Scroll View School

Hands-On Challenges

Scroll View School
Hands-On Challenges

Challenge G: Gesture Recognizers

So far, you can pan and zoom around your map, but what if you wanted to actually make something happen in your game in response to touches?

In this challenge, you'll get some practice with this by adding two types of gesture recognizers: a tap gesture recognizer, and a pan gesture recognizer.

Part 1: Tap Gesture Recognizer

In this part of the challenge, let's make it so that if you tap a treasure chest, a sound effect plays and the treasure chest disappears.

To do this, you'll need to add a tap gesture recognizer to the view controller. Add these lines to `viewWillLayoutSubviews`, right after setting the scene's `vc` property:

```
UITapGestureRecognizer *tapRecognizer =
  [[UITapGestureRecognizer alloc] initWithTarget:self.scene
    action:@selector(viewTapped:)];
[self.scrollView addGestureRecognizer:tapRecognizer];
```

Here you're passing the gesture recognizer callback straight to the scene for simplicity. So declare the method in **MyScene.h**:

```
- (void)viewTapped:(UITapGestureRecognizer *)recognizer;
```

Switch to **MyScene.m** and implement it as follows:

```
- (void)viewTapped:(UITapGestureRecognizer *)recognizer {
  // 1
  CGPoint tapLocation = [recognizer locationInView:self.view];
  tapLocation = [self convertPointFromView:tapLocation];
  // 2
  CGPoint mapLocation = [self.cave convertPoint:tapLocation
    fromNode:self];
  // 3
  SKNode *node = [self.cave nodeAtPoint:mapLocation];
  // 4
  if (node) {
    // 5
    if ([node.name isEqualToString:@"TREASURE"]) {
      // 6
      [node removeFromParent];
      [self runAction:[SKAction playSoundFileNamed:@"treasure.wav"

        waitForCompletion:NO]];
    }
  }
}
```

Let’s review this section by section:

1. Get the location inside the view for the tap. These are in UIKit coordinates, so you have to convert them to Sprite Kit coordinates with the `convertPointFromView:` method built-in to SKScene.
2. Next convert the point from scene to map coordinates with `convertPoint:fromNode:`.
3. The Cave class has a method that returns you the node at a certain point.
4. Make sure a node is returned.
5. Check to see if it’s a treasure.
6. If it’s a treasure, remove it from the scene, and play a sound effect.

And that’s it! Build and run, and now you should be able to scroll around the map and tap treasure chests to remove them.

Uber Haxx0r Challenge: Pan Gesture Recognizer

Your next challenge is to modify the game so that panning works as usual, except when you pan starting on a treasure chest. In that case, you should be able to drag the treasure chest to a new position.

Here are a few hints for how to accomplish this:

- Add a pan gesture recognizer to the scroll view, just like you did for the tap gesture recognizer. Send the callback to the scene.
- This time, set the delegate of the pan gesture recognizer to the scene.
- Inside the scene, implement `gestureRecognizer:shouldReceiveTouch:`. It should return NO unless the user is dragging on top of a treasure node. In this case, you

should remove it from its parent node, and add it as a direct node of the cave instead. Also save a copy of this node for future reference.

- In your callback for the pan geseture recognizer, check the state of the gesture recognizer. On begin, zoom up the treasure. On changed, move the treasure to the new position. One end, zoom back the treasure. Also remove it from its parent, and add it as a child of whatever node is at the current point.

If you get this working, congratulations – you now have the full power of scroll views, gesture recognizers, and Sprite Kit at your disposal! :]