with space in
between!

Apple’s Sprite Kit Framework

@PhillipKessels, Cocoaheads Aachen - May 2014

man phikes

ity RWTH RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN

nology

ask any time!

What We Will See/Do Today

- **What** is Sprite Kit?
- **How** does Sprite Kit work?

Hands-On

- Lessons **learned**

Hands-On

- Create a Scene
- Add Player and Enemy
- Shoot

What is Sprite Kit?

"Sprite Kit provides a graphics rendering and animation infrastructure that you can use to animate arbitrary textured images, or **sprites**. "

- Sprite Kit Programming Guide

2D graphics

"Sprite Kit also provides other functionality that is useful for games, including basic sound playback support and physics simulation."

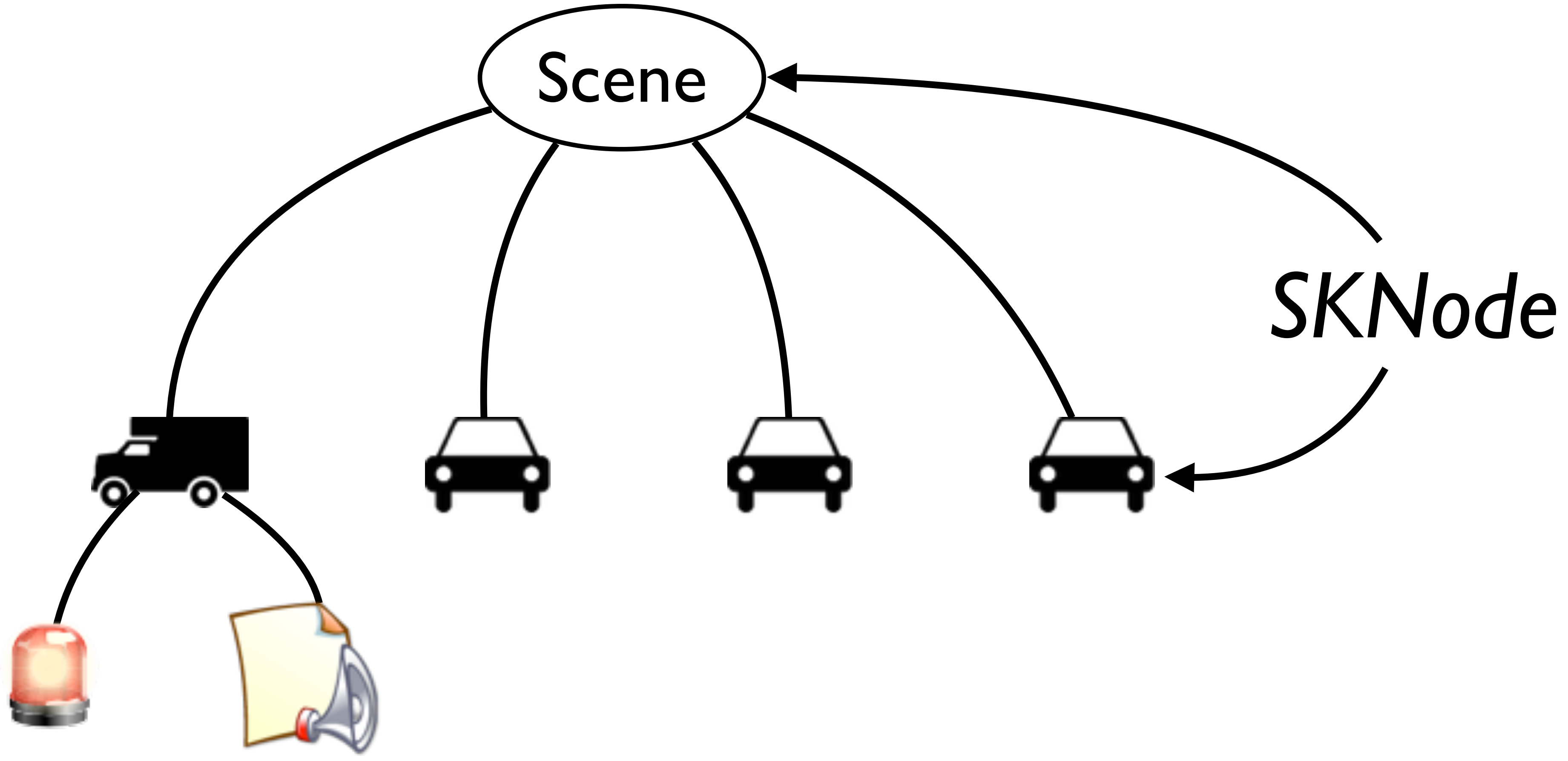
- Sprite Kit Programming Guide

audio, basic physics

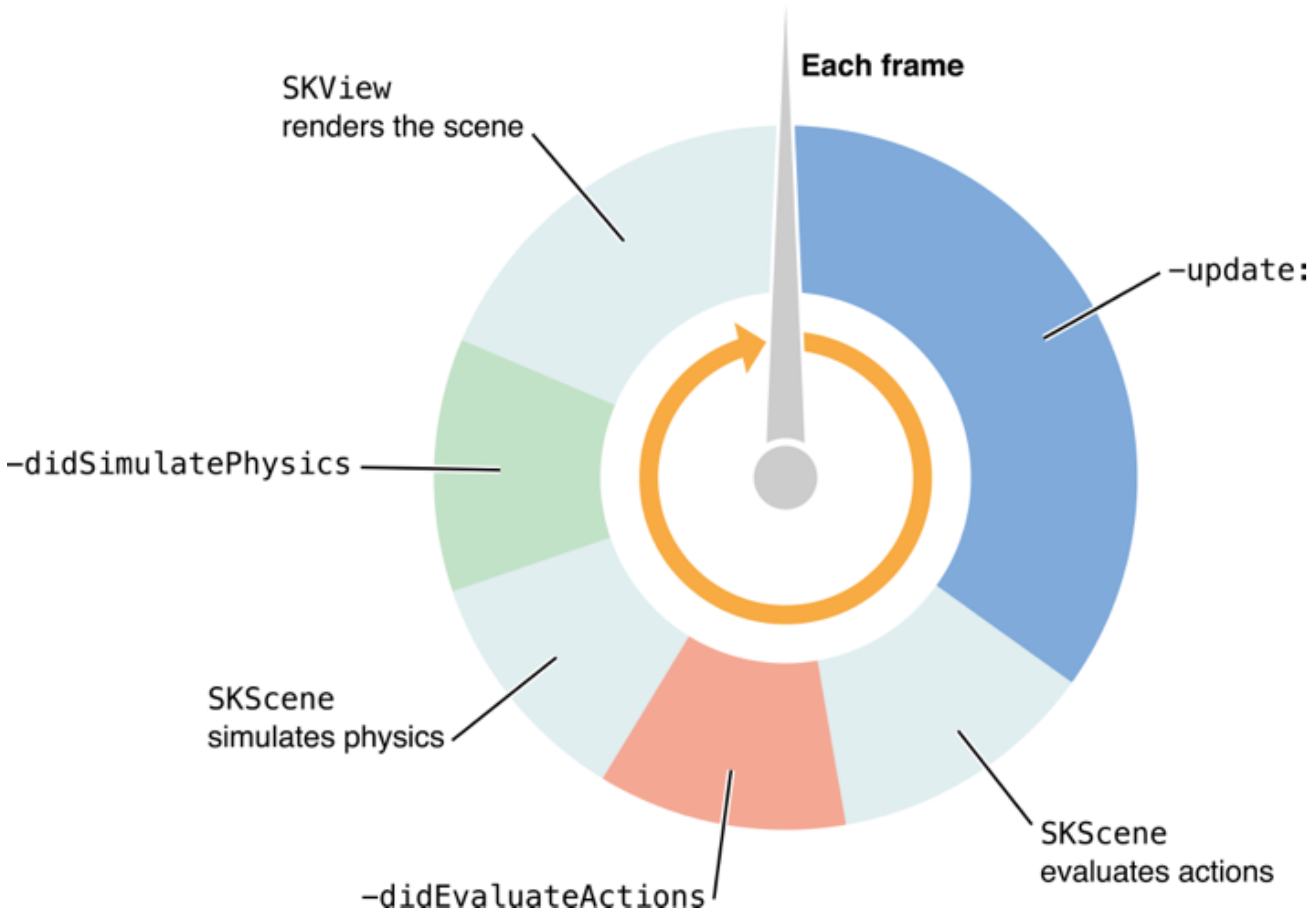
How does Sprite Kit work?

- Representation of **Scene as Graph**

(Includes graphics, sounds, effects, ...)



How does Sprite Kit work?



Hands-On

Get repository URL from git.io/**8HIUbg**

copy it here

```
git clone <URL>
```

Hands-On

- **Create a scene**
- Add player and enemy
- Shoot

Create a Scene

```
git checkout 9d8ba79
```

- **SKView (< UIView)** can present **SKScene**
- **SKView** allows for debugging (FPS, Nodes, ...)
- **SKScene** is the root of the scene graph
  - Added **SKLabelNode** as example

Hands-On

- Create a scene
- **Add player and enemy**
- Shoot

Add Player and Enemy

```
git checkout fcc365b
```

- **SKSpriteNode (< SKNode)** can present images ("sprites") in the scene
- **Defines** can be used for extracting static information (e.g. configuration)

Hands-On

- Create a scene
- Add player and enemy
- **Shoot**

Shoot

```
git checkout 97a11f7
```

- **SKNode** can run **SKAction** (e.g. moving)
- **SKScene** handles touches

Intermezzo - Software Design

- We tended to mix all kinds of stuff up
- Generally yields large classes with lots of responsibilities
- Our attempt is to use **Model-View-Controller**
- This is by far more complicated than in web e.g. → different layers

MVC Approach

- **Model:** Things related to *game logic*

- **View:** *Visual* representations of game logic, *Physical* representations of game logic

- **Controller:** Code which *communicates* changes from the View (collision, etc.) to Model

This is a matter of convention.

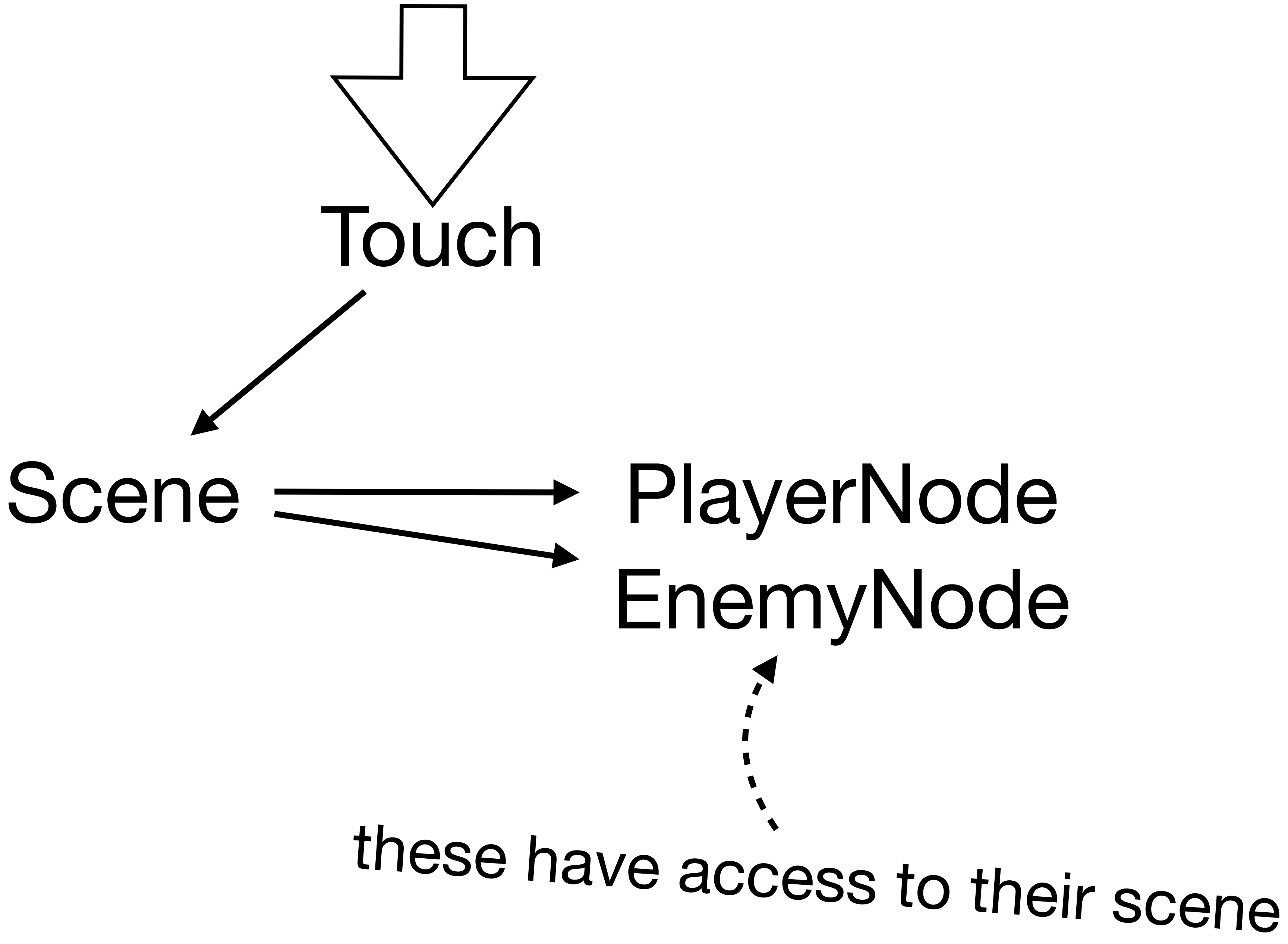
MVC Approach

- **Model:** Enemy, Player, Shot
- **View:** EnemyNode, PlayerNode, ShotNode
- **Controller:** SKScene

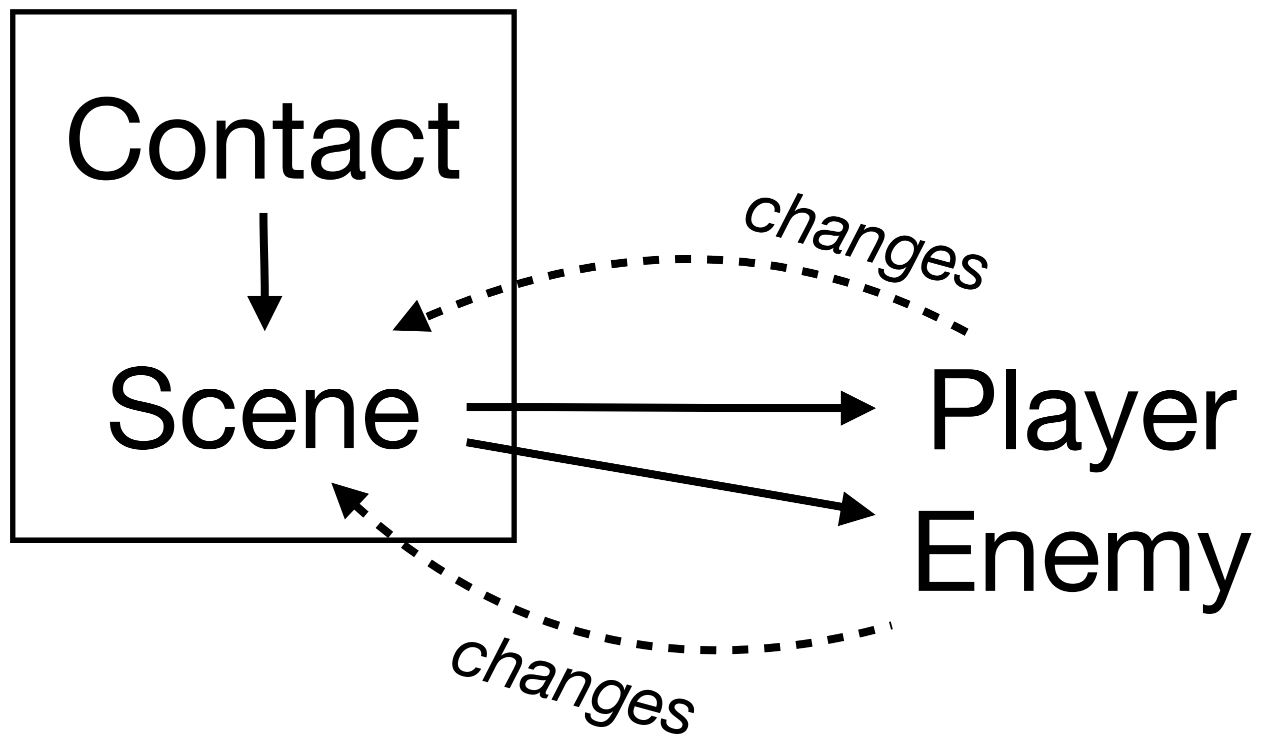
MVC Approach

- I only implemented View, but this is already much cleaner
- I used *Dependency Injection* on EnemyNode for its **connection** to the PlayerNode
- Proposal for **handling** view/model **communication**: Dictionary of view/model instances on Scene, *Double Dispatch Pattern*

MVC Approach - Handling Touches



MVC Approach - Handling Contact



MVC Approach - Conclusion

- Slightly more complicated code
- but cleaner
- Much easier to maintain
- This is only a possibility, see if it fits your needs

Lessons learned

- Sprite Kit is easy to set up
- Basic building blocks
  - SKView
  - SKScene
  - SKNode, SKSpriteNode, SKLabelNode
  - SKAction
- Software design is extremely important for the quality of the game/app

Questions?

- Fin-