GAME 3004

SpriteKit - Week 4

Lesson 4

Expectation

Introduction to **SpriteKit**

Outcome

Understanding SKScene's **Rendering Loop** and SKScene's **Node Tree**

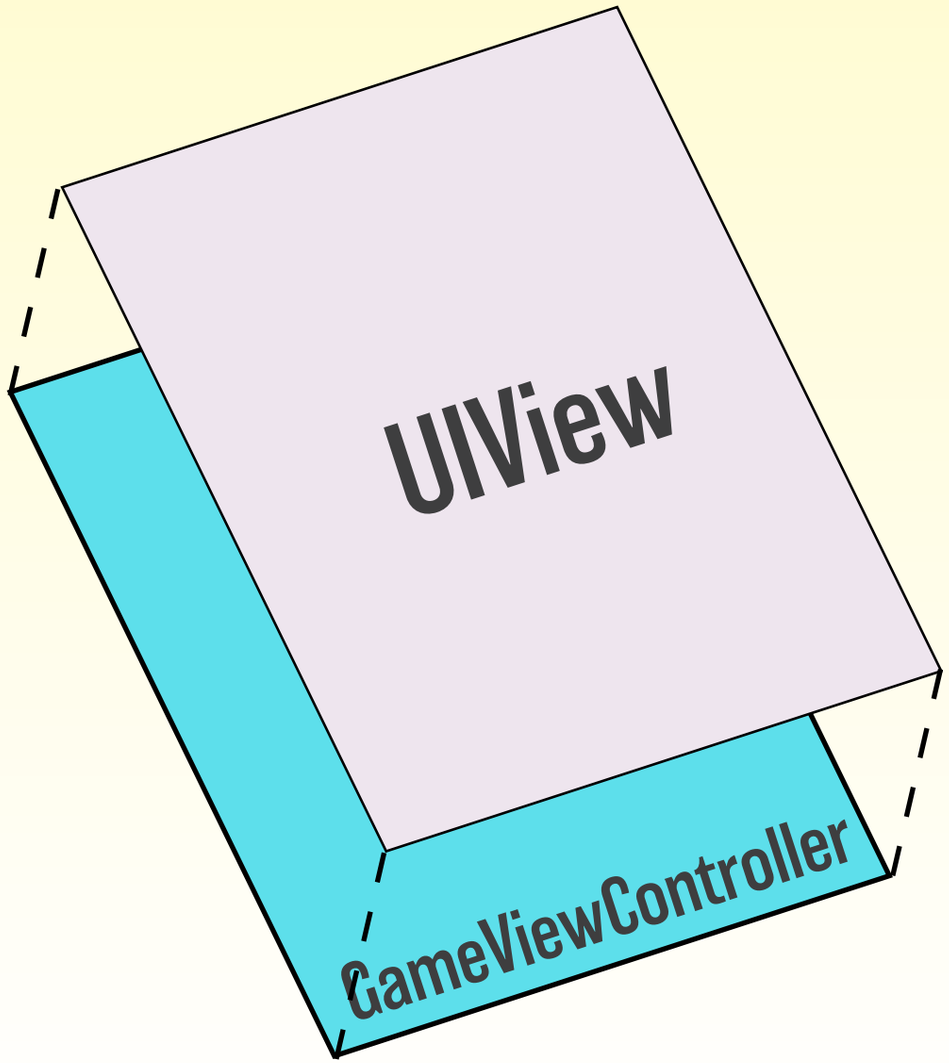
Key Concepts

SKScene Review
SKScene Rendering Loop
SKScene Node Tree
Rendering Nodes
Searching Nodes
Coordinates
Anchor Points

SKScene Review

1. Create the **GameViewController**

2. Have the **GameViewController** create its **UIView**



SKScene Review

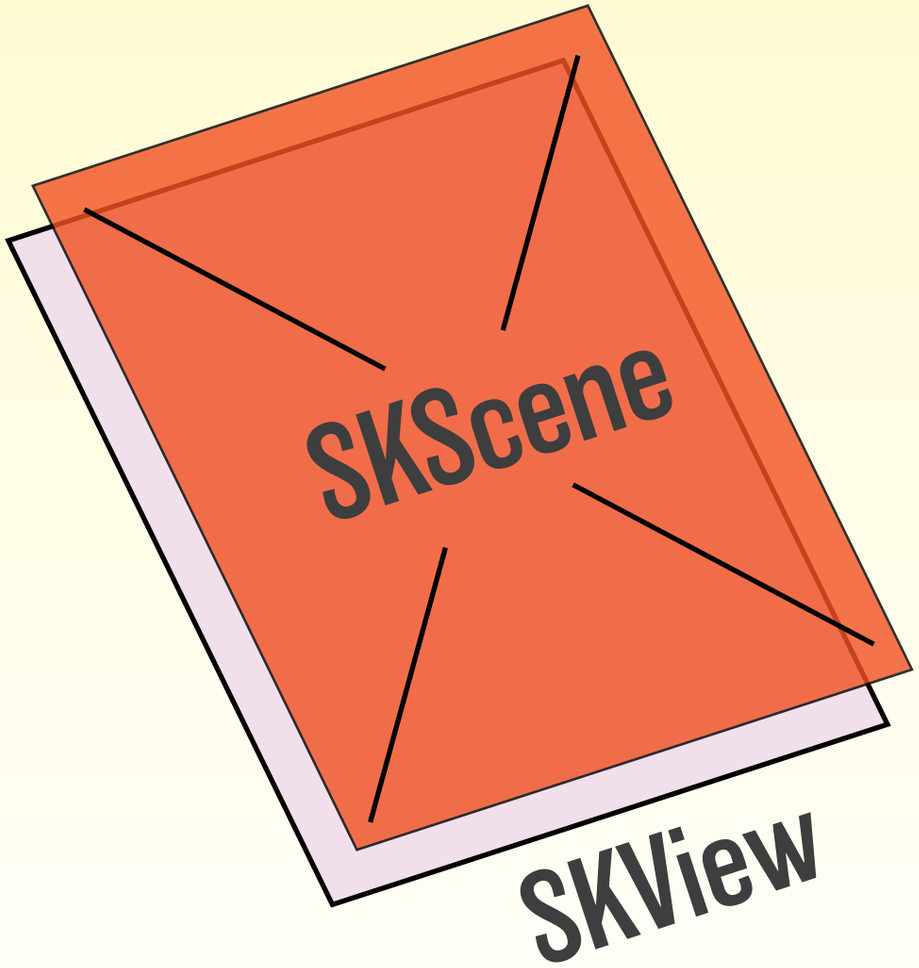
3. Inside the **GameViewController.viewDidLoad()**, down-cast the **UIView** to an **SKView** and set the **showFPS** property to **true**:

```
let skView = view as! SKView
skView.showFPS = true
```

SKScene Review

4. Create an instance of the SKScene named **scene**, passing it its size in the constructor and setting the scaleMode property:

```
scene = GameScene(size: skView.bounds.size)
scene.scaleMode = .aspectFill
```



SKScene Review

5. Inside the **init()** of the scene, we added the **backgroundNode and playerNode** objects in the scene:

```
let backgroundNode = SKSpriteNode(imageNamed: "Background")
backgroundNode.position = CGPoint(x: size.width / 2.0, y: 0.0)
addChild(backgroundNode)

let playerNode = SKSpriteNode(imageNamed: "Player")
playerNode.position = cGPoint(x: size.width / 2.0, y: 80.0)
addChild(playerNode)
```

SKScene Review

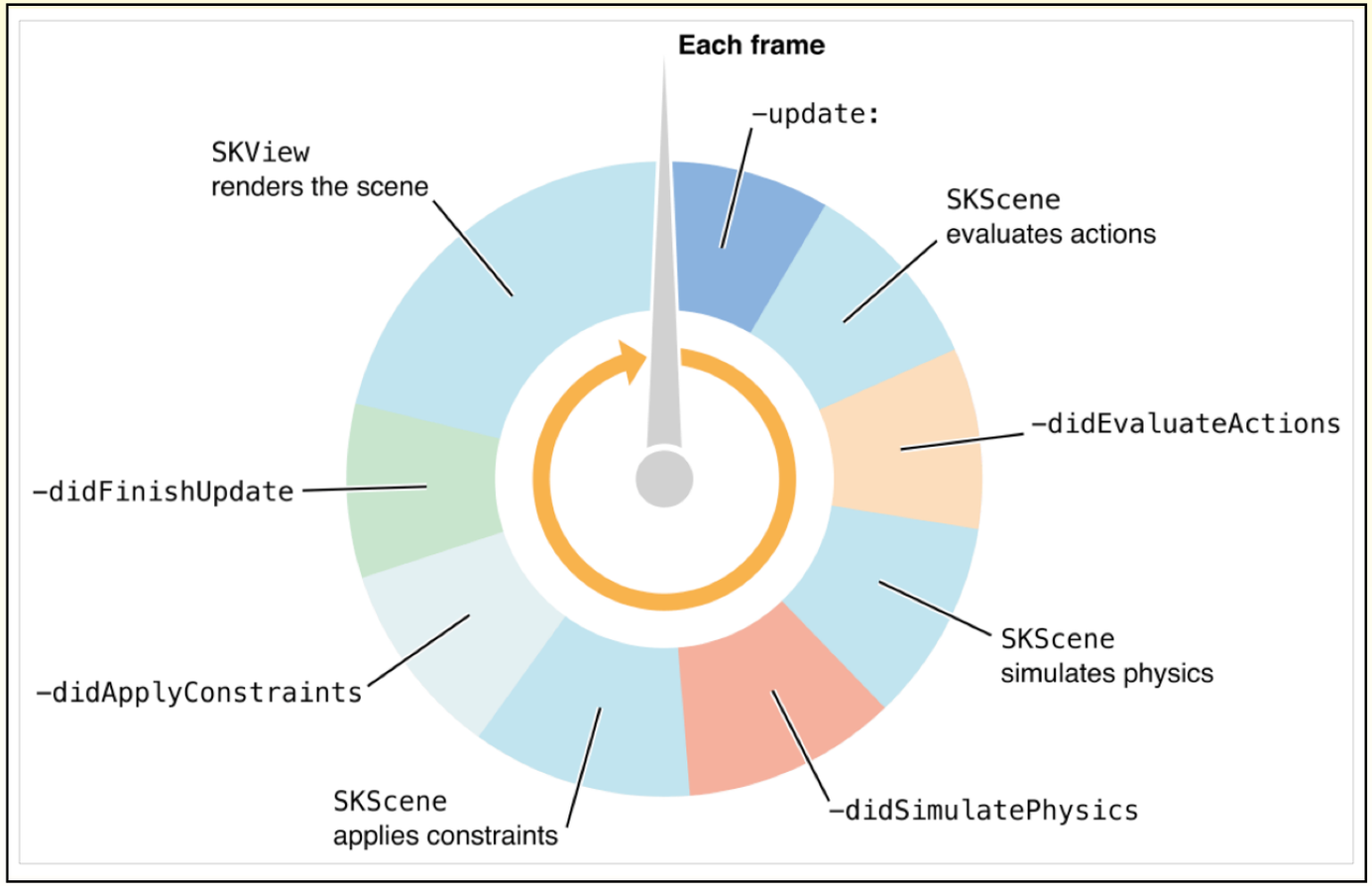
6. Present the complete scene in the **GameViewController** **viewDidLoad** method:

skView.presentScene(**scene**)

SKScene Rendering Loop

Each **iteration** of this **loop** generates the next frame in the scene

The **loop** has several steps that are called for **actions**, **physics**, **and constraints** on the scene



SKScene Rendering Loop

The scene calls its **update()** method

Called **before** actions are evaluated

This is where you will have most of your **game logic**

update() 1

Primary place to handle **AI**, **game scripting**, **input handling**, **actions** or other game logic.

SKScene Rendering Loop

Actions Perfomed	2
didEvaluateActions()	3

All actions will be performed

Next, the scene calls the **didEvaluateActions()**

Any **post-action game logic** can be put here

Test the position of a node, **after actions** were performed

SKAction

An **action** is an object that defines **a change** you want to make to a SKNode

Use an action when you are:

Animating a node through a series of textures
Modifying the **position**, **scale or rotation** of a node
Changing the **visibility** of a node
Adjusting the **translucency** of a node using its alpha property
Changing the **colour** of a node

SKAction - Move By

```
let moveNodeUp = SKAction.moveBy(x: 100.0,  y: 100.0, duration: 1.0)
node.run(moveNodeUp)
```

SKAction

```
let moveNodeUp = SKAction.moveBy(x: 100.0,  y: 100.0, duration: 1.0)
node.run(moveNodeUp)
```

1. To create an **action**, call the class method for the action you are **interested** in
2. **Configure** the action's **properties and duration**
3. Call a node's **run(_:)** method and **pass** it the action object

SKAction - Run()

You can **add unique keys** to identify actions added to a node

```
let moveNodeUp = SKAction.moveBy(x: 0.0, y: 100.0, duration: 1.0)
rocketNode.run(moveNodeUp, withKey: "ignition")  1
```

1 Adding a key allows you to remove **specific actions**, and check to see if an action is already **running**

SKAction - Run() Completion Block

```
let moveNodeUp = SKAction.moveBy(x: 0.0, y: 100.0, duration: 1.0)
rocketNode.run(moveNodeUp) {
    print("Rocket fired")        1
}
```

1 Here we added a **completion block** to do something once the action is **finished**

SKAction - Removing

There are different ways to remove an action

Use **removeAction(forKey:)** to remove an action associated with a **specific key**

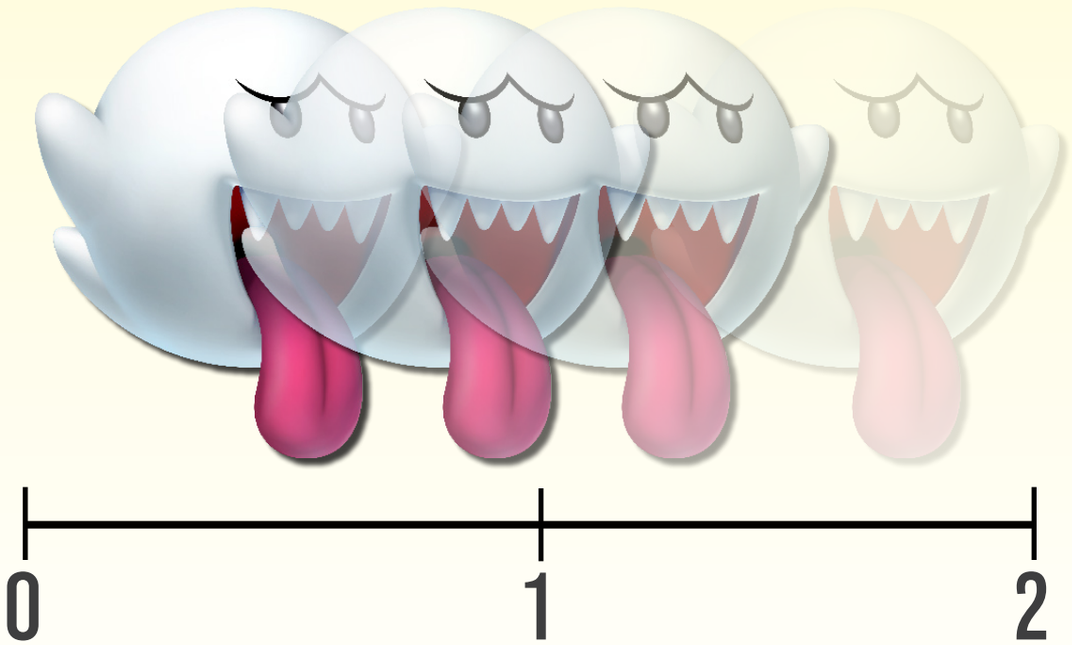
removeAllActions() ends and **removes all actions** from the node

SKAction - Scale To

```
let scaleTo = SKAction.scale(to: 2.0, duration: 0.25)
node.run(scaleTo)
```

SKAction - Fade Out

```
let fadeOut = SKAction.fadeOut(withDuration: 2.0)
node.run(fadeOut)
```

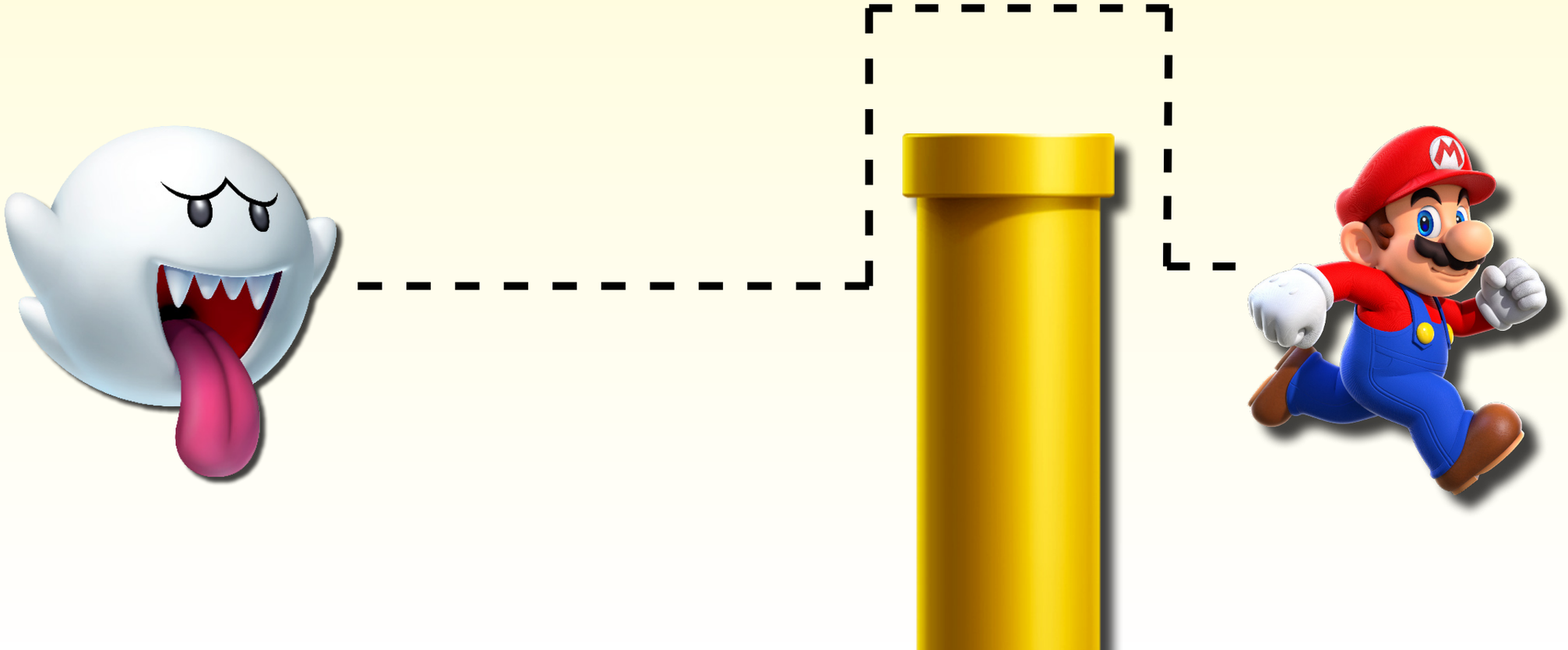


SKAction - Rotate By

```
let rotateBy = SKAction.rotate(byAngle: 45.0, duration: 2.0)
node.run(rotateBy)
```

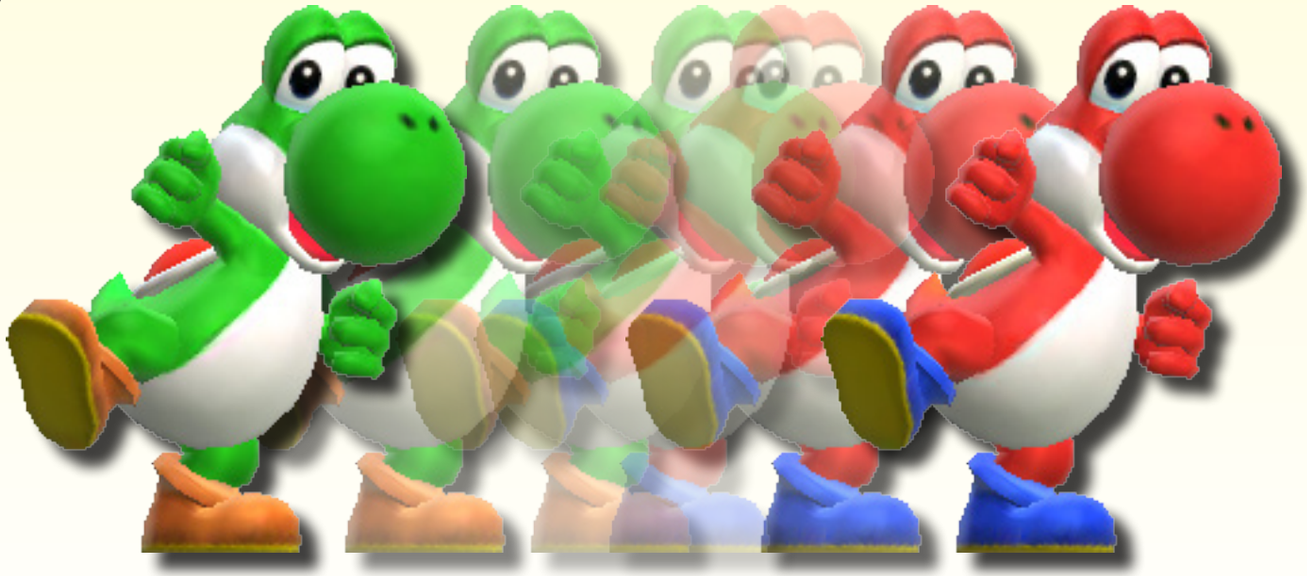
SKAction - Follow Path

```
let follow = SKAction.follow(CGPath, speed: 2.0)
node.run(follow)
```



SKAction - Change Colour

```
let toRed = SKAction.colorize(with: UIColor.red,
colorBlenderFactor: 0.4, duration: 2.0)
node.run(toRed)
```



SKAction Timing

By default, an SKAction **runs linearly**

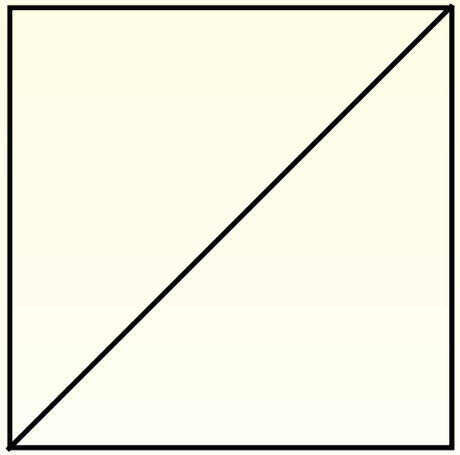
You can use an action's **timingMode** property to choose a **non-linear timing** mode for an animation

You can change the **speed** of an action by setting it's **speed value**

SKAction Timing

linear - Specifies linear pacing. **Linear pacing** causes an animation to **occur evenly** over its duration

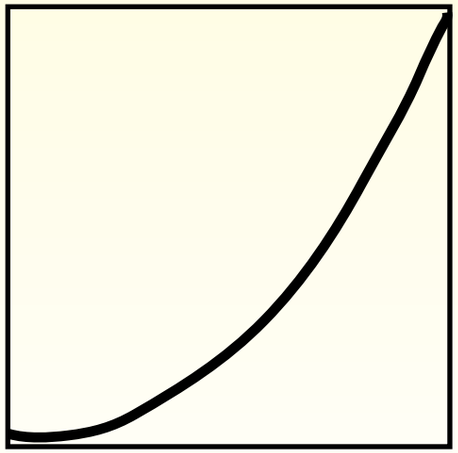
SKAction.**timingMode** = **.linear**



SKAction Timing

easeIn - Specifies ease-in pacing. **Ease-in** pacing causes the animation to **begin slowly** and then **speed up** as it progresses

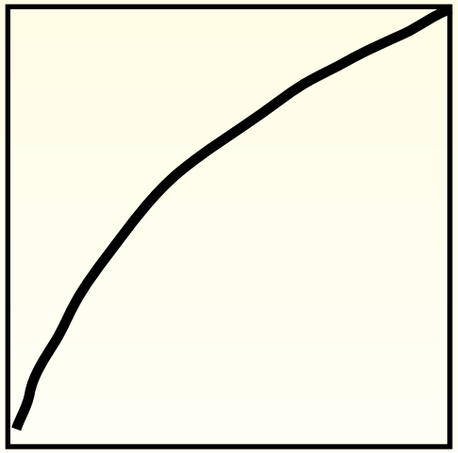
SKAction.**timingMode** = **.easeIn**



SKAction Timing

easeOut - Specifies ease-out pacing. **Ease-out** pacing causes the animation to **begin quickly** and **then slow** as it completes.

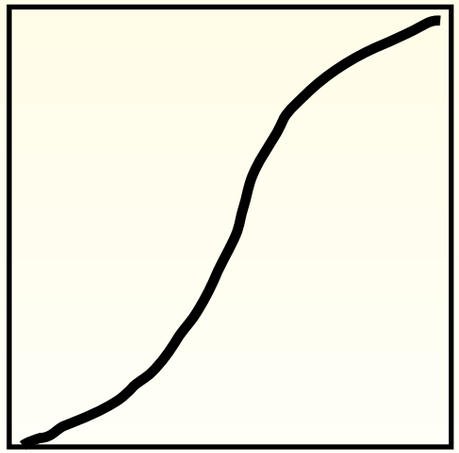
SKAction.**timingMode** = **.easeOut**



SKAction Timing

EaseInEaseOut - Specifies ease-in easeOut pacing. An **ease-in ease-out** animation **begins slowly**, **accelerates** through the **middle** of its duration, and then **slows again** before completing

SKAction.**timingMode** = **.easeInEaseOut**



SKAction Timing

Speed up or **slow down** an action's speed property using the **SKAction**.speed value. By default it is set to **1.0**

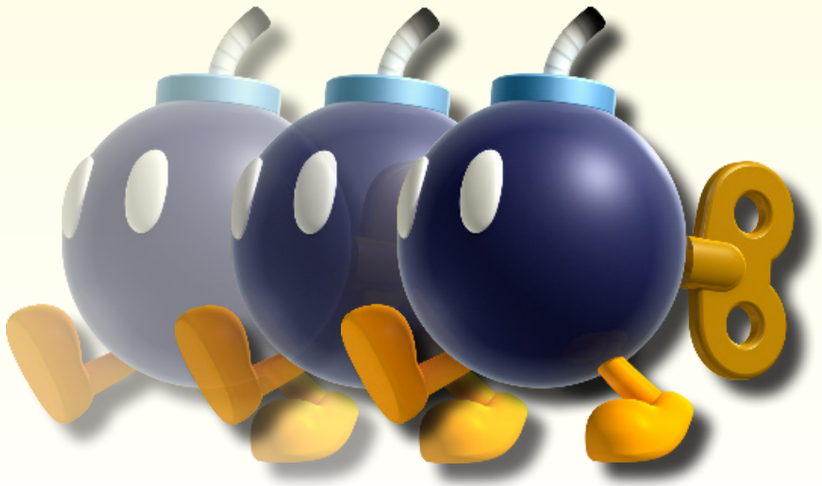
action.**speed** = **2.0**

action.**speed** = **0.5**

SKAction Timing

Use the **reversed()** function to play an action **backwards**

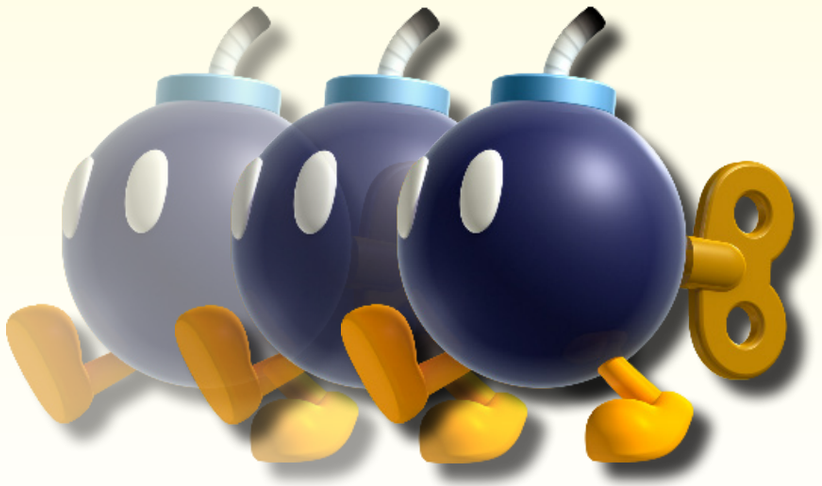
```
let action = SKAction.moveBy(x: 50.0, y: 0.0, duration: 2.0)

node.run(action.reversed())
```



SKAction Timing - Pausing

To pause an action you can set the speed to 0

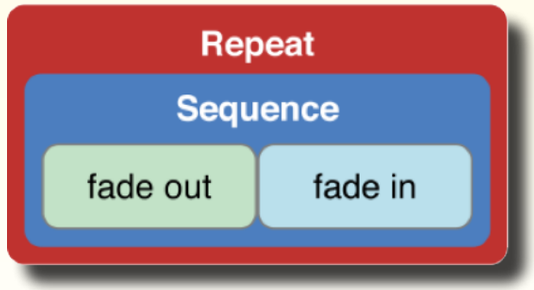
action.speed = 0

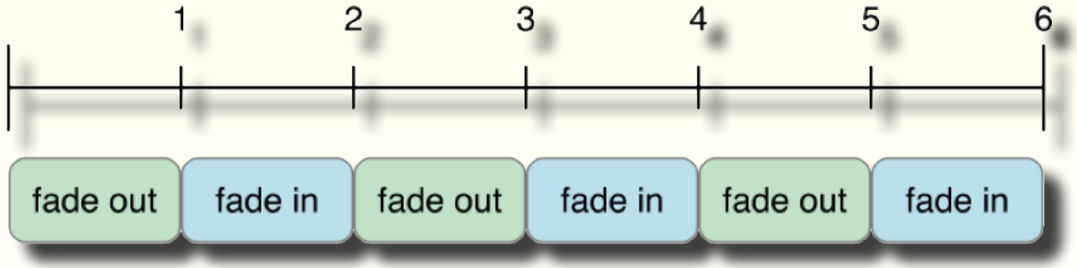


SKAction - Repeat

A **repeating action** has a single child action
When the action completes, it is **restarted**

You can repeat for a set amount of **times** or have it repeat **forever**





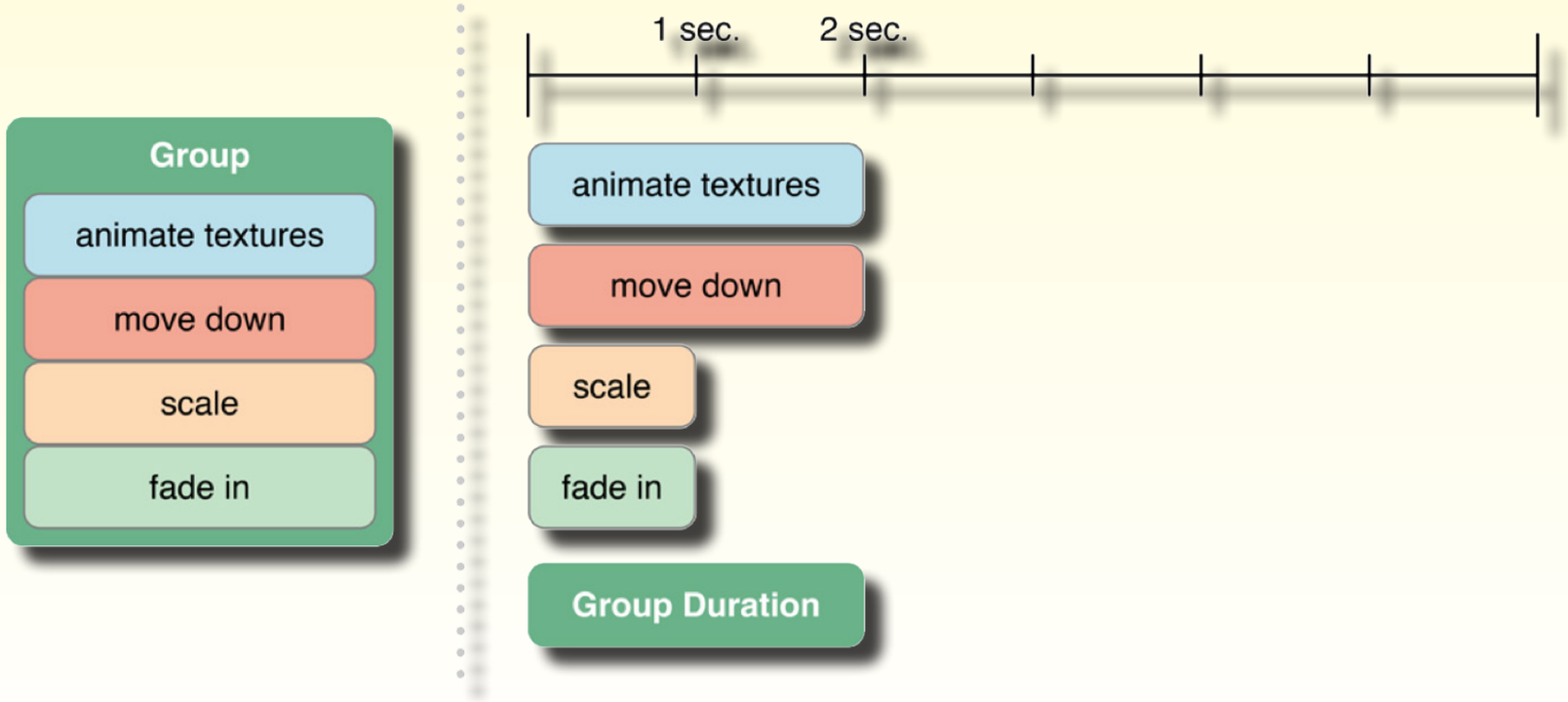
SKAction - Repeat

```
let repeat = SKAction.repeat(bite, count: 3)
node.run(repeat)

let repeatForever = SKAction.repeatForever(bite)
node.run(repeatForever)
```

SKAction - Group

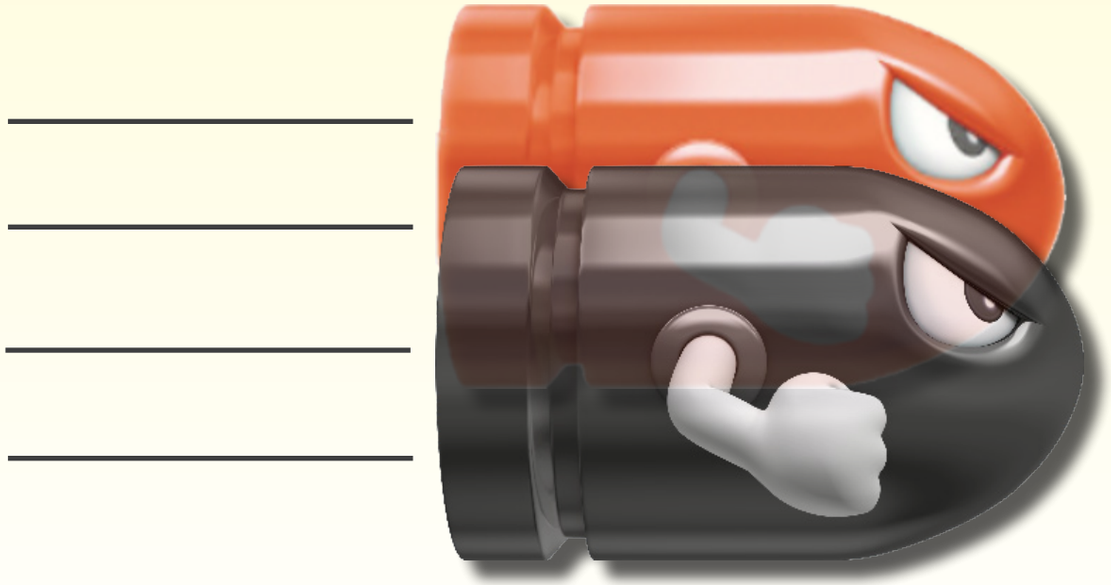
A **group action** has **multiple** child actions

All actions stored in the group begin executing at the **same time; in parallel**



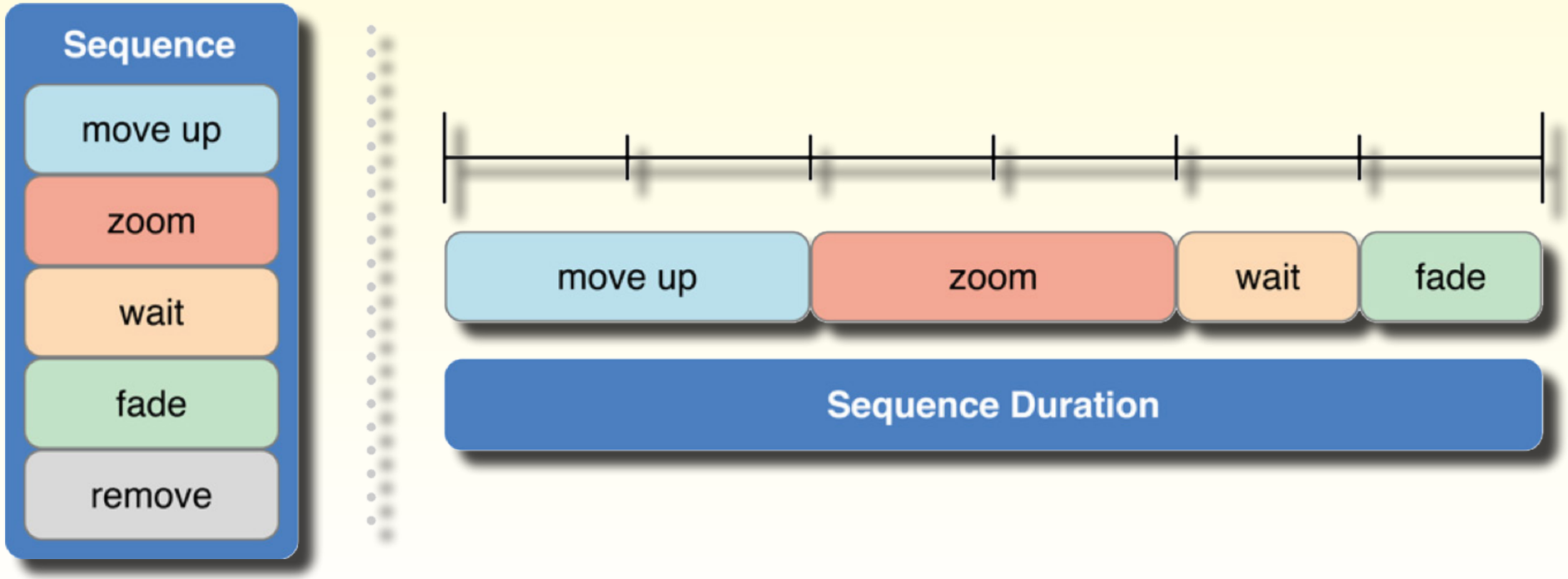
SKAction - Group

```
let group = SKAction.group([fire, goRed])
node.run(group)
```



SKAction - Sequence

The **sequence action** allows you to **chain** together a sequence of actions that are performed **in order**, one at a time

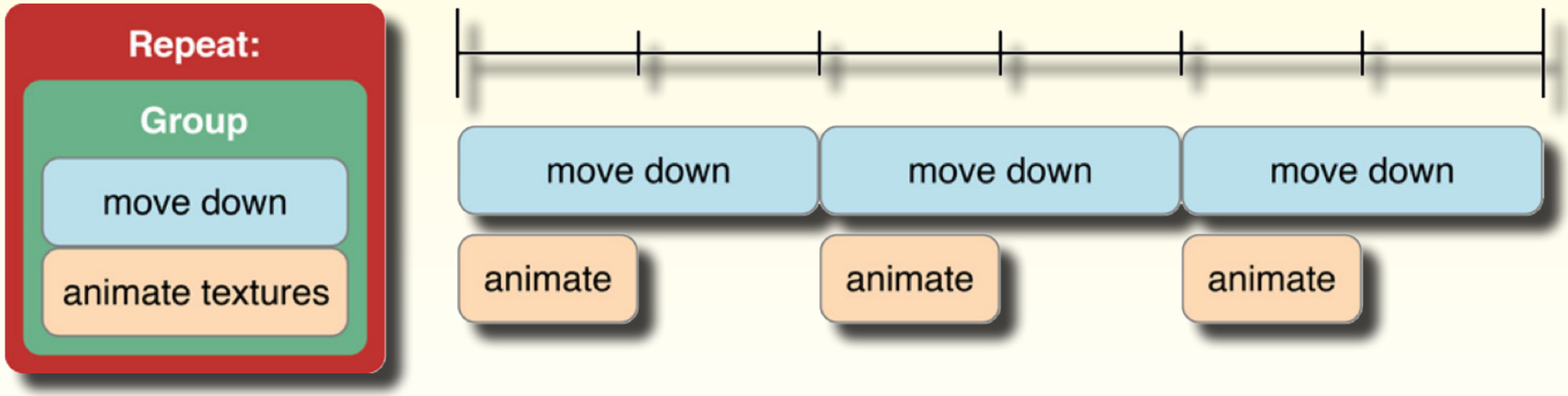


SKAction - Sequence

```
let sequence = SKAction.sequence([peek, throw, hide])
node.run(sequence)
```

SKAction - Repeating Sequences

You can repeat sequences and groups to create **very expressive** animations



SKAction - Wait

The **wait action** is a special action that is usually used in a sequence. This action simply **waits for a period** of time and **then ends**

SKAction.**wait(**forDuration: **5.0)**