SPRITE KIT

WHAT WE'LL COVER

- Nodes, include particle emitters
- Physics
- Touches
- Collisions
- Actions

BUILDING BLOCKS

STRUCTURE

- `SKView` displays your `SpriteKit` content
- `SKScene` provides the content to display and is the root node for your scene
- `SKNode` is the fundamental building block of most `SpriteKit` content.
  - `SKNode` is analogous to `UIView`, but the node doesn't draw any visual content.
  - `SKNode` has properties such as: `frame`, `alpha`, `isHidden`
  - `SKNode` can be identified via a `name` property

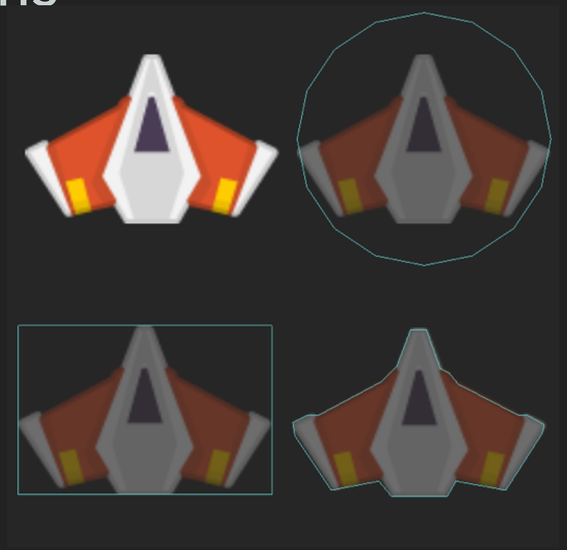
NODES WE'LL USE

- SKScene - root node of your scene
- SKSpriteNode - node that draws a rectangular texture, image or color
- SKEmitterNode - node that creates and renders particles

PHYSICS

PHYSICS

- `SKPhysicsWorld` - encapsulates a scene's physics simulations
  - Properties such as: `gravity`, `speed`, `contactDelegate`
- `SKPhysicsBody` - adds physics simulation to a node
  - Body Type: Circle, Rectangle, Alpha Mask
  - Nodes have a `physicsBody` property
  - Properties:
    - `isDynamic`, `allowsRotation`, `affectedByGravity`, `mass`, `density`, `friction`, `restitution`, `linearDamping`, `angularDamping`
  - Can be identified using `categoryBitMask`



TOUCHES

TOUCHES

- Use `UIKit` touch event methods to respond to touches.
  - `func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)`
  - `func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?)`
  - `func touchesEnded(_ touches: Set<UITouch>, with event: UIEvent?)`
- Uses touches set to get your touches.
  - `UITouch` contains `location` and `previousLocation` properties
- Use physics world, `body(at:)` method to determine which object is being touched

COLLISIONS

CATEGORY BIT MASKS

- `SpriteKit` uses category bit masks to identify different types of physics bodies
- Physics body have the following property
  - `var categoryBitMask: UInt32`
  - allows for 32 unique categories

Ball	`00000000000000000000000000000001`	`0x1 << 0`
Bottom	`00000000000000000000000000000010`	`0x1 << 1`
Brick	`00000000000000000000000000000100`	`0x1 << 2`
Paddle	`00000000000000000000000000001000`	`0x1 << 3`
Border	`00000000000000000000000000010000`	`0x1 << 4`

COLLISIONS AND BIT MASKS

- Physics bodies also have a `contactTestBitMask`
  - Provide a bit mask with the bits turned on (1) in each bit that corresponds to the bodies you want to be “notified” when a collision occurs.
    - Example: ball is interested in brick or bottom collisions
      - Use bitwise OR:

```
Bottom  00010
 Brick  00100
    OR  00110   Contact Test Mask
```

- When two objects collide the system will use the OR bit mask operation to generate, “collision bit mask”
  - example: ball and brick collide

```
  Ball  00001
 Brick  00100
    OR  00101   Collision Test Mask
```

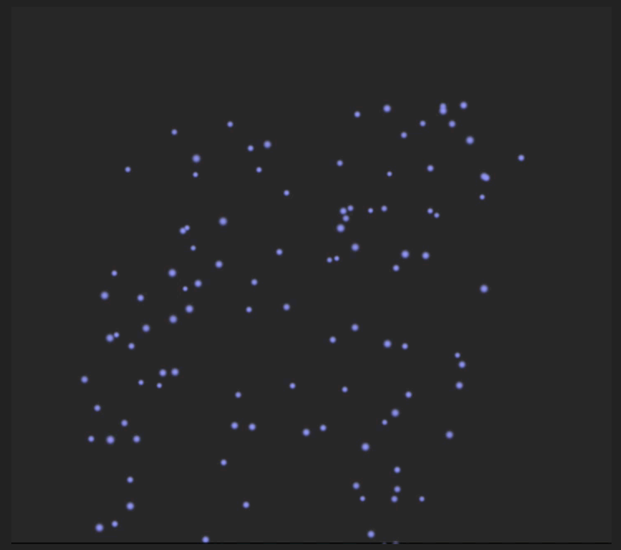
COLLISIONS AND BIT MASKS

- Using the contact test bit mask and the collision test bit mask, the system will use the bit wise AND to verify if the scene should be “notified” of the collision.
  - For example:
    Contact `00110`
    Collision `00101`
    AND `00100`
  - If there is a “1” in any column, the system call the delegate method
    - `func didBegin(_ contact: SKPhysicsContact)`
  - Using the `SKPhysicsContact` to determine the two bodies that collided, and take the appropriate action
    - Physics contact contains properties like: `bodyA` & `bodyB`

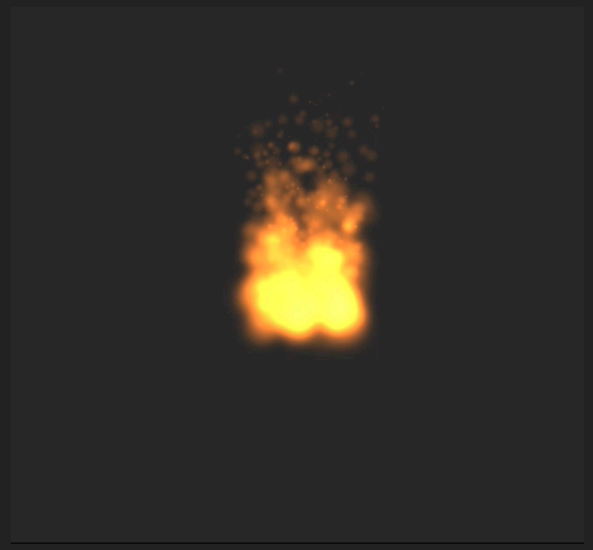
PARTICLE EMITTERS

PARTICLE EMITTERS

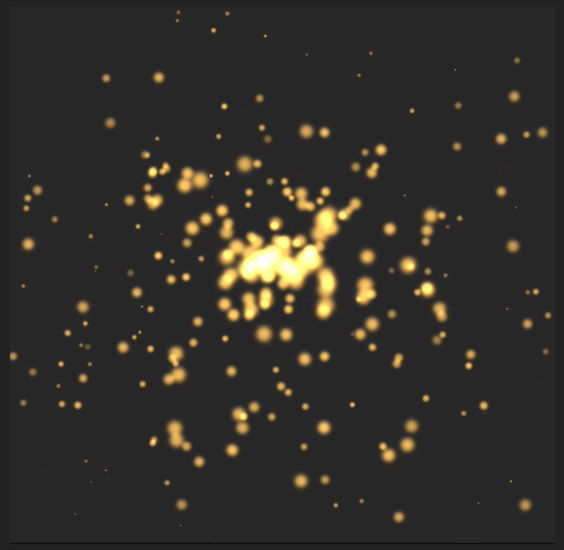
- `SKEmitterNode` - node that creates and renders particles
  - Several system defaults



Rain



Fire



Spark

PARTICLE EMITTERS

- `SKEmitterNodes` have properties such as:
  - `particleLifetime`, `particleLifetimeRange`
  - `particlePosition`, `particlePositionRange`
  - `particleSpeed`, `particleSpeedRange`
  - `emissionAngle`, `emissionAngleRange`
  - `particleRotation`, `particleRotationRange`
  - `particleTexture`
  - `particleSize`
  - `particleBirthRate`, `numParticlesToEmit`

ACTIONS

ACTIONS

- `SKActions` are an object that is executed by a node to change it structure or content.
  - Change the nodes position, orientation, size, scale
  - Change the nodes visibility, textures, colors
  - Play a simple sound
  - Remove a node
  - Call a closure or invoke a selector
  - Or any number of actions above combined into a sequence

ACTIONS

- Use the `run()` function on a node to execute your action
- Use “keys” to name your actions and retrieve them or perform the action

DEMO