# GD2S03
# Advanced Software Engineering & Programming for Games


## Bachelor of Software Engineering(BSE)
## Game Development

- Overview
  - Handling Gestures
  - SKTileMapNode
  - SKTileSet
  - SKTileGroup
  - SKTileDefinition

# Input Controls - UIGestureRecognizer

- UIGestureRecognizer is the base class for concrete gesture recognizers.
- A gesture-recognizer object recognizes a sequence of touches and act on that recognition.
- When a gesture-recognizer recognizes a common gesture or change in gesture, it sends an action to each designated target object.
- The concrete subclass of UIGestureRecognizer are the following –
  - UITapGestureRecognizer
  - UIPinchGestureRecognizer
  - UIRotationGestureRecognizer
  - UISwipeGestureRecognizer
  - UIPanGestureRecognizer
  - UIScreenEdgePanGestureRecognizer
  - UILongPressGestureRecognizer
- A gesture recognizer operates on specific view and all of that view's subviews. It thus must be associated with that subview.
- To make that association UIView's method addGestureRecognizer(_:) must be called.

# Input Controls - UIGestureRecognizer

- Steps to Create a Gesture
- ***Initialize a gesture*** and pass the name and location of the function which will handle the gesture as parameter. For e.g.
  - var longPressGesture = UILongPressGestureRecognizer()
    or
  - var longPressGesture = UILongPressGestureRecognizer(target: self, action: #selector(longPressHandler)

- ***Set other attributes* of the gesture** for e.g.
  - longPressGesture.minimumPressDuration = 1

- ***Add the gesture to the view***
  - view.addGestureRecognizer(longPressGesture)

- ***Define Gesture Handler***
  - @objc func longPressHandler(sender: UILongPressGestureRecognizer){
    //Handles what happens at long press
  - }

# Input Controls - Example

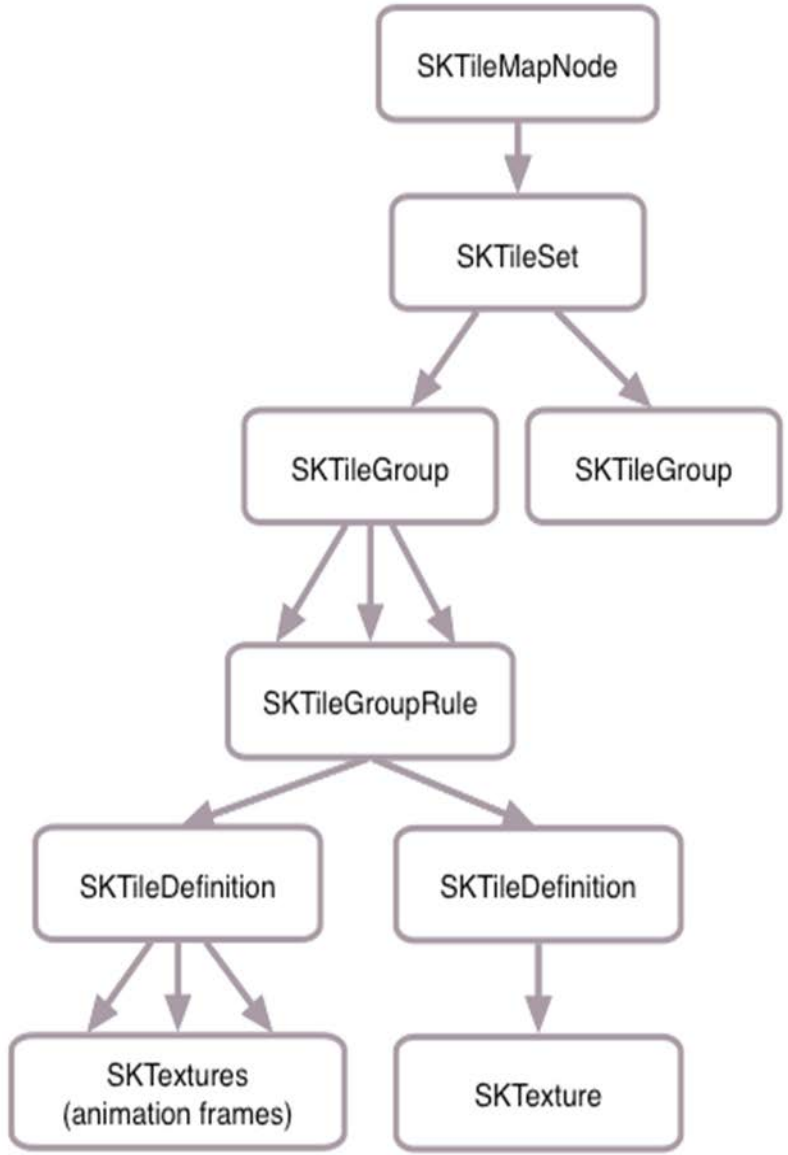
```
import SpriteKit

class GameScene: SKScene{
    private let node = SKSpriteNode()
    var longPressGestureRecognizer = UILongPressGestureRecognizer()
    var tapGestureRecognizer = UITapGestureRecognizer()
    var panGestureRecognizer = UIPanGestureRecognizer()
    var nodePosition = CGPoint()

    override func didMove(to view: SKView) {
        createNode()
        setupGestureRecognizer()
    }

    func createNode(){
        node.size = CGSize(width: 64, height: 64)
        node.color = UIColor.blue
        node.position = CGPoint(x: self.frame.width/2, y: self.frame.height/2)
        nodePosition = node.position
        addChild(node)
    }

    func setupLongPressGesture(){
        guard let view = view else { return}
        longPressGestureRecognizer = UILongPressGestureRecognizer(target: self, action: #selector(longPress))
        longPressGestureRecognizer.minimumPressDuration = 1
        view.addGestureRecognizer(longPressGestureRecognizer)
    }

    func setupTapGesture(){
        guard let view = view else { return}
        tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(tap))
        tapGestureRecognizer.numberOfTapsRequired = 2
        view.addGestureRecognizer(tapGestureRecognizer)
    }

    func setupPanGesture(){
        guard let view = view else { return}
        panGestureRecognizer = UIPanGestureRecognizer(target: self, action: #selector(pan))
        panGestureRecognizer.maximumNumberOfTouches = 1
        view.addGestureRecognizer(panGestureRecognizer)
    }

    func setupGestureRecognizer(){
        setupLongPressGesture()
        setupTapGesture()
        setupPanGesture()
    }
```

# Input Controls - UIGestureRecognizer

```
    @objc func pan(sender: UIPanGestureRecognizer){
        let translate = sender.translation(in: self.view)
        var tapLocation = sender.location(in: self.view)
        //map tapLocation
        tapLocation.y = abs(self.frame.height - tapLocation.y)
        if(node.contains(tapLocation)){
            node.position = CGPoint(x: node.position.x + translate.x, y: node.position.y - translate.y)
        }
        sender.setTranslation(CGPoint.zero, in: self.view)
    }

    @objc func tap(sender: UITapGestureRecognizer){
        var tapLocation = sender.location(in: self.view)
        //map tapLocation
        tapLocation.y = abs(self.frame.height - tapLocation.y)
        if node.contains(tapLocation){
            if node.color == UIColor.blue{
                node.color = UIColor.yellow
            }
            else if node.color == UIColor.yellow{
                node.color = UIColor.blue
            }
         }
    }

    @objc func longPress(sender: UILongPressGestureRecognizer){
        var tapLocation = sender.location(in: self.view)
        //map tapLocation
        tapLocation.y = abs(self.frame.height - tapLocation.y)
        if(sender.state == .began){
            if(node.contains(tapLocation)){
                node.size = CGSize(width: node.size.width * 2, height: node.size.height * 2)
            }
        }
        if(sender.state == .ended){
            node.size = CGSize(width: 64, height: 64)
        }

    }

}
```

# SKTileMapNode, SKTileSet, SKTileGroup,SKTileDefinition

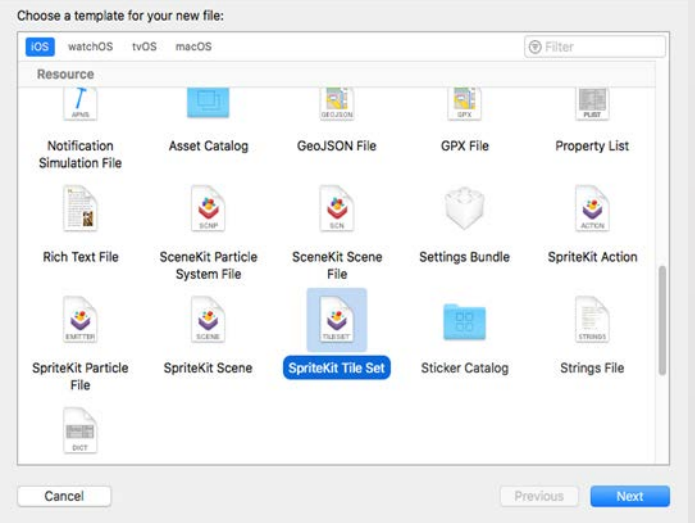
- SKTileMapNode is used to render a 2D array of textured sprites.
- SKTileMapnode must be supplied with tile sets (SKTileSet)
- SKTileset object contains an array of tile groups (SKTileGroup)
- SKTileGroup contains either-
  - The definition(SKTileDefinition) of a single tile or
  - An array of one or more SKTileGroupRule that allows for automatic placement of textures dependent on adjacency and the placement weights of their definition.
  - SKTileGroupRule defines how tile should be placed in a map
- SKTileDefinition describes a single type of tile that is used within the map

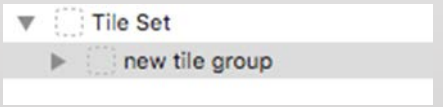


# Creating tile map - Programmatically

```
import SpriteKit

class GameScene: SKScene {

    override func didMove(to view: SKView) {
        initialiseTileMapNode()
    }

    func initialiseTileMapNode(){
        let columns = 4
        let rows = 4
        let size = CGSize(width: 64, height: 64)

        let tileTexture = SKTexture(imageNamed: "surprised.png")
        let tileDefinition = SKTileDefinition(texture: tileTexture, size: size)
        let tileGroup = SKTileGroup(tileDefinition: tileDefinition)

        let tileSet = SKTileSet(tileGroups: [tileGroup])
        let tileMapNode = SKTileMapNode(tileSet: tileSet, columns: columns, rows: rows, tileSize: size)
        tileMapNode.position = CGPoint(x: self.frame.width/2 , y: self.frame.height/2)
        let tile = tileMapNode.tileSet.tileGroups.first!
        tileMapNode.fill(with: tile)
        self.addChild(tileMapNode)
    }

}
```

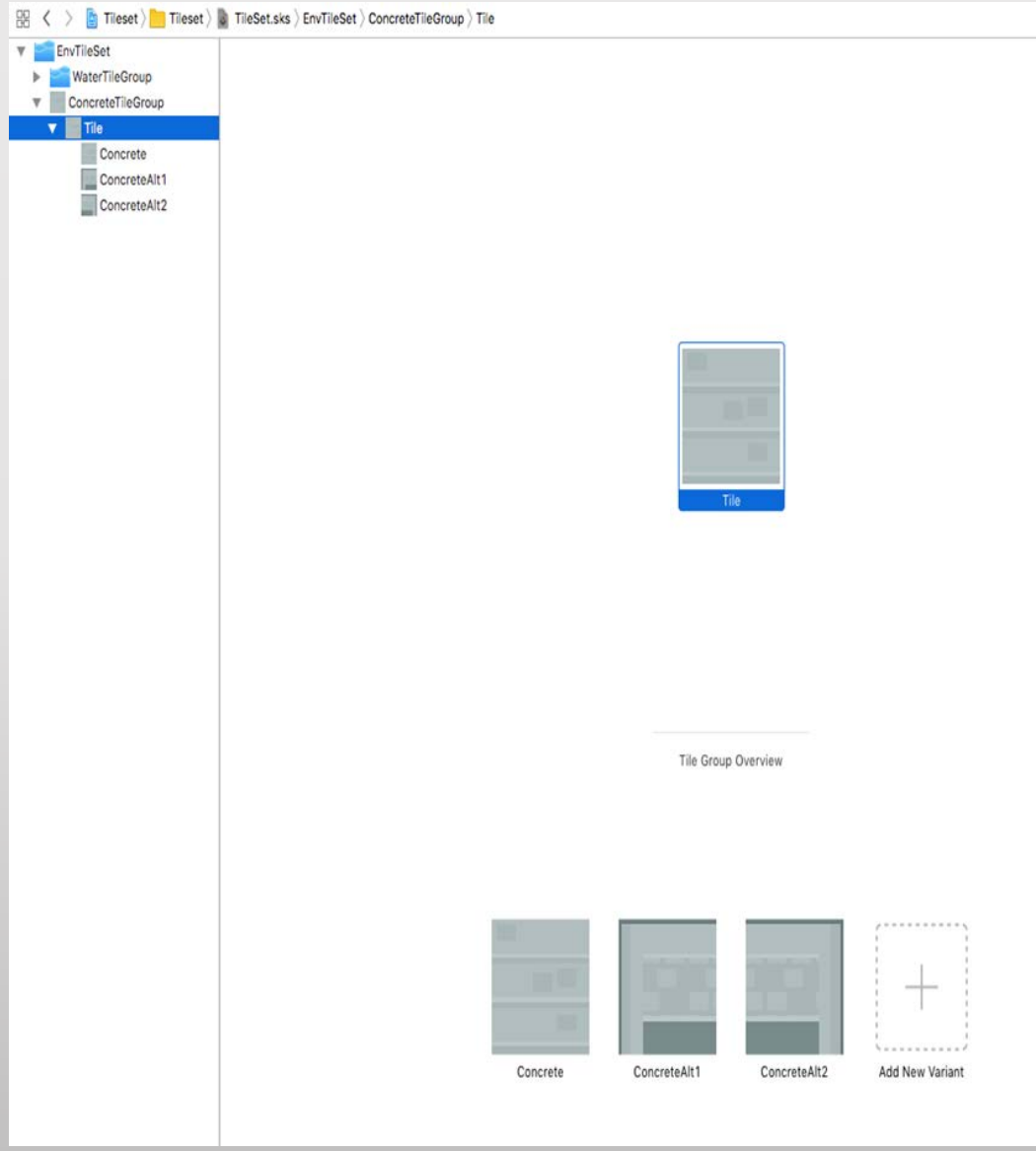
- Add or Drag and drop file “surprised.png” to “Assets.xcassets”
- Choose File->New - > File, scroll down and choose ios->SpriteKit Tile Set template under **Resource** and click Next
- From **Tile Set Template** drop down select **Empty Tile Set**. Press Next
- Name file as **TileSet** and press **Create**.
- Select **TileSet.sks** in the project navigator.
- A **Tile Set** and a **new tile group** has automatically been created.
- Rename **Tile Set** to **EnvTileSet**
- Rename **new tile group** to **WaterTileGroup**
- Locate **water.png** from Media Library and drag it onto the empty tile area
- Expand **WaterTileGroup** and select **Tile->water**
- Set tile definitions. for e.g. set size => width = 64, height = 64



# Creating tile map – Using Editor

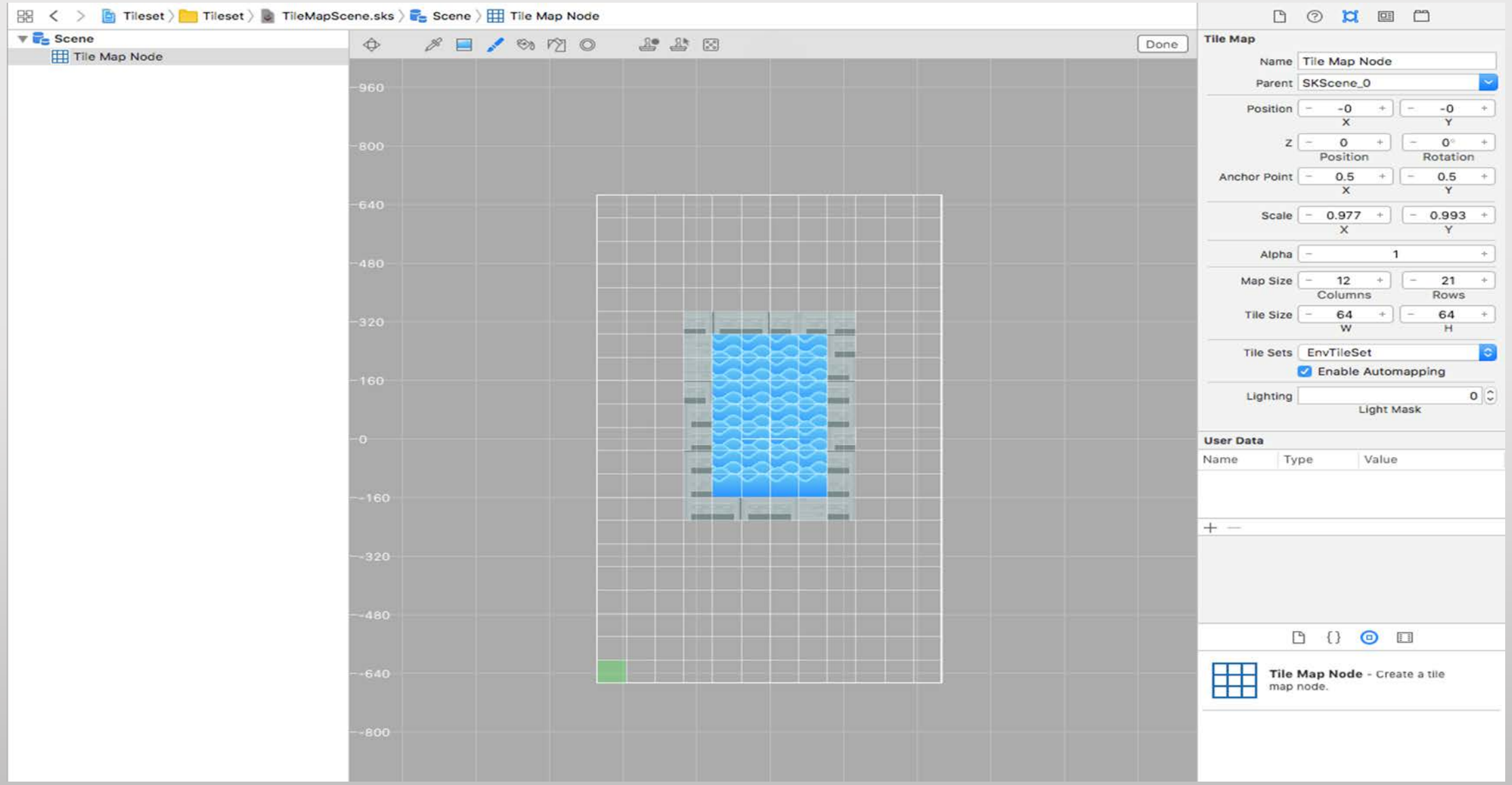
- **Working with Tile Variants**
  - Add or Drag and drop folder **Tiles** to the
  - **Assets.xcassets**
  - Select **TileSet.sks** in the project navigator.
  - Right click **EnvTileSet**. Select **New->Single Tile Group**
  - Select **New Tile Group** and in attribute selector and rename it to **ConcreteTileGroup**.
  - Expand **ConcreteTileGroup** and select **Tile**
  - Locate **concrete.png** from Media Library and drag it onto the empty tile area.
  - Select **Tile** and again from media library drag **ConcreteAlt1** to the **Add New Variant**.
  - Similarly add **ConcreteAlt2** to **Add New Variant**
  - For each tile, set the size with width = 64, height = 64 and set placement weight to 1, 2, 3 respectively



# Creating tile map – Using Editor

- Choose **File->New File**. In **ios** scroll down and from **Resources** choose **SpriteKit Scene** template
- Click **Next** and save it as **TileMapScene** and press **Create.**
- Select **TileMapScene.sks** in Project folder to open the editor.
- From object library select **Tile Map Node** and drag it to the center of the file **TileMapScene.sks**
- For Tile Map Node, in Attribute Inspector, set the following
  - Anchor Point-> x=0.5, y=0.5
  - Map Size - > columns = 12 and rows = 21
  - Tile size-> width = 64 and height = 64
  - Tile sets -> Env
  - Eanble Automapping - > check to select (helps to select whole tile group in place of individual tiles)
- If required adjust the size of the Tile Map Node to the size of the screen
- Double Click on **Tile Map Node** to highlight the rows and columns.
- In map editor click **select tile** and select **water** group. Fill the center of map node with this tile.
- Again click **select tile** to select **Block** and draw boundary around water with this tile.
- Notice the occurrence of blocks. The chance of occurrence of tile with higher Placement Weight is more than the ones with lower Placement Weight

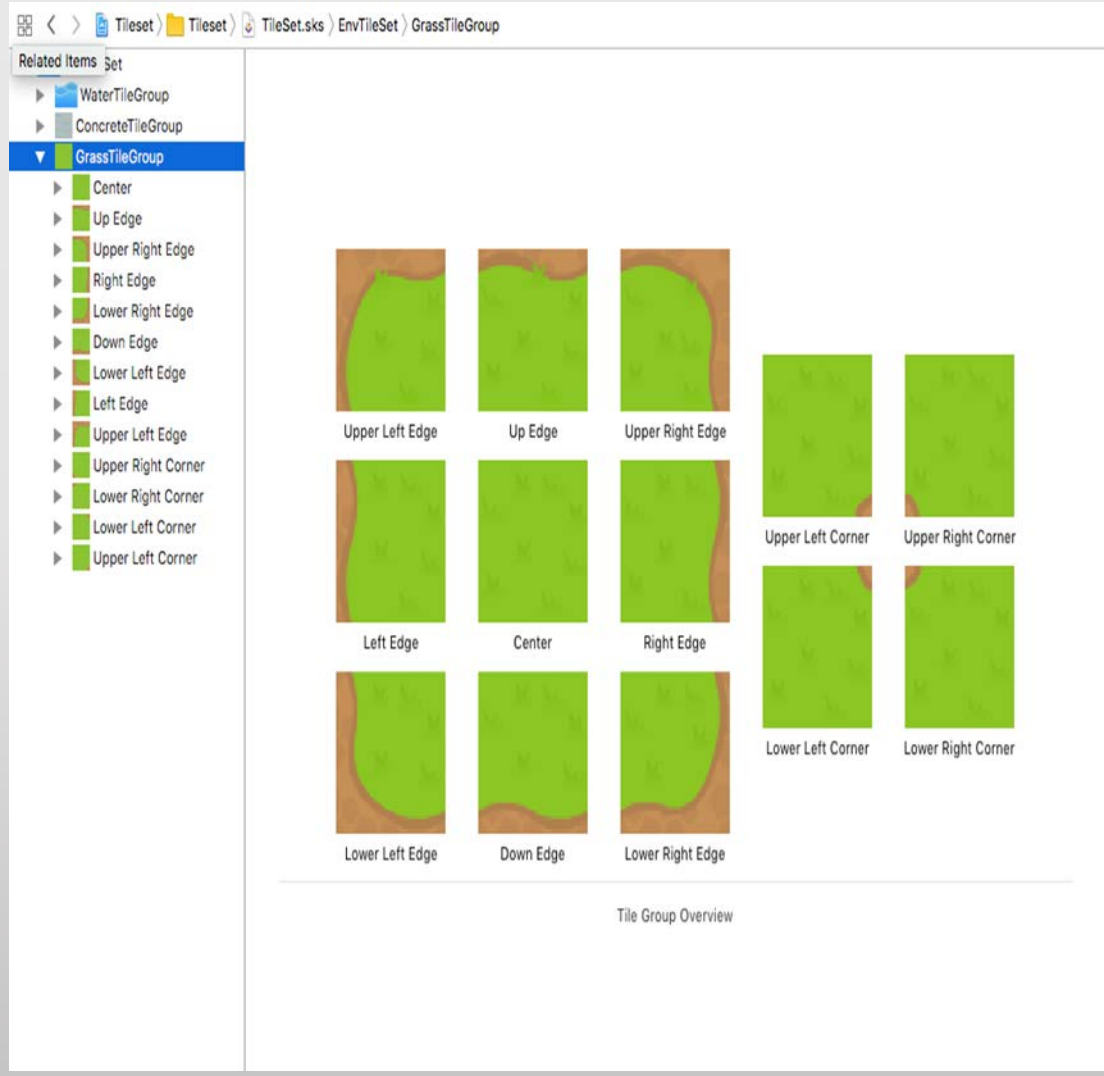
# Creating tile map – Using Editor



# Creating tile map – Using Editor

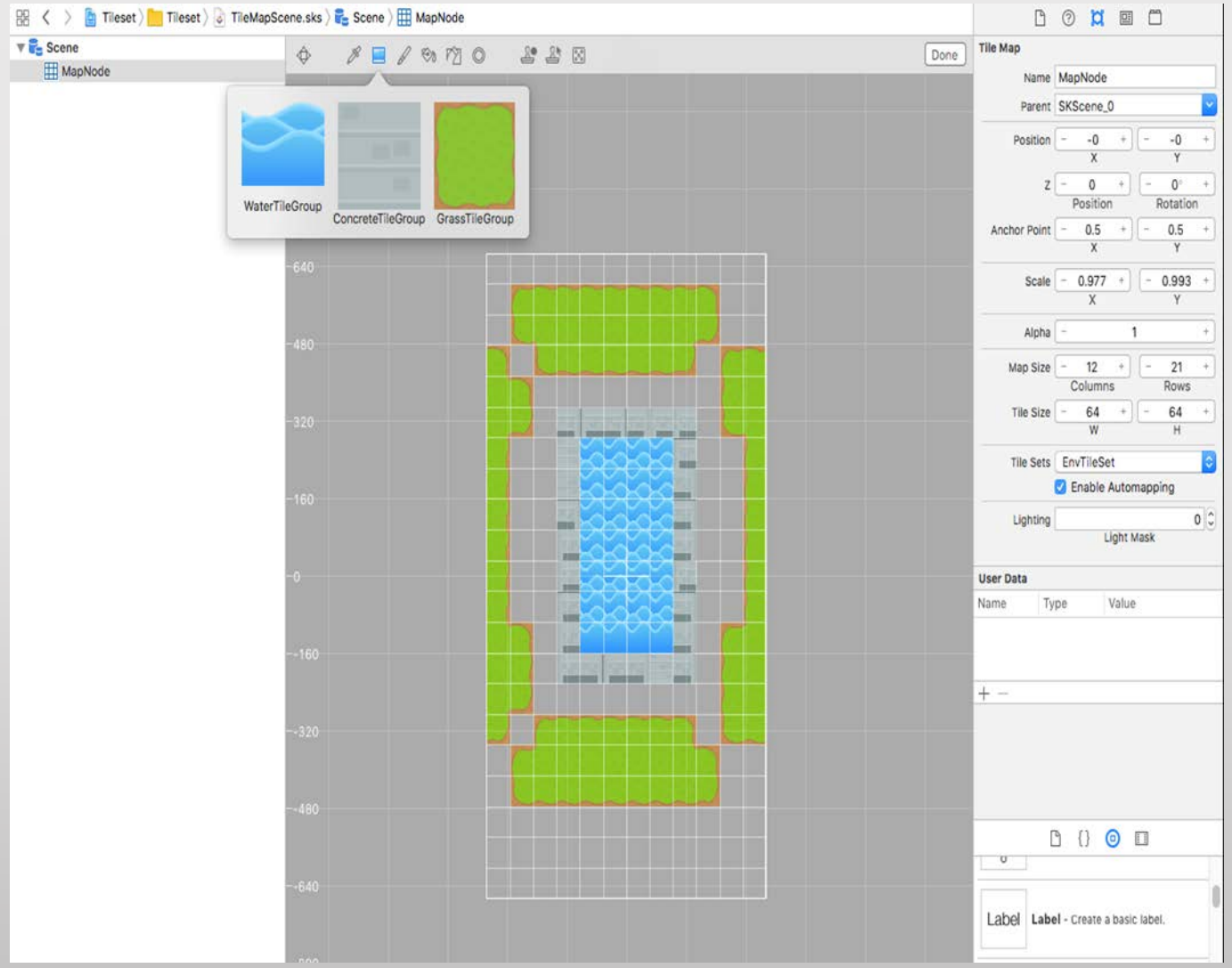
***SKTileGroupRule***

- Select **TileSet.sks** in the project navigator.
- Right click **EnvTileSet**.
- Select **New->8-Way Adjacency Group**
- Select **new tile group** and in attribute selector rename it to **GrassTileGroup**. Select **GrassTileGroup**
- Locate respective images from the Media Library and drag it onto the respective place.
- ULE = grass0 ULC = outside0
- UE = grass1 RE = grass5 URC = outside1
- URE = grass2 LLE = grass6 LLC = outside2
- LE = grass3 DE = grass7 LRC = outside3
- Center = grass4 LRE = grass8
- For each tile, set the size with width = 64, and height = 64



# Creating tile map – Using Editor

- Select **TileMapScene.sks**
- Make Sure **Enable Automapping** is selected
- Click **Select Tile** and select **Grass Tile Group**.
- Try creating tiles of grass on map.
- Uncheck **Enable Automapping** and try to see the difference.



# Attaching SpriteKitScene(sks) file to Custom Class

- Click to open the file **TileMapScene.sks** and select **Tile Map Node.**
- In attribute selector name **Tile Map Node** to **MapNode.**
- Again select **scene** and in **custom class inpector**, enter **GameScene** as custom class.
- Now select **GameScene.swift** from project folder and enter the code in the right.
- Execute to see the result

```
import SpriteKit
import GameplayKit

class GameScene: SKScene {
    var node: SKSpriteNode!

    override func didMove(to view: SKView) {
        node = SKSpriteNode(imageNamed: "surprised")
        node.size = CGSize(width: 32, height: 32)
        node.run(SKAction.repeatForever(
            SKAction.sequence([SKAction.scale(by: 2, duration: 0.5),
                               SKAction.scale(by: 0.5, duration: 0.5)])))
        let child  = self.childNode(withName: "MapNode") as! SKTileMapNode
            child.addChild(node)

    }
    override func update(_ currentTime: TimeInterval) {
        // Called before each frame is rendered
    }
}
```