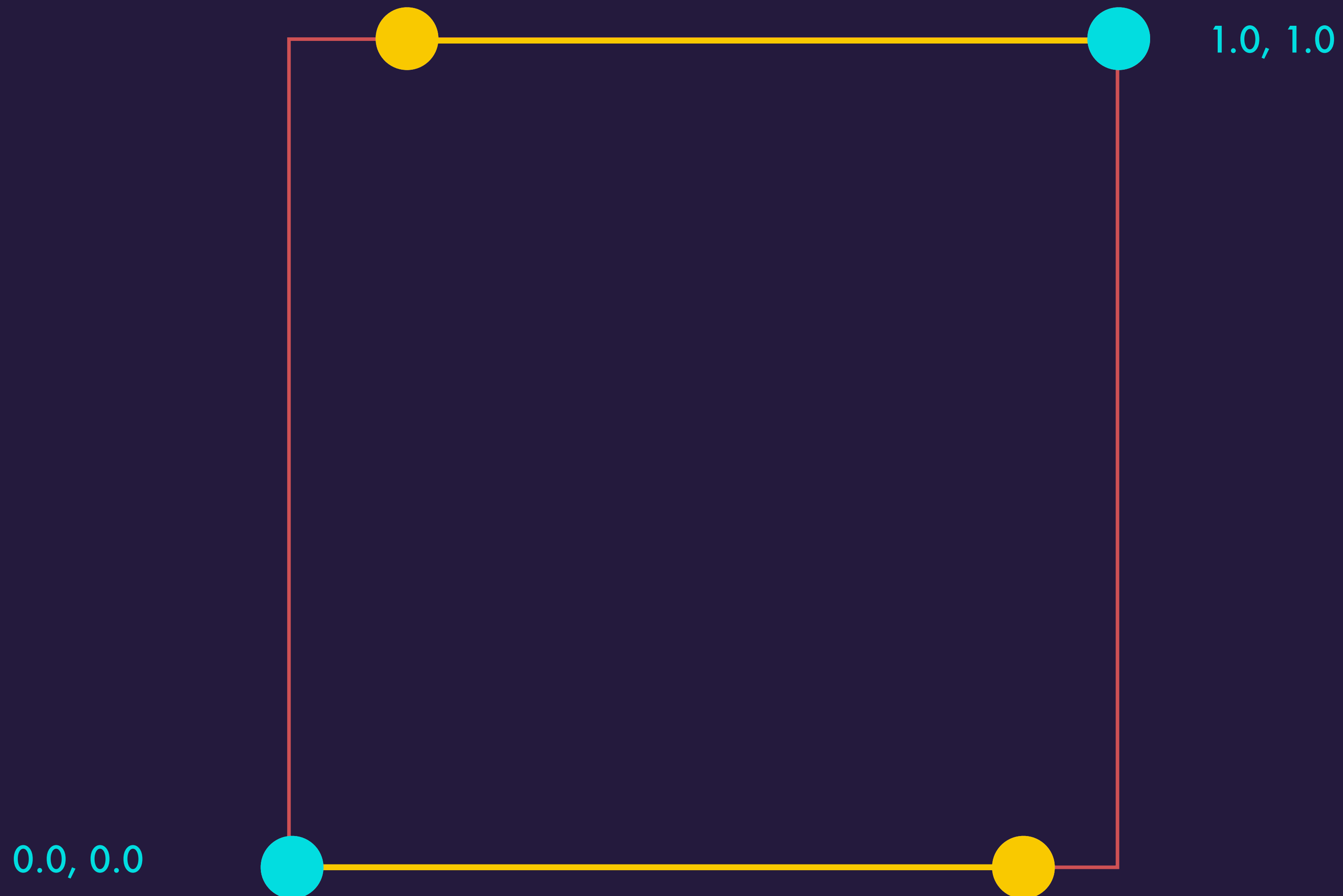


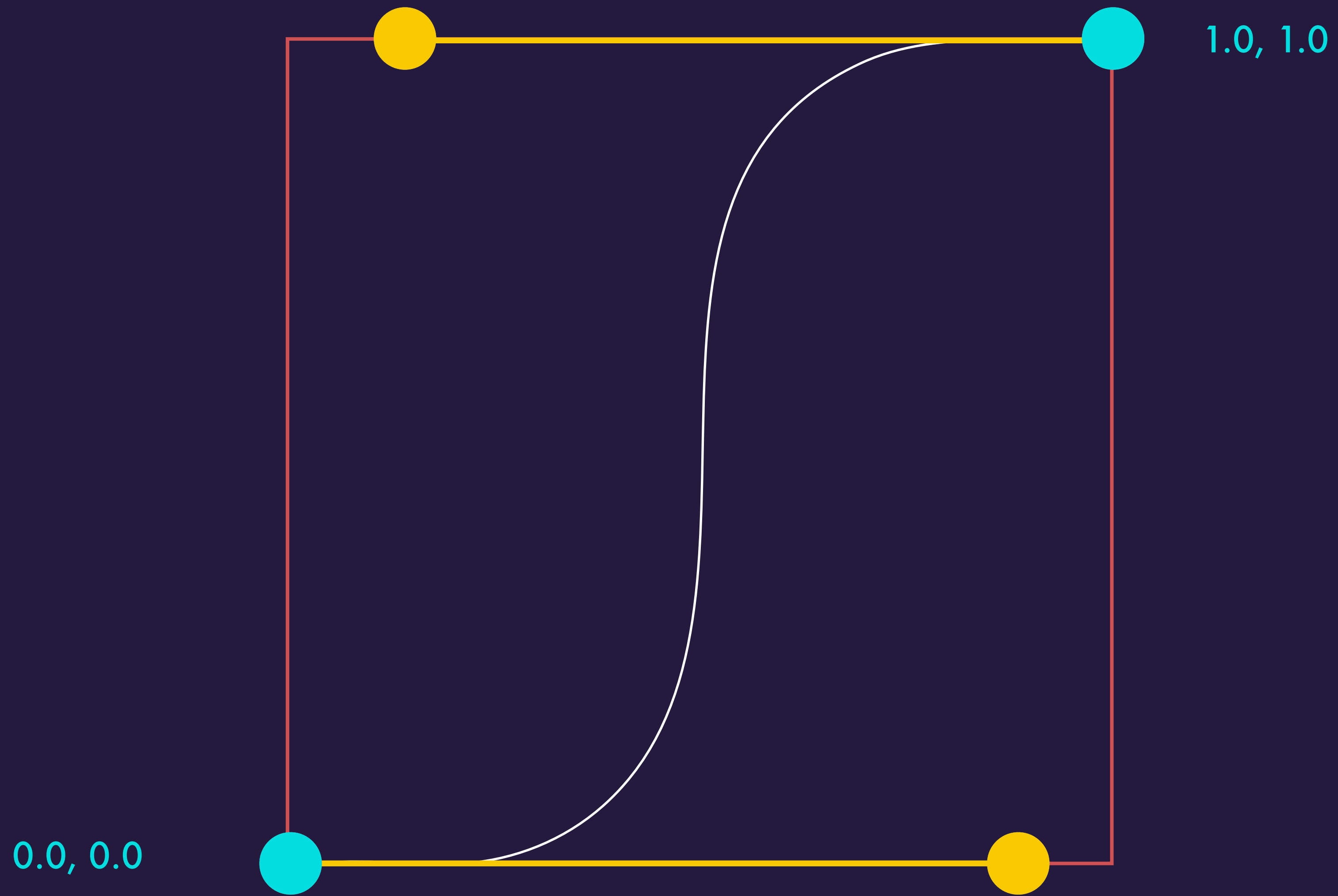
- Remove implicit animations from CALayers
- `fillMode = kCAFillModeBoth`
- prefer `CAKeyFrameAnimations` over chaining animations
  - avoids state checking
  - easy to remove
- prefer nested hierarchy over chaining animations, path animations or additive animations
  - simpler animations
  - easier to visualise in your head
  - easier to change

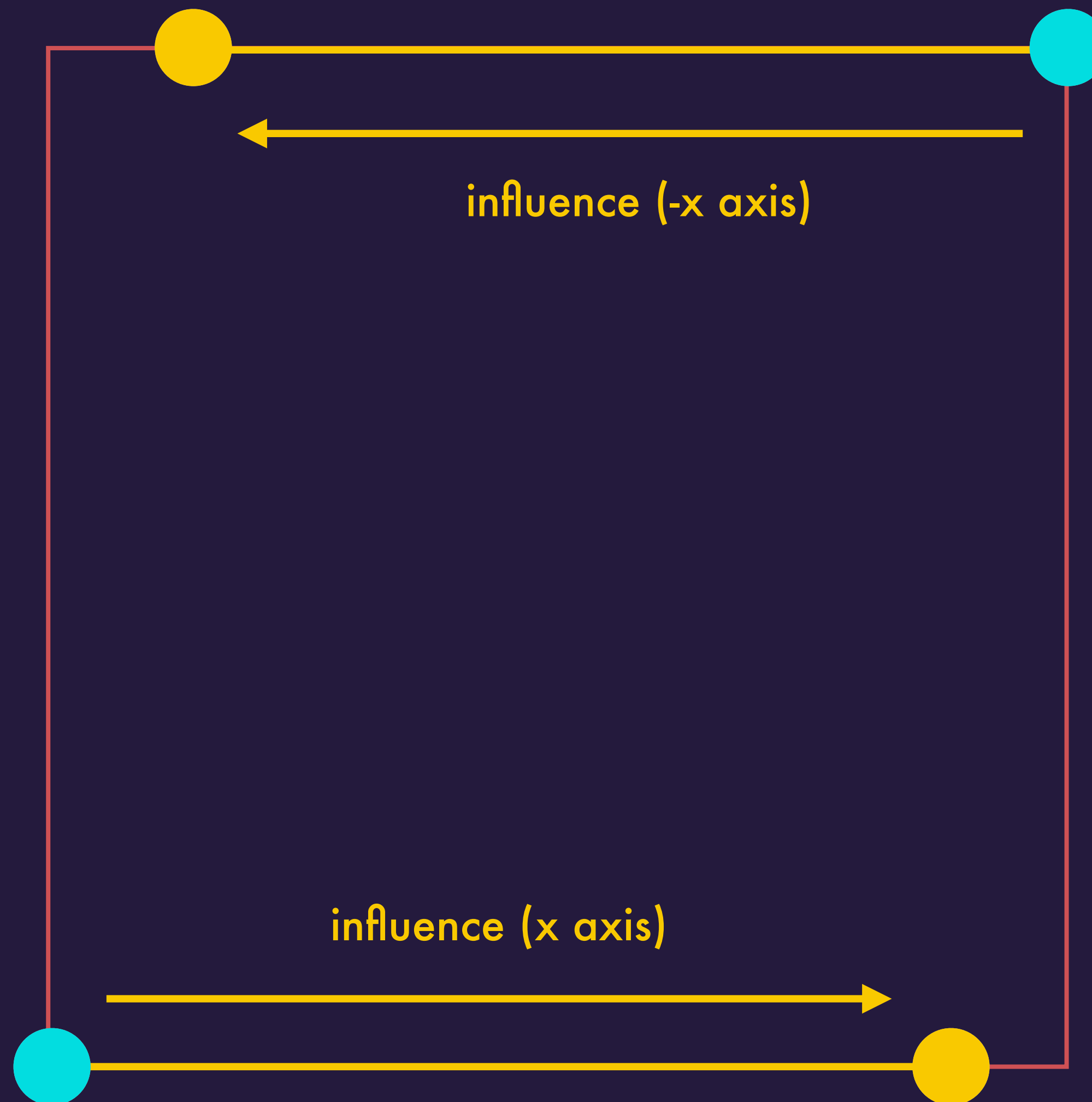
# Designing Animations

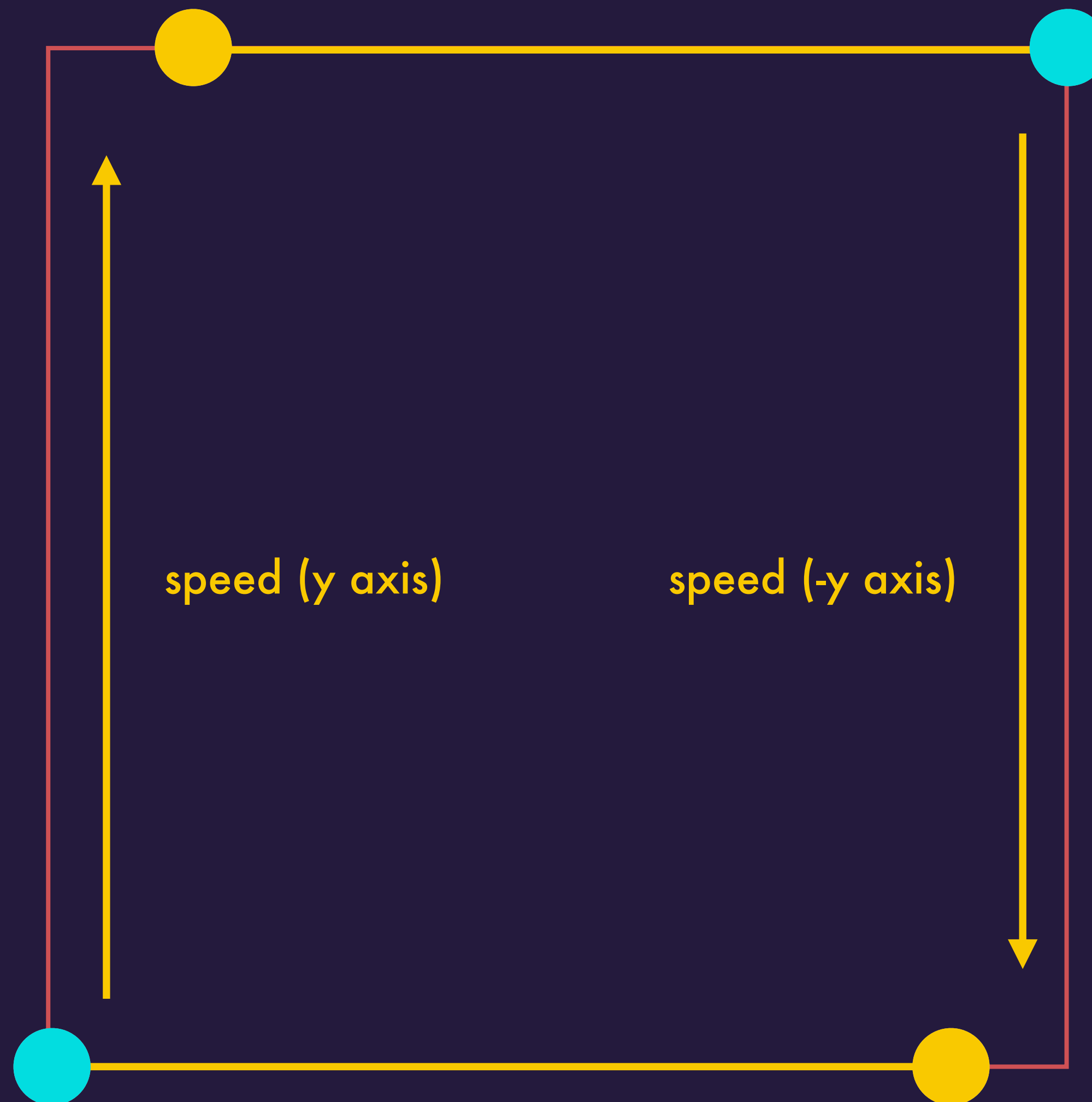


```
CAMediaTimingFunction(controlPoints: 0.9, 0.0, 0.1, 1.0)
```









- Use bezier curves to define timing
  - intuitive
  - easy to chain
- Use overshoot, undershoot (etc.) to add character to your animations
- cheat sheet: <http://cubic-bezier.com>

# Example

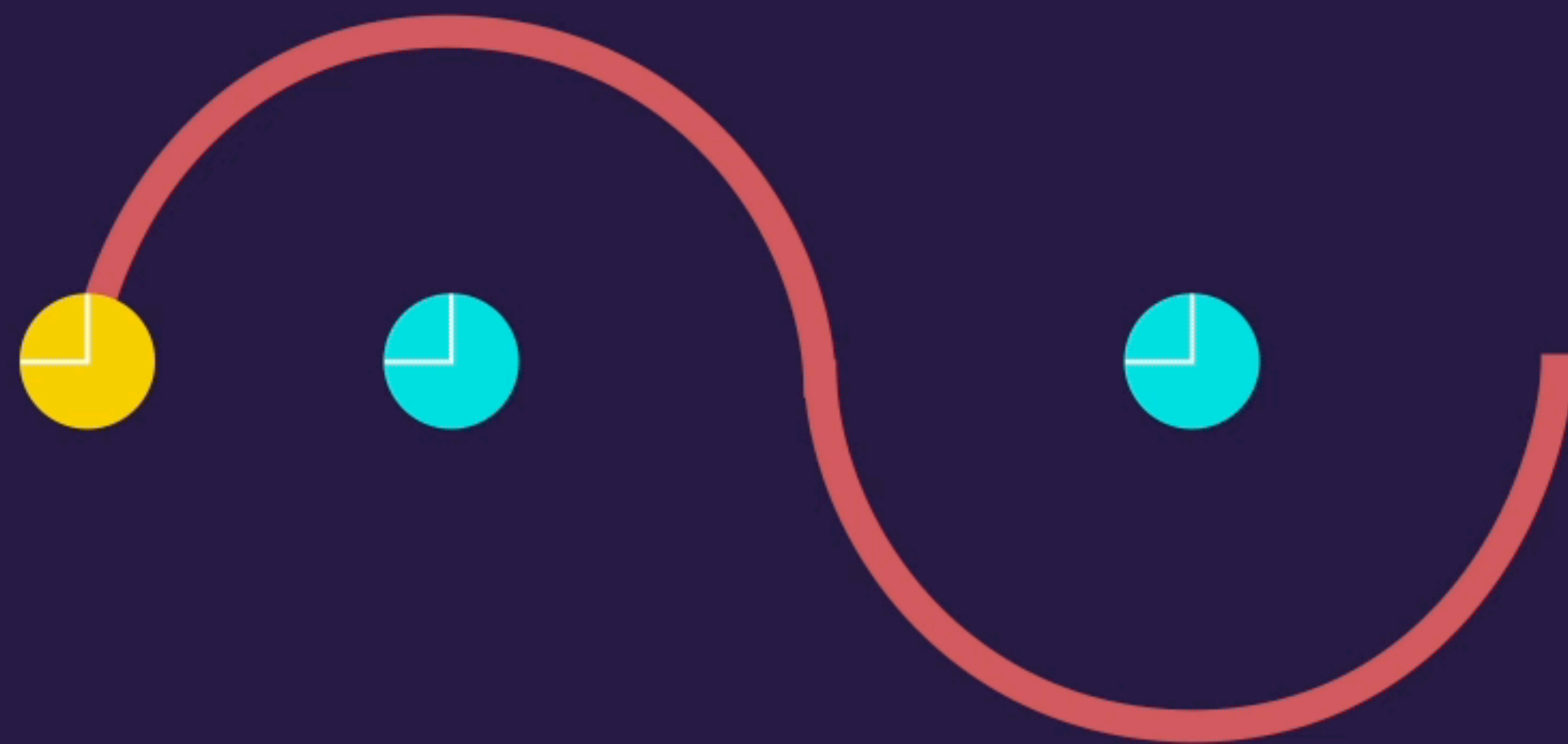


@marcus\_eckert









# Demo

- Set your test animation up to be repeatable

- `DispatchQueue.main.async...`
- Gesture Recognizer

- Use breakpoints to avoid recompiling

- Tick «Automatically continue after evaluating actions»
- add an expression with «Add Action»: «e [your code]»

Your turn.

GitHub: [github.com/marcuseckert/aveiro](https://github.com/marcuseckert/aveiro)



Output:

Scenes/Skeleton.swift

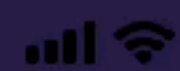
Example:

Scenes/ExampleCircleLine.swift

Animations:

Resources/TargetA.mov

Resources/TargetB.mov



09:41



Task

 @marcus\_eckert