Unity Engineer: Take Home Test

Task

Showcase your capability in creating a Unity x Native Swift implementation.

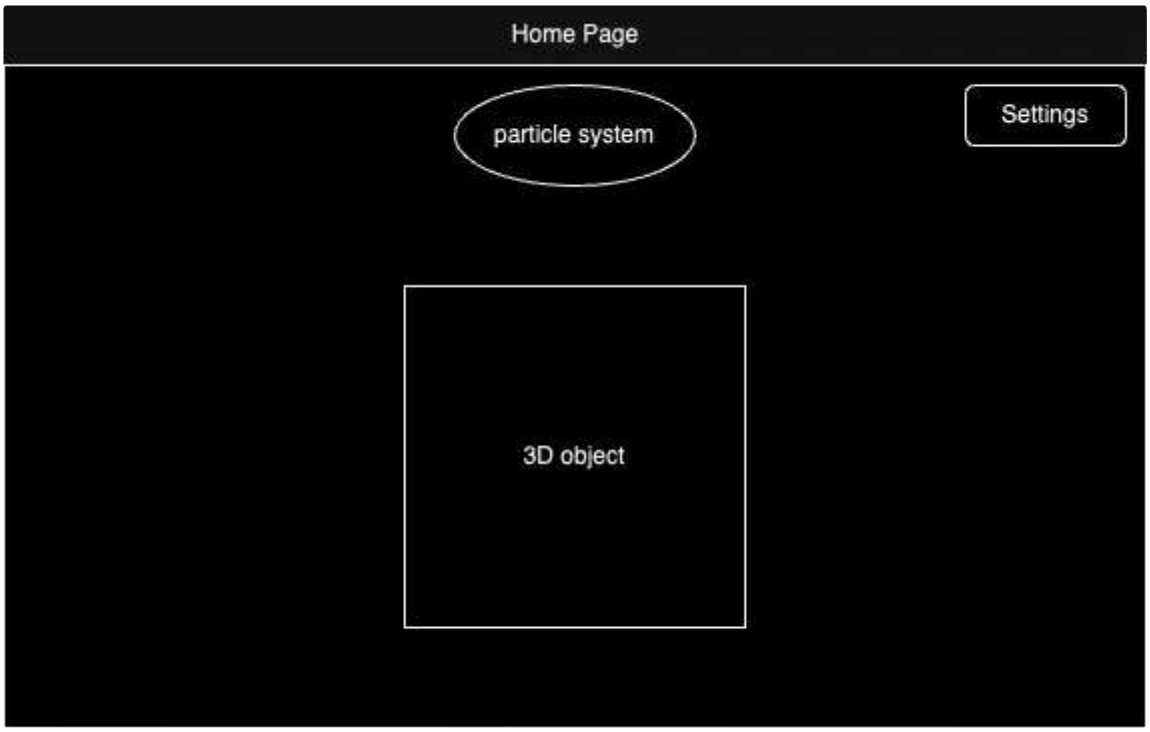
Create a simple app that show a 3D object with several integrations with native counterpart.

Assumptions

- You are working in a team. Your main expertise is in Unity, there will be iOS Engineer in the team. As for this test, native side code will be provided.
- You need to make the bridging between native swift and unity code to communicate with each other.
- The target is only for iOS Platform

Minimum Specification

- Home Screen



  - show a 3D object (can be any object of your choice - the object itself doesn't matter).
  - When user opens the home screen animate the object as an intro (translation / rotation / scale) - be creative! Use Unity's animator for this purpose
  - After the intro animation is finished, allow user to interact with the object. User can ONLY rotate the object, limit other interactions
  - Create an entry point to settings page on the top right corner
  - Create a particle system that will produce a burst of fire sparks from the middle top of the screen

- Settings Page

  - Show current date and time as a marquee text (moving from right of the screen to the left)
  - Show a button with text `Open Native Page`
- Native Code Integration: Sending Data to Native Side
  - When the object is rotated: call this block of function in swift

```
func objectRotated(x: Float, y: Float, z: Float) { print("Object is rotated: \(x), \(y), \(z)") }
```

  - The function will receive 3 float parameters that represents x, y and z rotation of the object

- Native Code Integration: Receiving data and command from native side
  - implement this class on iOS Code base

```
class RandomIntervalRunner { private var timer: Timer? func start(action:
@escaping () -> Void) { scheduleNextRun(action: action) } func stop() {
timer?.invalidate() timer = nil } private func scheduleNextRun(action:
@escaping () -> Void) { let randomInterval = TimeInterval(Int.random(in:
1...5)) timer = Timer.scheduledTimer(withTimeInterval: randomInterval,
repeats: false) { [weak self] _ in action() self?.scheduleNextRun(action:
action) // Schedule the next run } } }
```

  - When Home screen page is ready, call this function in swift. This function should trigger a part of the Unity code that will instruct the particle system to shoot a burst of fire spark particles for 0.5 seconds with the color produced from native side passed as parameter into Unity

```
let runner = RandomIntervalRunner() func setupEmitter() { runner.start {
let red = Float.random(in: 0.0...1.0) let green = Float.random(in:
0.0...1.0) let blue = Float.random(in: 0.0...1.0) print("Action executed
at \(Date()). Color: \(red) \(green) \(blue)") // TODO: call the unity
code here to trigger the spark } }
```

- Native Code Integration: Opening new Native Page
  - When `Open Native Page` in settings is pressed, call this function in swift

```
func showNativePage() { let vc = UIViewController()
vc.view.backgroundColor = .blue self.present(vc, animated: true) }
```

Optional Specification (Nice to Have)

- Intro animation creativity
- Custom shaders on Unity to make the object pop

- ray tracing
- camera post-processing

Requirements

- Deployment Target: iOS version 13.0+
- Use Unity 6 LTS version

Items to Submit

- Zip of the unity project. make sure the iOS build folder is included in the zip
- readme.md file to include details of your project structure & any explanations you