iOS Dev Accelerator
Week 7 Day 1

- SpriteKit
- Linked Lists

Sprite Kit

- “Sprite Kit provides a graphics rendering and animation infrastructure that you can use to animate arbitrary textured images, or sprites.”
- Sprite Kit use your device’s hardware to efficiently animate your games.
- Also includes sound playback, physics simulation, special effects, and texture atlases.

SKView and SKScenes

- The root object of a sprite kit stack is the SKView.
- SKView is just like a UIView, and it is placed inside of a window so it can start rendering content.
- The content of your game is organized into SKScene objects. Think of these as the levels of your game. A Scene can also be your main menu and your end game credits.
- Only one SKScene can be presented at any time by the SKView.

SKNodes

- Nodes are the ‘fundamental building blocks’ for all your game content.
- The SKScene class is actually a subclass of SKNode, and will act as the root node for all the nodes in the level.
- Just like a UIView and its SuperView, A Node’s position is specified in the coordinate system of its parent.
- You typically don't directly instantiate an instance of SKNode, instead you instantiate one of its myriad subclasses:

SKNodes

Class	Description
`SKSpriteNode`	A node that draws a textured sprite.
`SKVideoNode`	A node that plays video content.
`SKLabelNode`	A node that renders a text string.
`SKShapeNode`	A node that renders a shape based on a Core Graphics path.
`SKEmitterNode`	A node that creates and renders particles.
`SKCropNode`	A node that crops its child nodes using a mask.
`SKEffectNode`	A node that applies a Core Image filter to its child nodes.

SKActions

- Your scene's nodes are brought to life by using instances of the SKAction class.
- You tell Nodes to execute actions you have defined.
- There are common actions like moving, scaling, rotating, transparency, etc.
- Actions can also do things like execute code, change the node tree, and manage children actions (very common).

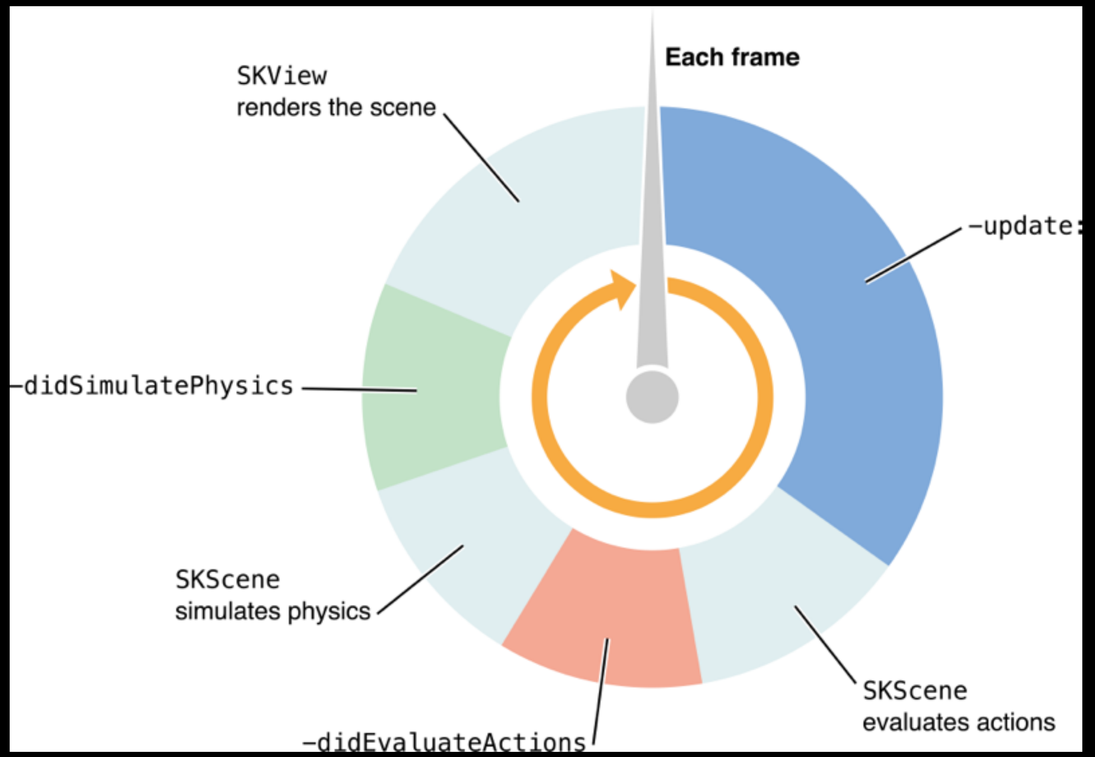
SKActions with Children

- Sequence action: each action in the sequence begins after the previous action ends.
- Group action: All actions stored in the group begin executing at the same time.
- Repeating Action: When the child action completes, it is restarted.

SpriteKit & Physics

- Besides using actions, you can let Sprite Kit's physics simulation system take care of things like nodes colliding and falling realistically.
- To achieve this, you create instances of SKPhysicsBody class and attach them to your nodes.
- Each Physics Body is defined by shape, size, mass,etc.
- A lot of the forces are applied automatically once the bodies are attached to the node, but you can also explicitly apply your own forces on the bodies.
- You have complete control over which nodes can collide and contact with other nodes.
- Your scene can define global physics characteristics by having an SKPhysicsWorld object attached to it.
- The SKPhysicsWorld has a contact Delegate, which has a contact method fired every time two nodes collide that are enabled to collide.

The update Loop



- In a regular view system, like in our View Controllers using UIKit, the contents of our views are rendered once and then only rendered again when their contents change.
- SpriteKit is designed to handle much more dynamic content, so it is continuously updating the scenes contents and rendering the updates.
- This is called the update loop, and it is a fundamental concept of game programming.

The update Loop

- Each time through the update loop, the scenes contents are updated and then rendered:
    1. The scene's update: method is called with time elapsed so far in the simulation. (This is the primary method you will use to implement your own in-game logic, AI, scripting, and input handling.)
    2. The scene processes actions on all the nodes in the tree.
    3. The scene's didEvaluateActions method is called after all actions for the free have been processed.
    4. The scene simulates physics on the nodes that have physics bodies.
    5. The scene's didSimulatePhysics method is called after all physics for the frame has been simulated.
    6. The Scene is rendered.

Collisions and Contacts

- Contact is used when you need to know that two bodies are touching each other.
- Collision is used to prevent two bodies from occupying the same space. Sprite Kit will automatically compute the results of the collision and applies the appropriate impulses.
- Every physics body has a category. Each scene can have up to 32 categories. When you configure a physics body, you define which categories it belongs to and which categories of bodies it wants to interact with.
- Contacts and Collisions are specified separately.

Defining your categories

- Each category is defined by a 32-bit mask (32 1's or 0's).
- When a potential interaction occurs, the category mask of each body is tested against the contact and collision masks of the other body.
- The test is logically ANDing the two masks together. If the result is a nonzero number, then that interaction occurs.
- If you don't set a collision mask on a physicsBody, it is all bits set to 1 by default.
- It's the opposite for contact mask, all zeros by default.
- Use the | bitwise operator to OR together multiple categories if you need a node to collide or contact with multiple other categories.

Linked Lists

- “A linked list is a data structure consisting of a group of nodes which together represent a sequence”
- Each node contains two things: some sort of data, and then a reference (or link) to the next node in the sequence.
- Very efficient insertion and removal.
- Linked Lists are a very simple data structure, and many other data structures actually use them as their core structure.

Linked Lists