Gems of Game play Kit

CocoaHeads Malmö – Tobias Due Munk – 7 September 2017

Tobias Due Munk

@tobiasdm

“Unlike high-level game engines such as SpriteKit and SceneKit, GameplayKit is not involved in animating and rendering visual content. Instead, you ...

...

use GameplayKit to develop your gameplay mechanics and to design modular, scalable game architecture with minimal effort.”

...

use GameplayKit to develop your gameplay mechanics and to design modular, scalable game architecture with minimal effort.”

Shuffle arrays

Naive

```
extension MutableCollection {
    mutating func shuffle() {
        guard count > 1 else {
            return
        }
        for (firstUnshuffled, unshuffledCount)
          in zip(indices, stride(from: count, to: 1, by: -1))
        {
            let d = Int(arc4random_uniform(Int(unshuffledCount)))
            let i = index(firstUnshuffled, offsetBy: d)
            swapAt(firstUnshuffled, i)
        }
    }
}
```

Stackoverflow

Naive

```
extension MutableCollection {
    mutating func shuffle() {
        guard count > 1 else {
            return
        }
        for (firstUnshuffled, unshuffledCount)
          in zip(indices, stride(from: count, to: 1, by: -1))
        {
            let d = Int(arc4random_uniform(Int(unshuffledCount)))
            let i = index(firstUnshuffled, offsetBy: d)
            swapAt(firstUnshuffled, i)
        }
    }
}
```

Naive

```
extension Sequence {
    func shuffled() -> [Element] {
        var result = Array(self)
        result.shuffle()
        return result
    }
}
```

Naive

```
[0, 1, 2, 3].shuffled()

// [2, 1, 0, 3]
```

Gem A

```
import GameplayKit
```

Gem A

```
import GameplayKit

GKARC4RandomSource
    .sharedRandom()
    .arrayByShufflingObjects(
        in: [0, 1, 2, 3]
    )
```

Gem A

```
import GameplayKit

extension Array {

    func shuffled(
      source: GKRandomSource = GKARC4RandomSource.sharedRandom()
    ) -> [Element] {
        let shuffled = source.arrayByShufflingObjects(in: self)
        return shuffled as! [Element]
    }
}
```

Gem A

```
import GameplayKit

extension Array {

    func shuffled(
      source: GKRandomSource = GKARC4RandomSource.sharedRandom()
    ) -> [Element] {
        let shuffled = source.arrayByShufflingObjects(in: self)
        return shuffled as! [Element]
    }
}
```

Gem A

```
[0, 1, 2, 3].shuffled()

// [2, 0, 1, 3]
```

Gem B

```
import GameplayKit

extension Array {

    func perceivedShuffled(
        source: GKRandomSource = GKARC4RandomSource.sharedRandom()
    ) -> [Element] {
        let distribution = GKShuffledDistribution(
            randomSource: source,
            lowestValue: 0,
            highestValue: count - 1
        )
        var shuffledArray: [Element] = []
        while shuffledArray.count < count {
            let index = distribution.nextInt()
            shuffledArray.append(self[index])
        }
        return shuffledArray
    }
}
```

Gem B

```
import GameplayKit

extension Array {

    func perceivedShuffled(
        source: GKRandomSource = GKARC4RandomSource.sharedRandom()
    ) -> [Element] {
        let distribution = GKShuffledDistribution(
            randomSource: source,
            lowestValue: 0,
            highestValue: count - 1
        )
        var shuffledArray: [Element] = []
        while shuffledArray.count < count {
            let index = distribution.nextInt()
            shuffledArray.append(self[index])
        }
        return shuffledArray
    }
}
```

Gem B

```
[0, 1, 2, 3].perceivedShuffled()

// [3, 1, 0, 2]
```

Original

Gem A

Gem B

0 1 2 3 4 5 6 7 8 9

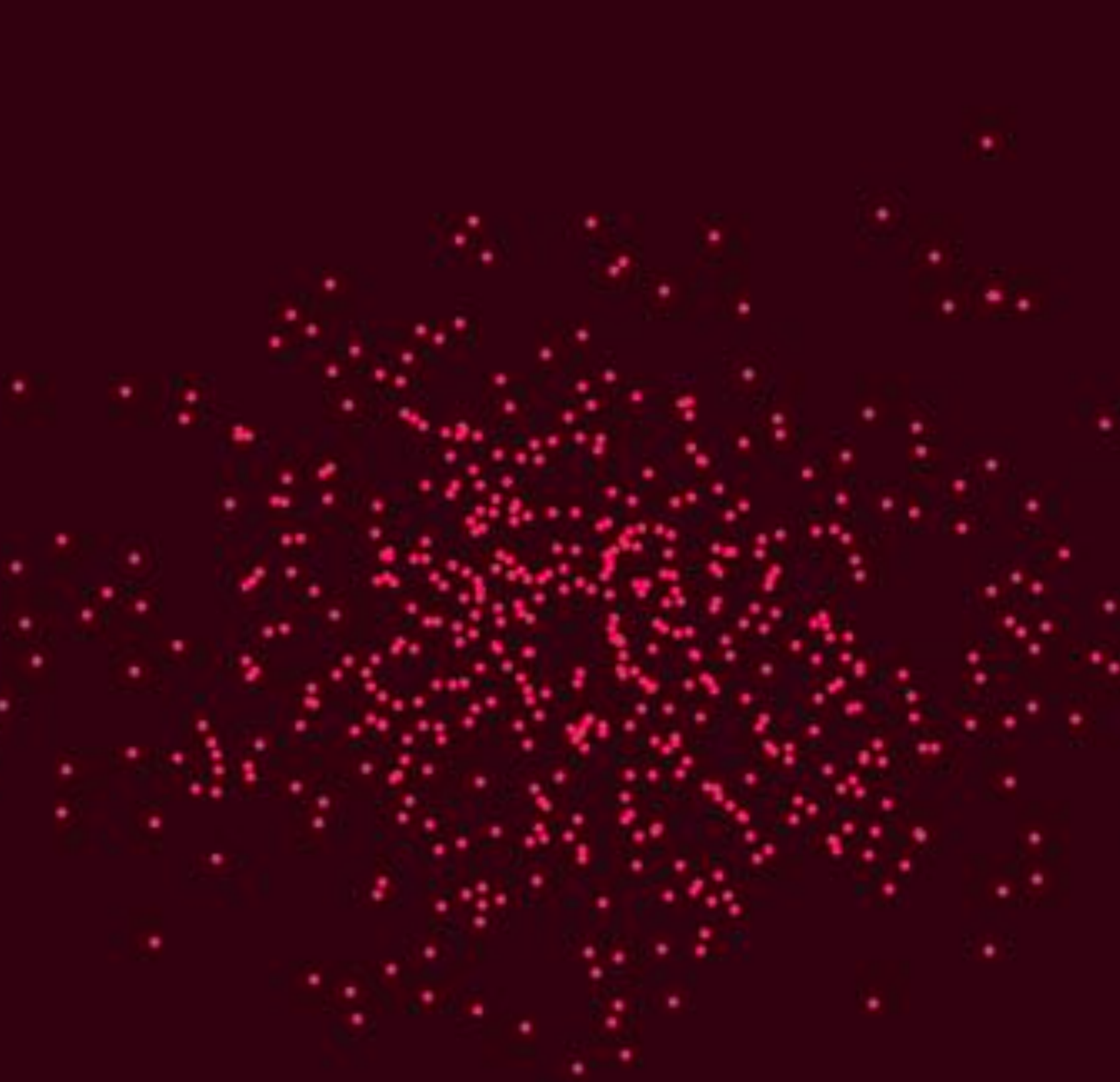
6 2 0 1 3 7 4 8 5 9

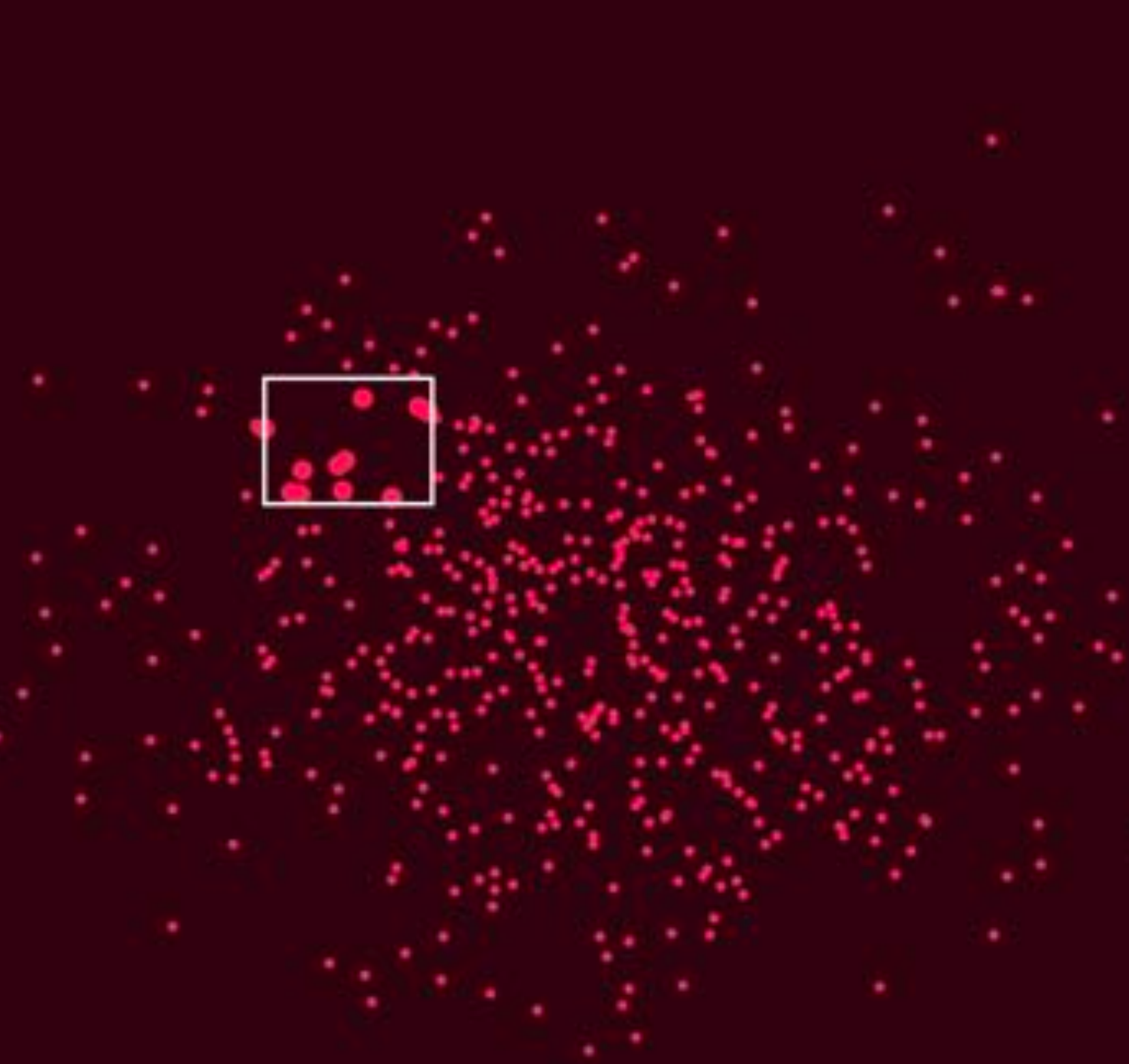
9 5 8 4 7 3 1 0 2 6

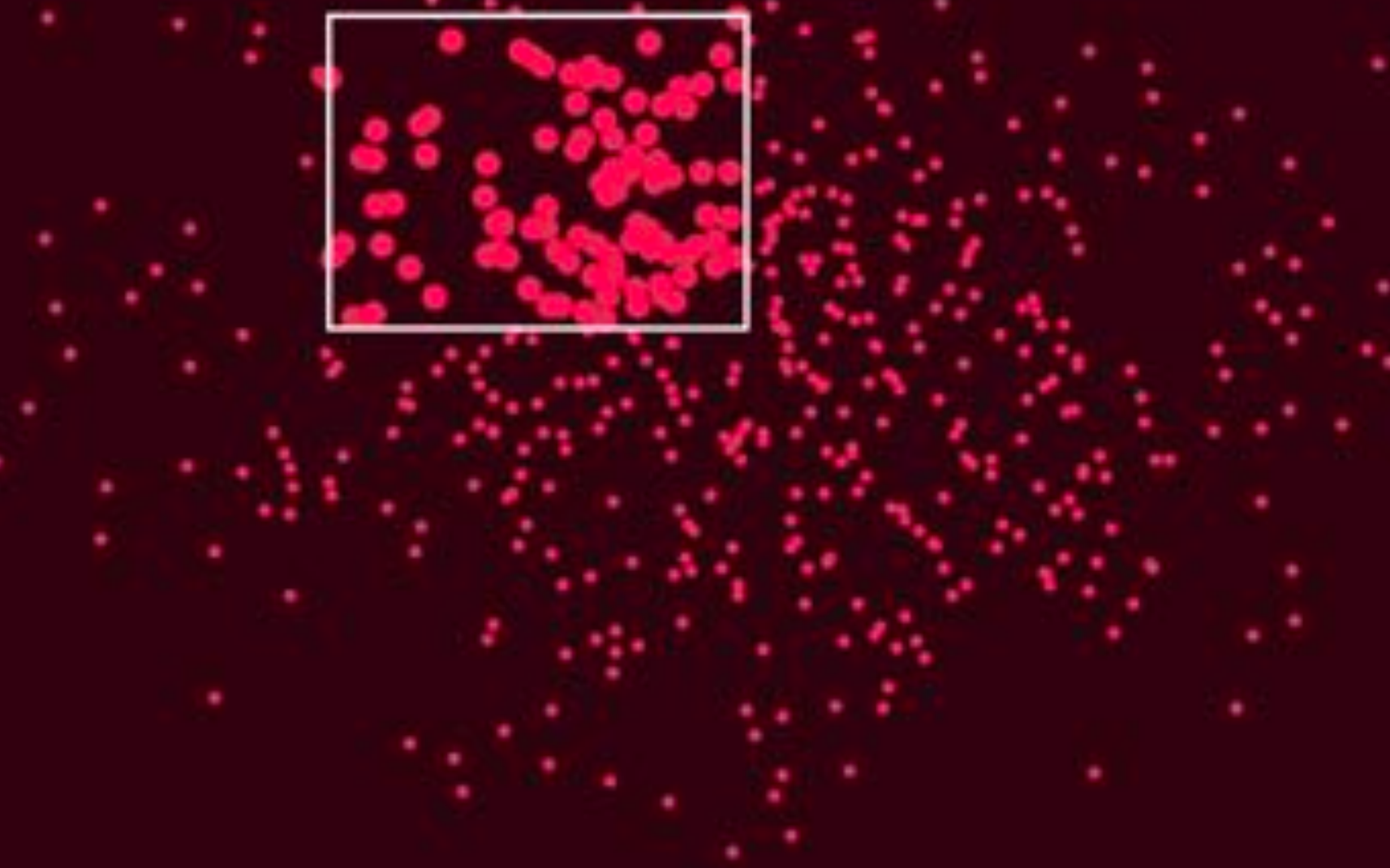
Performant

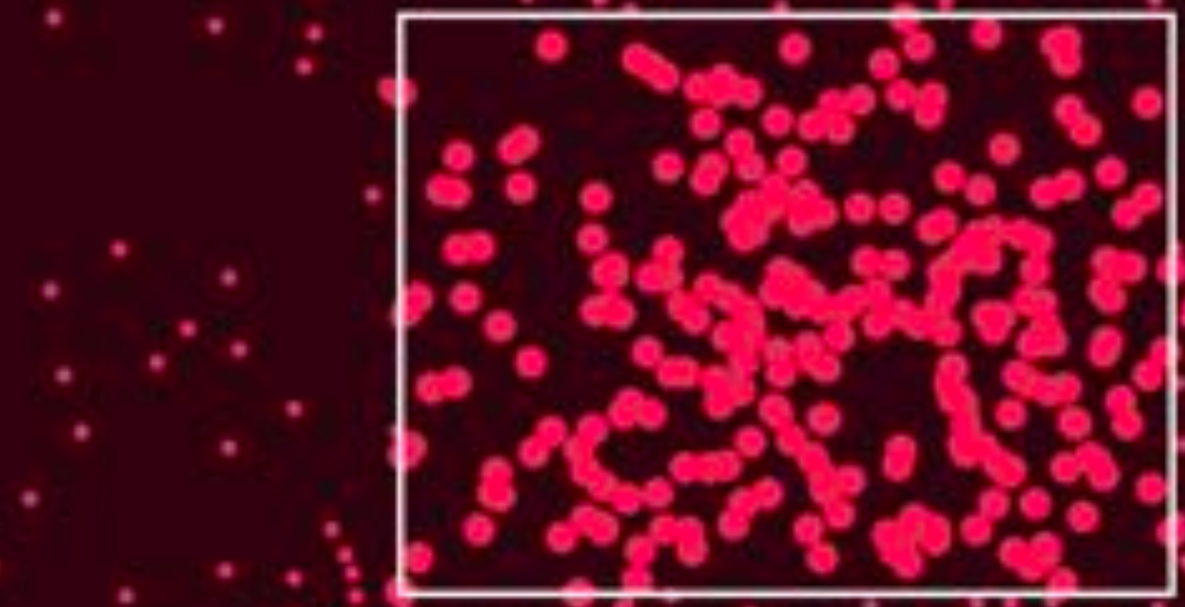
Visual

Search









Naive

```
let points = [
    CGPoint(x: 0, y: 0),
    CGPoint(x: 1, y: 1),
    CGPoint(x: 2, y: 0)
]

let rect = CGRect(
    origin: .zero,
    size: CGSize(width: 1, height: 1)
)
```

Naive

```
points.filter { point in
    return rect.contains(point)
}
```

Gem

```
import GameplayKit
```

Gem

```
import GameplayKit

class Point: NSObject {
    let x: CGFloat
    let y: CGFloat
}
```

Gem

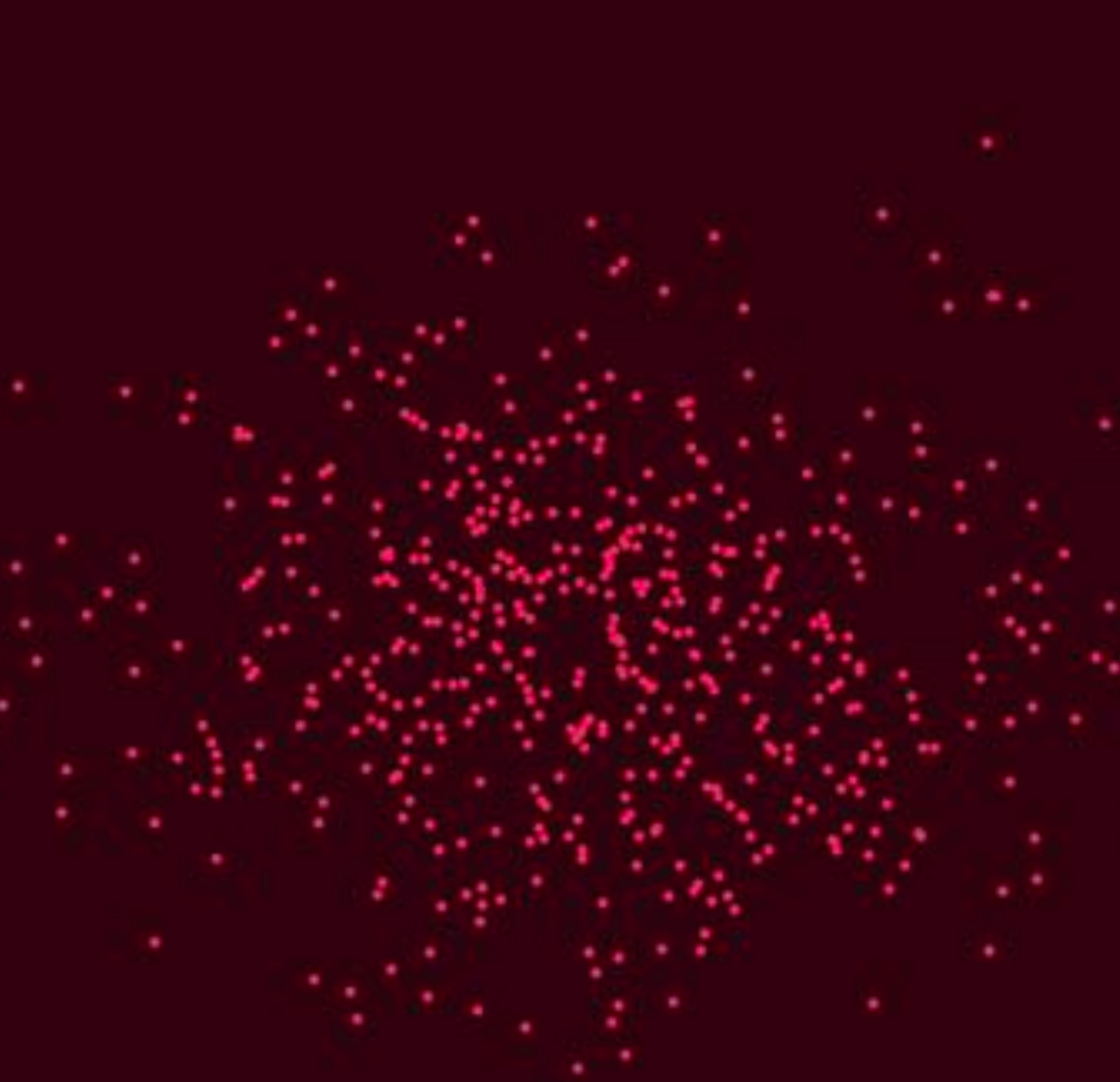
```
import GameplayKit

let tree = GKRTree<Point>(maxNumberOfChildren: 10)
```

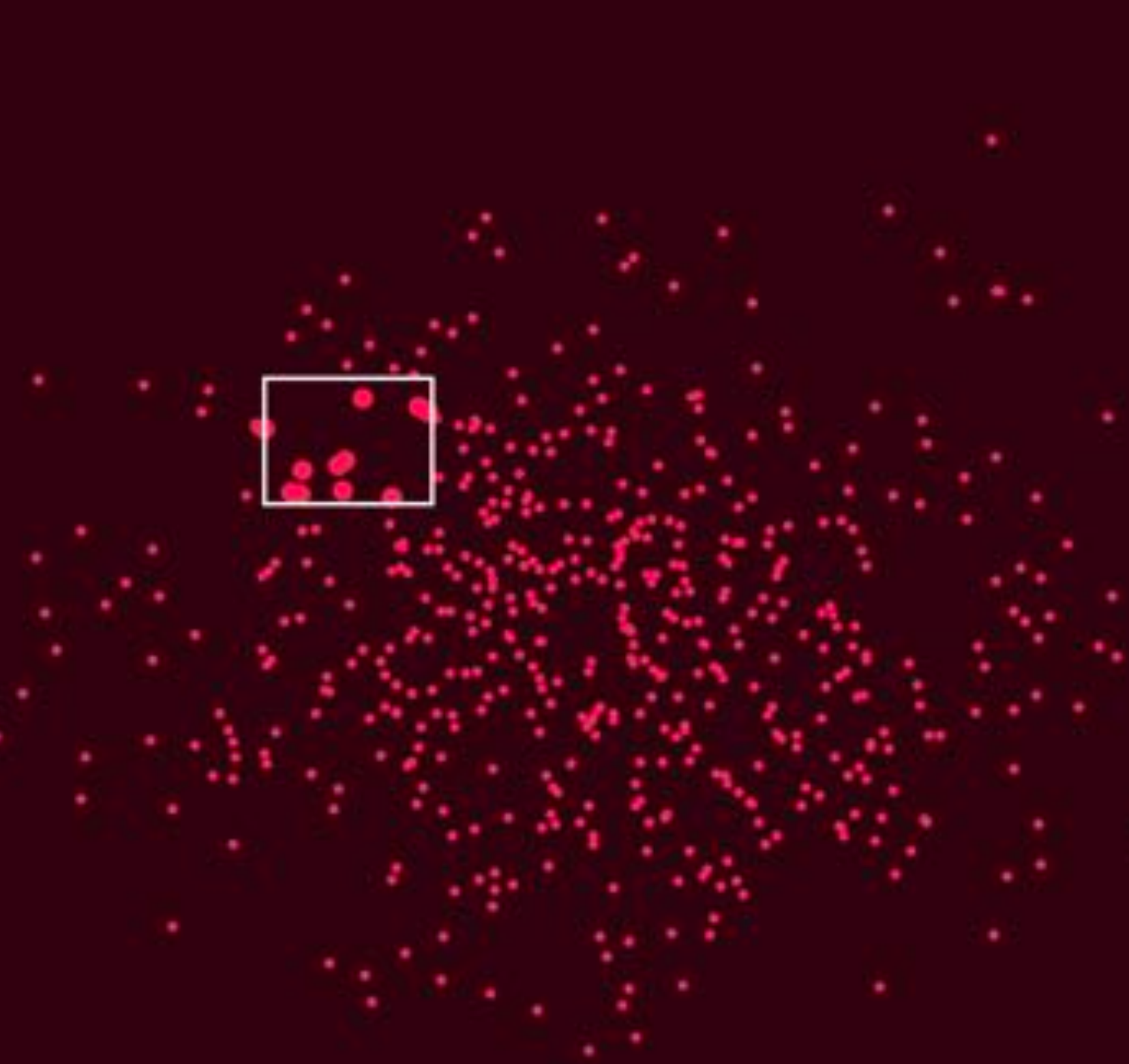
Gem

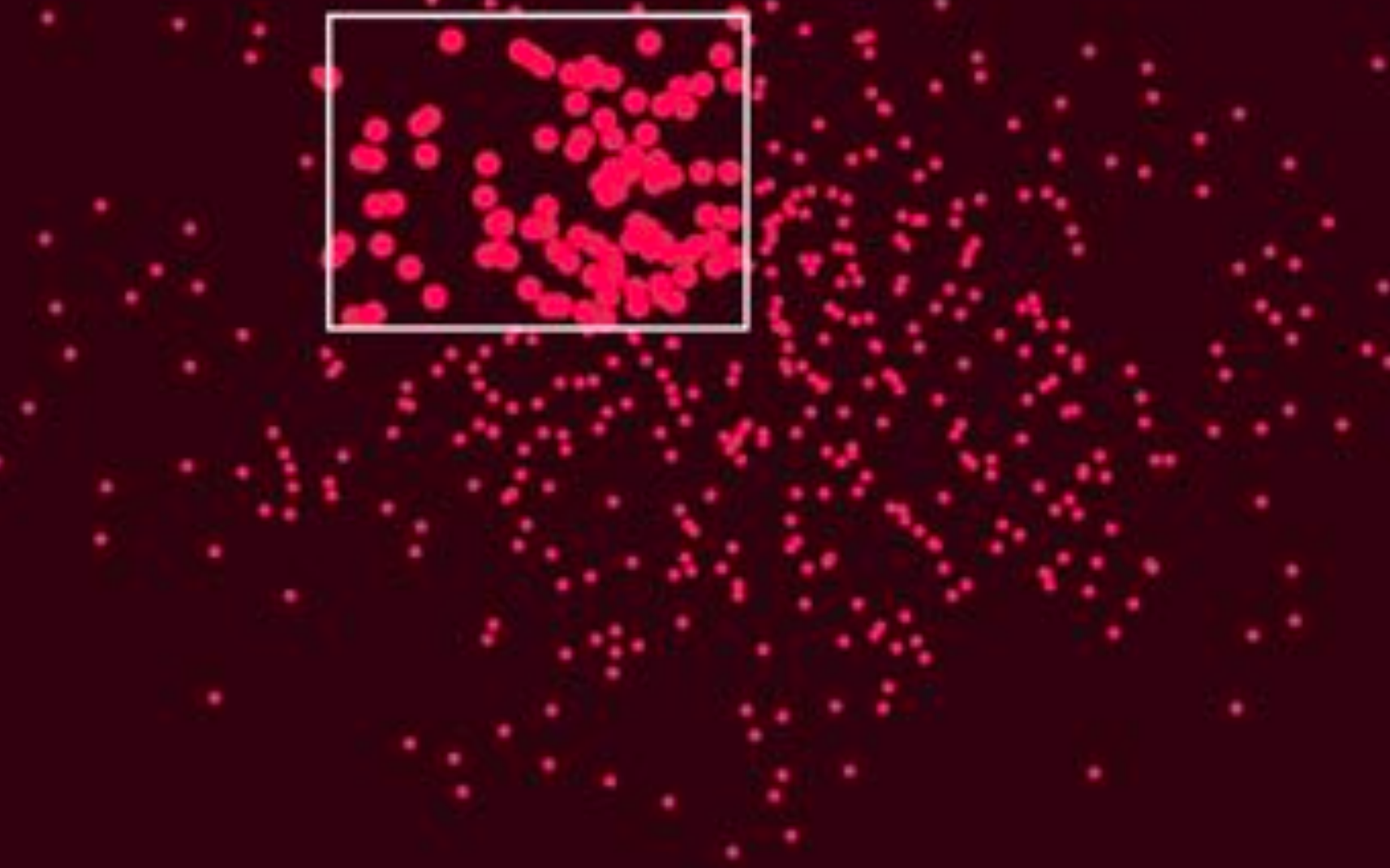
```
import GameplayKit

let tree = GKRTree<Point>(maxNumberOfChildren: 10)
for point in points {
    let vector = vector_float2(
        x: Float(point.x),
        y: Float(point.y)
    )
    tree.addElement(
        point,
        boundingRectMin: vector,
        boundingRectMax: vector,
        splitStrategy: .reduceOverlap
    )
}
```

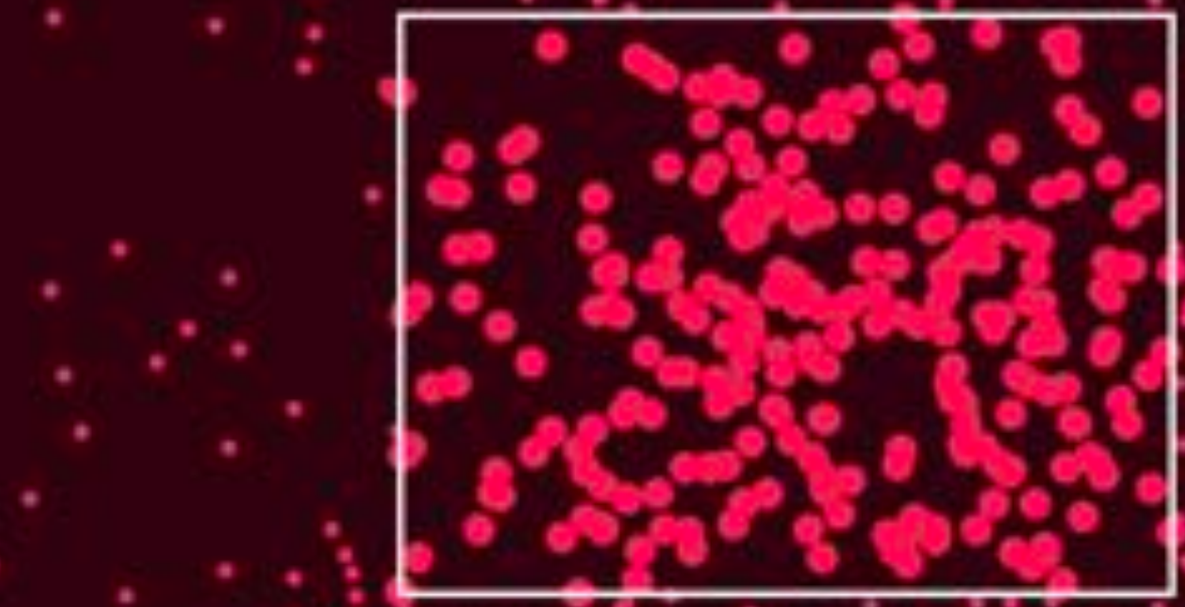
Gem

```
let rectMin = vector_float2(
    x: rect.minX,
    y: rect.minY
)
let rectMax = vector_float2(
    x: rect.maxX,
    y: rect.maxY
)
tree.elements(
    inBoundingRectMin: rectMin,
    rectMax: rectMax
)
```









Natural

Randomness

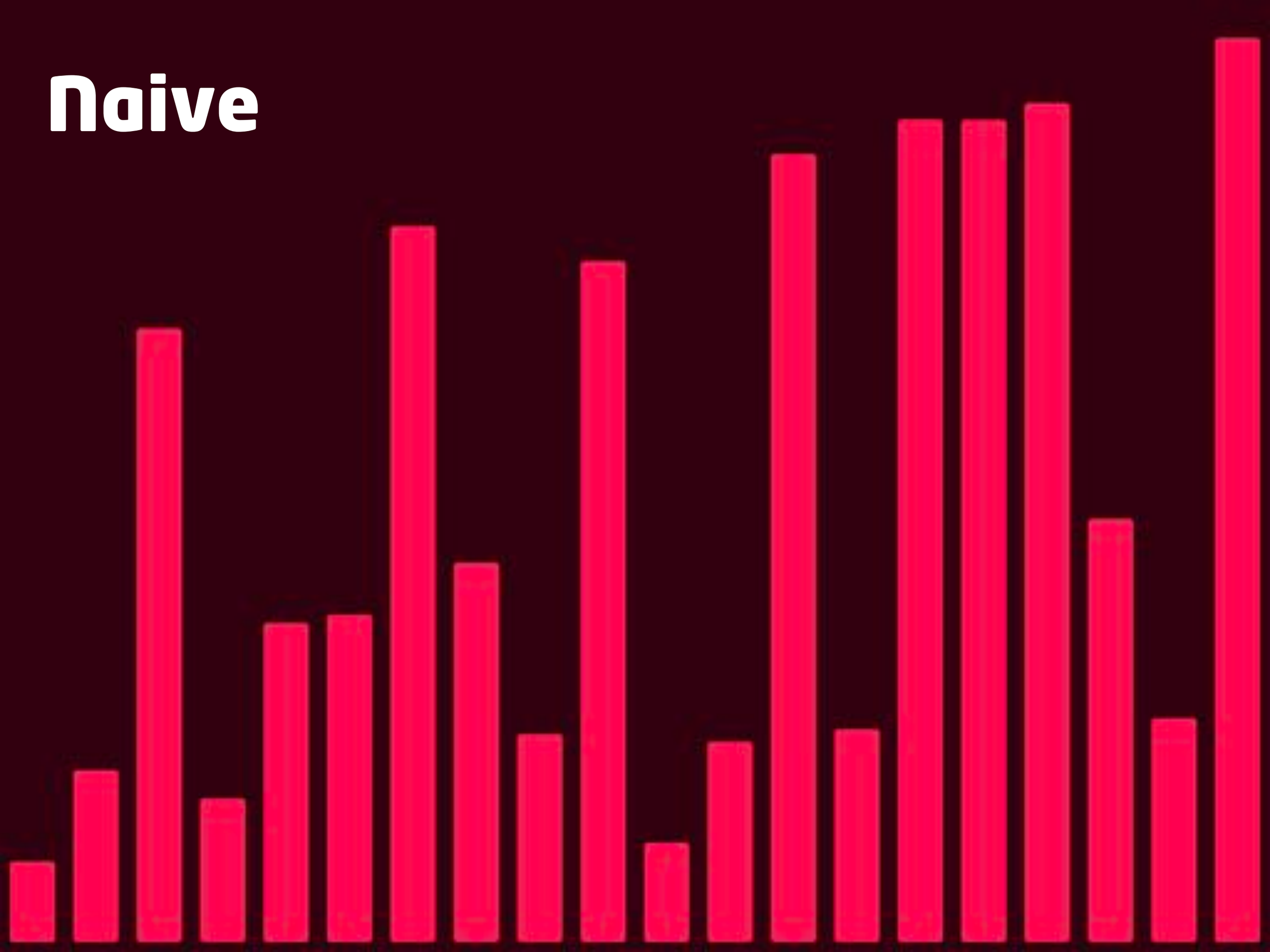
Naive

```
CGFloat(arc4random()) / CGFloat(UINT32_MAX)
```

Naive

Naive

Naive



Gem

```
import GameplayKit
```

Gem

```
import GameplayKit
```

Gem

```
import GameplayKit

let source = GKPerlinNoiseSource(
    frequency: 2,
    octaveCount: 3,
    persistence: 0.5,
    lacunarity: 2
)
```

Gem

```
import GameplayKit

let source = GKPerlinNoiseSource(
        frequency: 2,
        octaveCount: 3,
        persistence: 0.5,
        lacunarity: 2
)
```

Gem

```
let noise = GKNoise(source)
```

Gem

```
let map = GKNoiseMap(
    noise,
    size: vector2(1, 1),
    origin: vector2(0, 0),
    sampleCount: vector2(3, 5),
    seamless: true
)
```

Gem

```
map.value(at: vector2(0, 0))
```

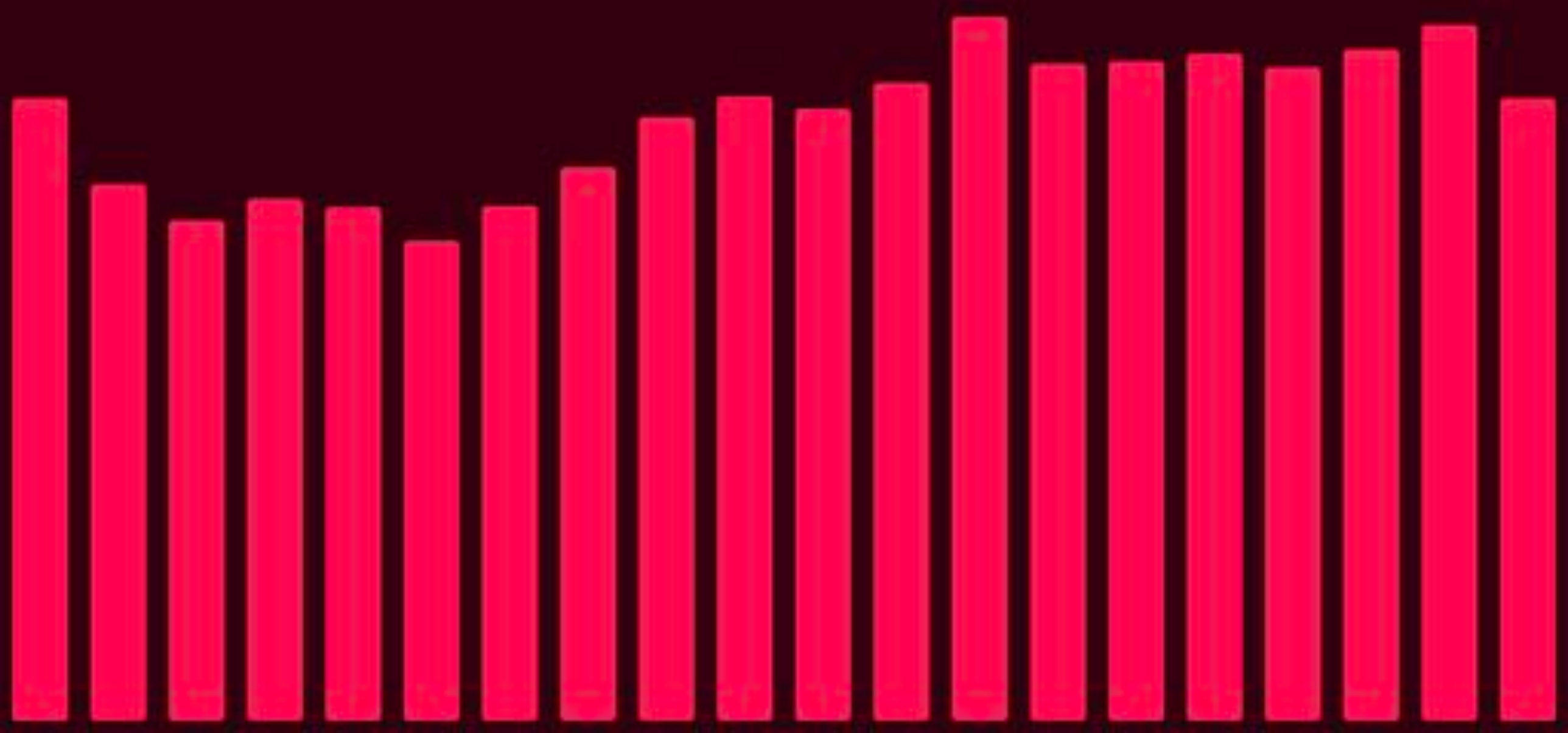
Gem

```
map.value(at: vector2(0, 0))

map.value(at: vector2(1, 0))
map.value(at: vector2(2, 0))
```

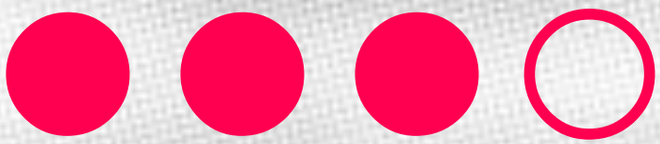
Gem

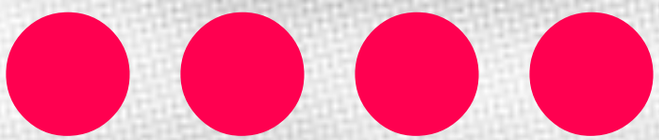
```
map.value(at: vector2(0, 0))

map.value(at: vector2(1, 0))
map.value(at: vector2(2, 0))

map.value(at: vector2(0, 1))
map.value(at: vector2(0, 2))
map.value(at: vector2(0, 3))
map.value(at: vector2(0, 4))
```

Gem

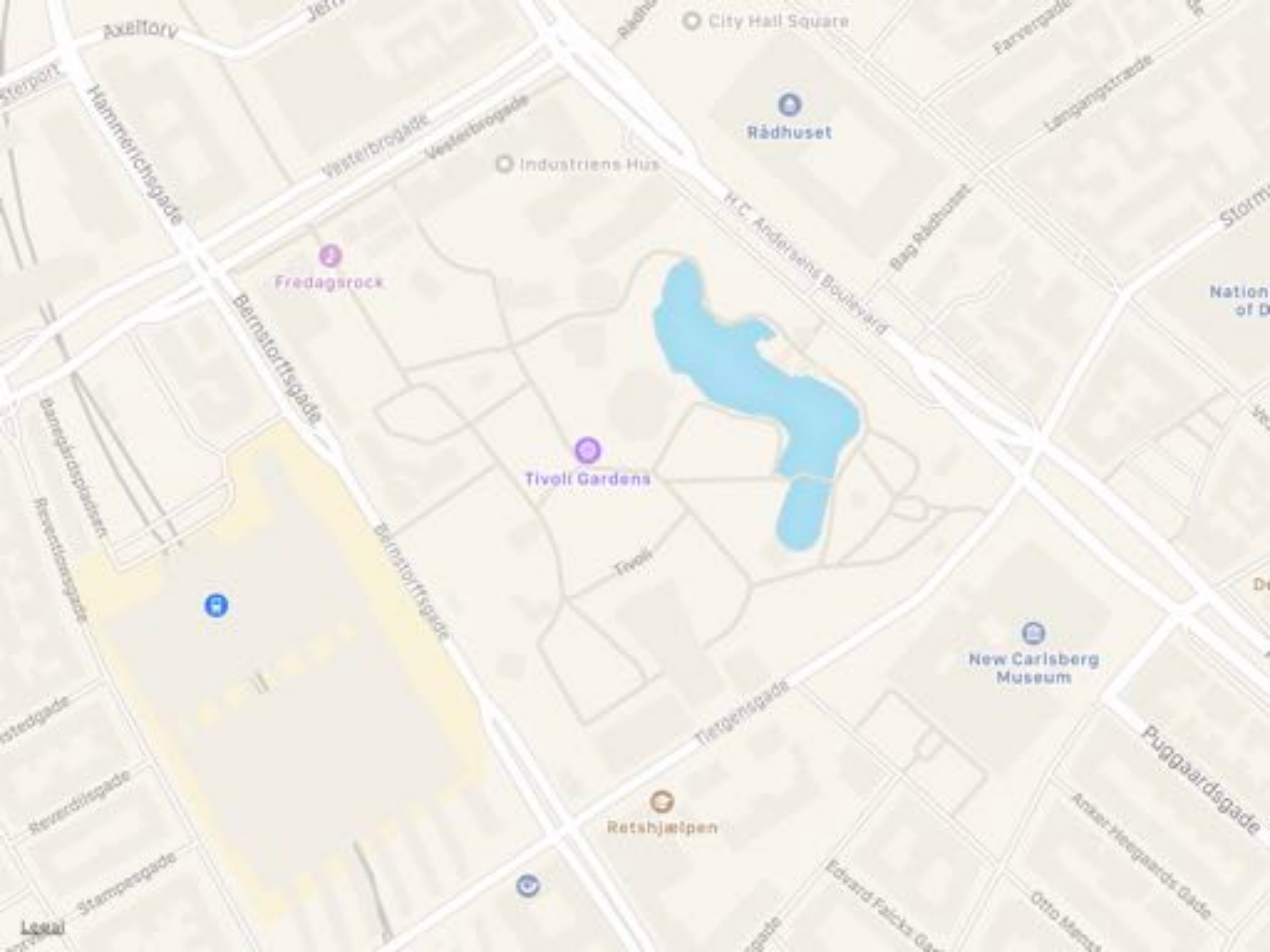


Gem





Path finding


City Hall Square
Axeltorv
Rådhuset
Farvergade
Langangstræde
Vesterbrogade
Vesterbrogade
Industriens Hus
Hammerichsgade
H.C. Andersens Boulevard
Bag Rådhuset
Fredagsrock
Bernstorffsgade
Tivoli Gardens
Tivoli
Bernstorffsgade
New Carlsberg Museum
Tietgensgade
Puggaardsgade
Retshjælpen
Reverdilsgade
Stampesgade
Edvard Falcks Gade
Anker Heegaards Gade
Legal

Tivoli Gardens, Copenhagen

Obstacles

Open Street Map data

Obstacles

```
let obstacle =

GKPolygonObstacle(
        points: [
                float2(0, 0),
                float2(0, 2),
                float2(1, 2),
                float2(1, 0)
        ]
)
```

⚠️ Counterclockwise[1]

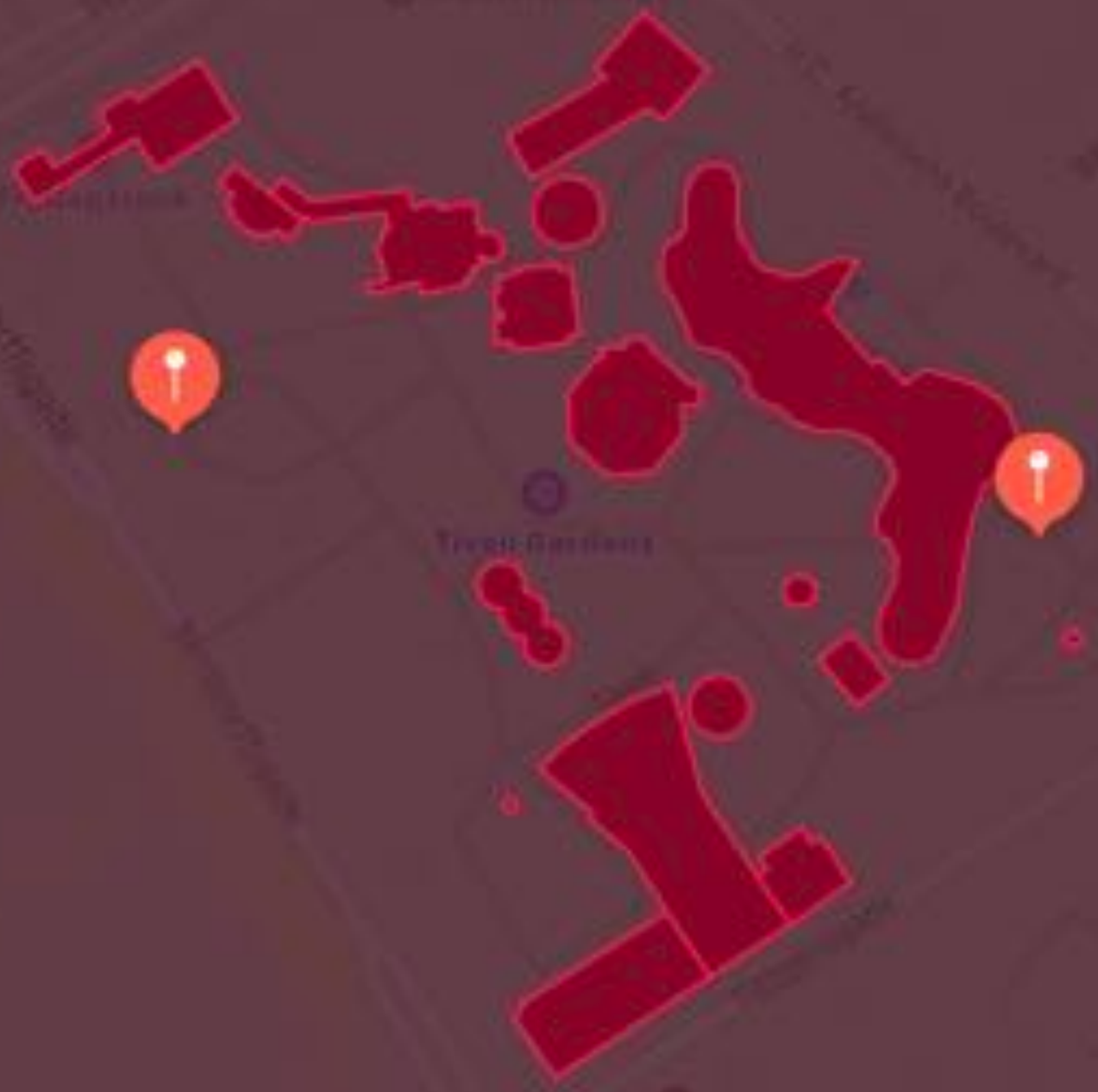
```
let obstacle =

GKPolygonObstacle(
    points: [
        float2(0, 0),
        float2(1, 0),
        float2(1, 2),
        float2(0, 2)
    ]
)
```

Obstacle Graph

```
let graph =

GKObstacleGraph(
        obstacles: [obstacle],
        bufferRadius: 0
)
```

Obstacles

A → B

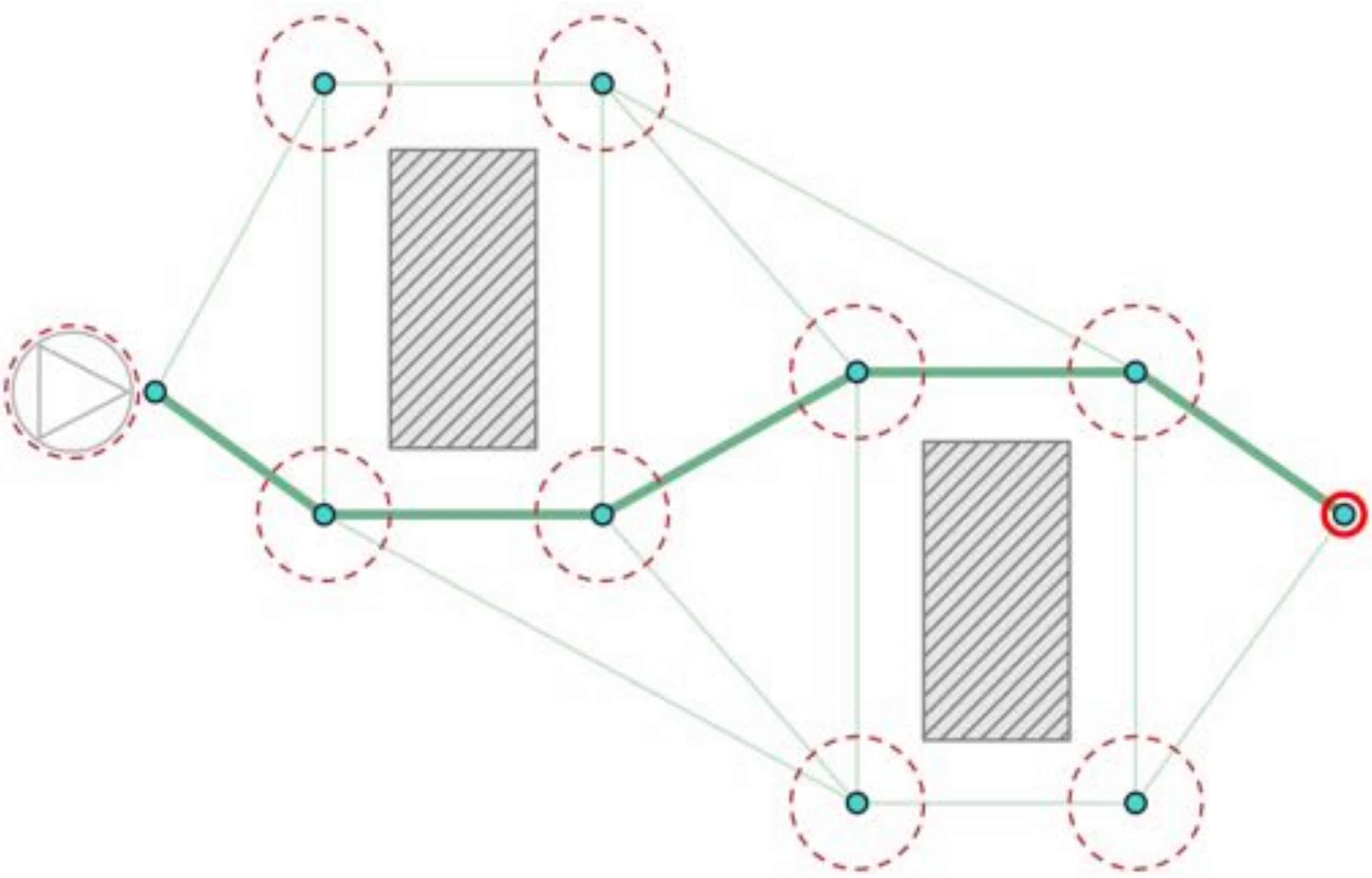


Path

```
let from = GKGraphNode2D(point: float2(x: -1, y: 1))
let to   = GKGraphNode2D(point: float2(x:  2, y: 1))
```

Path

```
let from = GKGraphNode2D(point: float2(x: -1, y: 1))
let to   = GKGraphNode2D(point: float2(x:  2, y: 1))

graph.connectUsingObstacles(node: from)
graph.connectUsingObstacles(node: to)
```



Path

```
let from = GKGraphNode2D(point: float2(x: -1, y: 1))
let to   = GKGraphNode2D(point: float2(x:  2, y: 1))

graph.connectUsingObstacles(node: from)
graph.connectUsingObstacles(node: to)
```

```
let path = graph.findPath(from: from, to: to)
```

Path

```
let from = GKGraphNode2D(point: float2(x: -1, y: 1))
let to   = GKGraphNode2D(point: float2(x:  2, y: 1))

graph.connectUsingObstacles(node: from)
graph.connectUsingObstacles(node: to)
```

```
let path = graph.findPath(from: from, to: to)
```

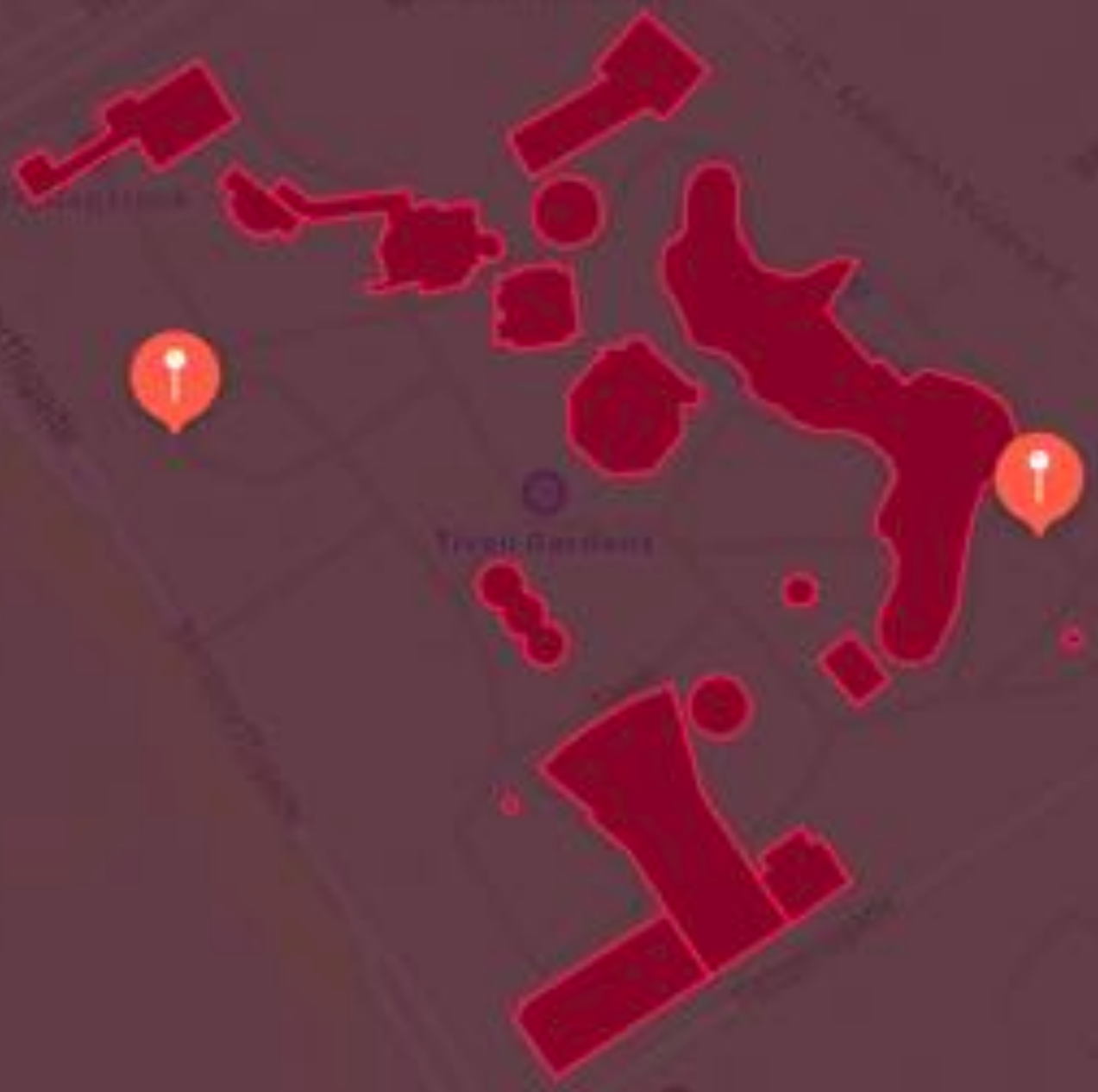
```
[
    GKGraphNode2D: {-1, 1},
    GKGraphNode2D: { 0, 0},
    GKGraphNode2D: { 1, 0},
    GKGraphNode2D: { 2, 1}
]
```

Buffer Radius

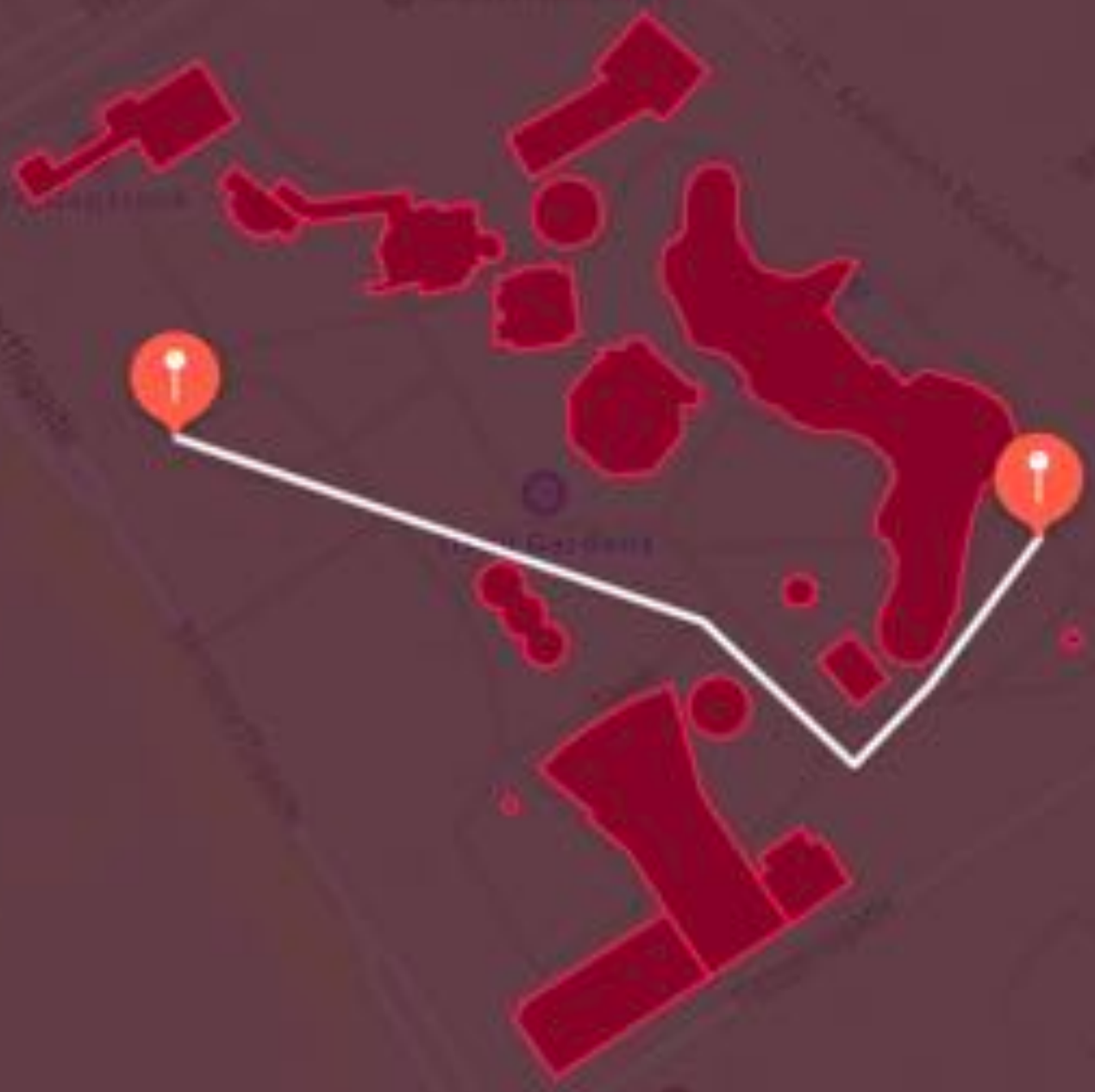
```
let graph = GKObstacleGraph(
      obstacles: [obstacle],
      bufferRadius: 0.5
)
```

```
[
      GKGraphNode2D: {-1.0, 1.0},
      GKGraphNode2D: {-0.5, 0.0},
      GKGraphNode2D: { 1.5,-0.5},
      GKGraphNode2D: { 2.0, 1.0}
]
```

A → B



A → B



C → D

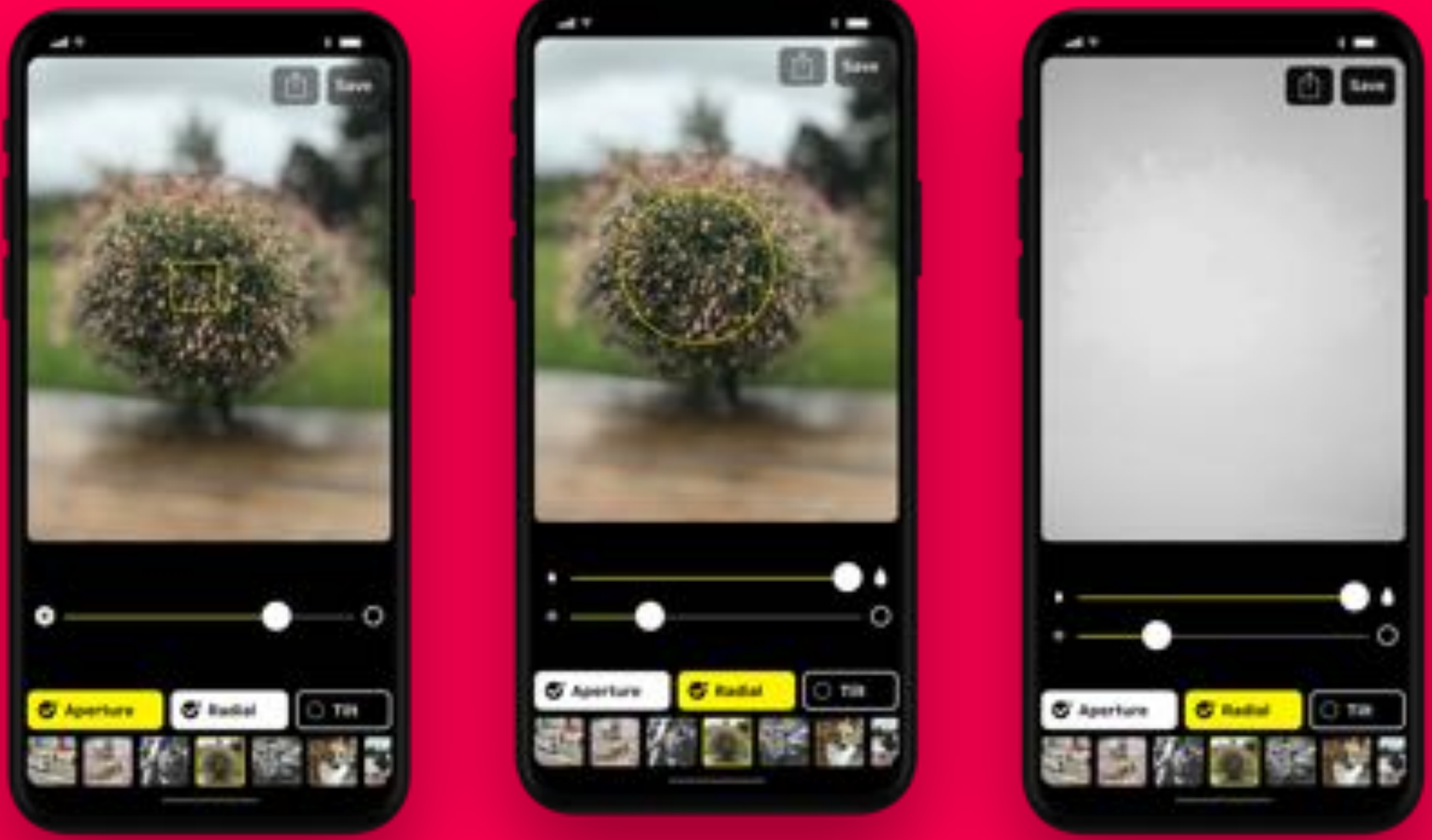
Links

- About GameplayKit *by Apple*
- Random Talk: The Consistent World of Noise *by Natalia Berdys*
- Playground Examples

Gems of Game play Kit

CocoaHeads Malmö – Tobias Due Munk – 7 September 2017

slor


Save
Aperture
Radial
Tilt
Save
Aperture
Radial
Tilt
Save
Aperture
Radial
Tilt



iPhone 7 Plus

+

iOS 11

=

Beta Tester

developmunk.dk

/slor

Gems of Game play Kit

CocoaHeads Malmö – Tobias Due Munk – 7 September 2017