SceneKit in Swift Playgrounds

Session 605

Michael DeWitt, Gem Collector
Lemont Washington, Draw Call Blaster
**Goal:** Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a for loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```python
for i in 1 ... 4 {
    moveForward()
    collectGem()
    moveForward()
    moveForward()
    moveForward()
    turnRight()
}
```
Prototyping
Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You'll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a for loop from the code library, then drop it above the existing code.
2. Tap the bottom curly brace to select the loop.
3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```java
for i in 1...4 {  
    moveForward()  
    collectGem()  
    moveForward()  
    moveForward()  
}  
moveForward()  
turnRight()
```
Early Feedback
Early Feedback

Change gem
Add border
Pivot camera at the end
Looping All the Sides

Goal: Use a for loop to repeat a sequence of commands.

In this puzzle, you must collect four gems that are located in the same relative locations around a square. You’ll create a loop that repeats the code below for each of the sides to solve the entire puzzle.

1. Drag a for loop from the code library, then drop it above the existing code.

2. Tap the bottom curly brace to select the loop.

3. Tap and hold that curly brace, then drag it downward to pull the existing code into the loop.

```python
for i in 1 ... number:
    moveForward()
    collectGem()
    moveForward()
    moveForward()
```

moveForward()

turnRight()
SpriteKit
SKScene
SKNode
SKAction

SceneKit
Effort to Change

Project Timeline

Prototyping
Prototyping
Prototyping

Test interaction model
Prototyping

Test interaction model

Interpret feedback
Prototyping

Test interaction model

Interpret feedback

Take the insights, leave the code
Iterating
Modeling the World
// World building API

let world = GridWorld(columns: 5, rows: 5)
// World building API

let world = GridWorld(columns: 5, rows: 5)

// Add an actor to the scene.
let actor = Actor()
world.place(actor, facing: north, at: Coordinate(column: 0, row: 0))
// World building API

let world = GridWorld(columns: 5, rows: 5)

// Add an actor to the scene.
let actor = Actor()
world.place(actor, facing: north, at: Coordinate(column: 0, row: 0))

// Place additional items.
let gems = [
  Coordinate(column: 0, row: 1),
  Coordinate(column: 1, row: 4),
  Coordinate(column: 3, row: 0),
  Coordinate(column: 4, row: 3)
]
world.placeGems(at: gems)

let water = world.coordinates(inColumns: 1..<4, intersectingRows: 1..<4)
world.placeWater(at: water)
Modeling the World
Separate data and visuals
Modeling the World
Separate data and visuals

Swap assets easily
Modeling the World
Separate data and visuals

Swap assets easily
Send gameplay logic
Modeling the World
Separate data and visuals

Swap assets easily
Send gameplay logic
Optimize the geometry
Modeling the World
Separate data and visuals

Swap assets easily
Send gameplay logic
Optimize the geometry
Need debugging tools ⚠️
Modeling the World
Separate data and visuals

Swap assets easily
Send gameplay logic
Optimize the geometry

Need debugging tools ⚠️
Animating Up
Animating Up
Animating Up

Use a ramp
Animating Up

Use a ramp
Animating Up

Use a ramp
Animating Up

Use a ramp
Animating Up

Use a ramp
Animating Up

Use a ramp
Animating Up

Use a ramp

Inverse kinematics
Animating Up

Use a ramp

Inverse kinematics
Animating Up

Use a ramp

Inverse kinematics

Displace Animation
Geometry

(0, 0, 0)

Node
Geometry

Node

(1, 0, 0)
Displace Animation

Geometry

Node

(0, 0, 0)
Displace Animation

Node (0, 0, 0)

Geometry
Displace Animation

Geometry

(1, 0, 0)

Node
Displace Animation

SCNTransaction

Geometry

(1, 0, 0) Node
// Synchronize animation completion and node position.

SCNTransaction.begin()
SCNTransaction.animationDuration = 0.0

SCNTransaction.commit()
// Synchronize animation completion and node position.

SCNTransaction.begin()
SCNTransaction.animationDuration = 0.0

actor.position = newPosition

SCNTransaction.commit()
// Synchronize animation completion and node position.

SCNTransaction.begin()
SCNTransaction.animationDuration = 0.0

actor.position = newPosition

// Remove all animations.
for key in node.animationKeys {
    node.removeAnimation(forKey: key)
}

SCNTransaction.commit()
Dynamic Scenery
isValid

/*
Access: ReadWrite
Stages: Vertex shader only
*/

struct SCNShaderGeometry {
  vec4 position;
  vec3 normal;
  vec4 tangent;
  vec4 color;
  vec2 texcoords[kSCNTexcoordCount];
} _geometry;

// modifier
_geometry.texcoords[0].y -= u_time*0.5;
VoiceOver Support
VoiceOver Support

Great non-visual experience
VoiceOver Support

Great non-visual experience

Enable all interactions

Support custom actions

Describe important elements
VoiceOver Support

Great non-visual experience
Enable all interactions
Support custom actions
Describe important elements
// Making accessible coordinates

class CoordinateAccessibilityElement: UIAccessibilityElement {

}
// Making accessible coordinates

class CoordinateAccessibilityElement: UIAccessibilityElement {
    var world: GridWorld
    var coordinate: Coordinate

    override var accessibilityLabel: String? {
        get {
            return world.speakableContents(of: coordinate)
        }
        ...
    }
}

// Setup accessibility elements.

// 1. Create accessibility element.
let accessibilityElement = CoordinateAccessibilityElement(coordinate: coordinate)

// 2. Find coordinates in view’s coordinate space.
let upperLeft = scnView.projectPoint(SCNVector3(...))
accessibilityElement.frame = ...

// 3. Add element to the view.
view.accessibilityElements!.append(accessibilityElement)
// Setup accessibility elements.

// 1. Create accessibility element.
let accessibilityElement = CoordinateAccessibilityElement(coordinate: coordinate)

// 2. Find coordinates in view’s coordinate space.
let upperLeft = scnView.projectPoint(SCNVector3(...))
accessibilityElement.frame = ...

// 3. Add element to the view.
view.accessibilityElements!.append(accessibilityElement)
// Setup accessibility elements.

// 1. Create accessibility element.
let accessibilityElement = CoordinateAccessibilityElement(coordinate: coordinate)

// 2. Find coordinates in view’s coordinate space.
let upperLeft = scnView.projectPoint(SCNVector3(...))
accessibilityElement.frame = ...

// 3. Add element to the view.
view.accessibilityElements!.append(accessibilityElement)
// Setup accessibility elements.

// 1. Create accessibility element.
let accessibilityElement = CoordinateAccessibilityElement(coordinate: coordinate)

// 2. Find coordinates in view’s coordinate space.
let upperLeft = scnView.projectPoint(SCNVector3(...))
accessibilityElement.frame = ...

// 3. Add element to the view.
view.accessibilityElements!.append(accessibilityElement)
// Setup accessibility elements.

// 1. Create accessibility element.
let accessibilityElement = CoordinateAccessibilityElement(coordinate: coordinate)

// 2. Find coordinates in view's coordinate space.
let upperLeft = scnView.projectPoint(SCNVector3(...))
accessibilityElement.frame = ...

// 3. Add element to the view.
view.accessibilityElements!.append(accessibilityElement)
Iterating
Iterating

Separate data and visuals
Iterating

Separate data and visuals

Value flexibility
Iterating

Separate data and visuals

Value flexibility

Audit accessibility support
Tuning
gameView.showsStatistics = true
gameView.showsStatistics = true
Rendering Statistics

gameView.showsStatistics = true
gameView.showsStatistics = true
Reduce Draw Call Count
Reducing Draw Calls

Geometry
Materials
Lighting
1 draw call per mesh
Flattening Geometry

Geometry A

Geometry B

Grass Material

Grass Material
Flattening Geometry

Flattened Geo.

Grass Material
Flattening Geometry

grassNode.flattenedClone()

Grass Material
Flattening Geometry

Tips

Store nodes to be flattened in a parent node

Save flattened scene
Caveat

Large environments
Reducing Draw Calls

Geometry
Materials
Lighting
Reducing Materials

Geometry A

Sand Material

Geometry B

Grass Material
Reducing Materials

Texture Atlas

Flattened Geo.
1 Material
Reducing Draw Calls

Geometry
Materials
Lighting
# Lights

<table>
<thead>
<tr>
<th>Type</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spotlight</td>
<td>1</td>
</tr>
<tr>
<td>Omni-Directional</td>
<td>4</td>
</tr>
<tr>
<td>Ambient</td>
<td>1</td>
</tr>
</tbody>
</table>
# Lights

## Baked lighting

<table>
<thead>
<tr>
<th>Light Type</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spotlight</td>
<td>1</td>
</tr>
<tr>
<td>Omni-Directional</td>
<td>4</td>
</tr>
<tr>
<td>Ambient</td>
<td>1</td>
</tr>
</tbody>
</table>
Lights
Baked lighting
Optimizing Performance

Flatten geometry

Use a texture atlas

Use light maps
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancements in SceneKit</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>Advanced in SceneKit Rendering</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>SceneKit: What's New</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Creating Content for Swift Playgrounds Lab</td>
<td>Technology Lab E</td>
</tr>
</tbody>
</table>