**Ý tưởng trò chơi:** Game Sky Bridges với ý tưởng dựa trên các tựa game running infinity nổi tiếng như Subway, Temple Run, v.v. …  
**Môi trường:** Trò chơi diễn ra trong một môi trường có đặc điểm địa hình gồm các con đường uốn lượn, các cầu xoay.

**Nhân vật:**  Người chơi có thể lựa chọn một nhân vật từ các nhân vật khác nhau.

**Thu thập vật phẩm:** Người chơi có thể thu thập các đồng xu trên các đoạn đường để mua nhân vật trong cửa hàng

**Điểm số và thành tích:** Điểm số của người chơi được tính dựa trên khoảng cách đã chạy được và số lượng đồng xu thu thập được trên đường chạy.

**Cốt truyện:** Trò chơi xoay quanh nhân vật chính cố gắng chạy trên các đoạn đường rời rạc nối với nhau bằng những cây cầu xoay và cố gắng thu thập càng nhiều vàng càng tốt để mở khóa các vật phẩm.

**Gameplay:** Trò chơi không có điểm dừng, người chơi cứ chạy đi mà không có mục tiêu cụ thể cho đến khi bị rơi khỏi đường chạy. Người chơi chỉ với 1 thao tác duy nhất là click vào màn hình để xoay cầu sao cho nhân vật chính không rơi khỏi đường. Người chơi cố gắng chạy càng xa càng tốt và thu thập điểm số cao nhất có thể.

**Thuật toán Game Sky Bridges  
Start:**   
**Khởi tạo giá trị ban đầu cho trò chơi**:

* ScoreManager.Instance.Reset(): Đặt lại điểm số của người chơi về giá trị ban đầu.

**Thiết lập danh sách đường đi và các giá trị khởi đầu**:

* listPath.Add(firstPath): Thêm đường đầu tiên (firstPath) vào danh sách các đường đi.
* turnPathNumberPooled = initialPath: Đặt số lượng đường rẽ tối đa ban đầu.
* goldNumberPooled = initialPath \* 3: Đặt số lượng vàng tối đa ban đầu (gấp ba lần số lượng đường rẽ).

**Lấy kích thước của đoạn đường đầu tiên**:

* planeSize = LastPlaneOfPath(firstPath).GetComponent<Renderer>().bounds.size: Lấy kích thước của đoạn đường đầu tiên bằng cách lấy đối tượng Renderer và truy cập thuộc tính bounds của nó.

**Xác định số lượng đường đi ngẫu nhiên**:

* pathNumber = Random.Range(minPathNumber, maxPathNumber): Sinh ra một số ngẫu nhiên trong khoảng từ minPathNumber đến maxPathNumber để xác định số lượng đoạn đường.

**Thiết lập vị trí và hướng tạo đường đi**:

* fixedPosition = new Vector3(LastPlaneOfPath(firstPath).transform.position.x, firstPath.transform.position.y, LastPlaneOfPath(firstPath).transform.position.z): Xác định vị trí cố định dựa trên vị trí của đoạn đường cuối cùng trong đường đầu tiên.
* pathCreationDirection = Vector3.right: Đặt hướng tạo đường đi là hướng sang phải.
* pathCreationRotation = new Vector3(0, 0, 0): Đặt góc xoay của việc tạo đường đi.

**Tính toán vị trí tiếp theo của đường đi**:

* nextPathPosition = fixedPosition + pathCreationDirection \* planeSize.x: Xác định vị trí tiếp theo để tạo đoạn đường mới bằng cách thêm kích thước của đoạn đường vào vị trí cố định.

**Chuẩn bị trò chơi và bắt đầu Coroutine kiểm tra và làm mới đường đi**:

* CRPrepareGame(): Gọi hàm chuẩn bị trò chơi (nội dung của hàm này không có trong đoạn mã).
* StartCoroutine(CheckAndRefeshPath()): Bắt đầu Coroutine để kiểm tra và làm mới đường đi.

**Khởi tạo các đối tượng: vàng, đoạn đường rẽ trái, phải**  
**Khởi tạo các đối tượng vàng (Gold)**:

* Sử dụng vòng lặp for để khởi tạo một số lượng vàng bằng giá trị của goldNumberPooled.
* Mỗi lần lặp:
  + Tạo một đối tượng vàng mới từ mẫu (goldPrefab) bằng cách sử dụng Instantiate.
  + Đặt đối tượng vàng này thành không hoạt động (SetActive(false)).
  + Đặt đối tượng vàng này làm con của CoinManager.Instance.transform.

**Khởi tạo các đoạn đường rẽ trái**:

* Khởi tạo biến turnLeftPathIndex với giá trị ban đầu là -1.
* Sử dụng vòng lặp for để khởi tạo một số lượng đoạn đường rẽ trái bằng giá trị của turnPathNumberPooled.
* Mỗi lần lặp:
  + Tính toán turnLeftPathIndex để lần lượt lấy các mẫu đường rẽ trái từ mảng turnLeftPathArray. Nếu đã tới cuối mảng, nó sẽ quay lại đầu mảng.
  + Tạo một đoạn đường rẽ trái mới từ mẫu tương ứng bằng cách sử dụng Instantiate.
  + Đặt đoạn đường rẽ trái này thành không hoạt động (SetActive(false)).
  + Đặt đoạn đường rẽ trái này làm con của turnLeftPathManager.transform.

**Khởi tạo các đoạn đường rẽ phải**:

* Tương tự như các đoạn đường rẽ trái, nhưng sử dụng biến turnRightPathIndex và mảng turnRightPathArray.

**Tạo các đoạn đường ban đầu**:

* Sử dụng vòng lặp for để tạo một số lượng đoạn đường ban đầu bằng giá trị của initialPath.
* Mỗi lần lặp, gọi hàm CreatePath() để tạo một đoạn đường mới.

**Tạo các đoạn đường đi :**

**Tăng pathCounter:**

* Mỗi lần hàm CreatePath được gọi, pathCounter được tăng lên để theo dõi số đoạn đường đã được tạo ra trong chuỗi hiện tại.

**Kiểm tra xem có phải là đoạn đường cuối cùng trong chuỗi không:**

* Nếu pathCounter bằng với pathNumber, tức là đã đến đoạn đường cuối cùng trong chuỗi hiện tại, cần thay đổi hướng của đường.

**Đặt lại pathCounter và chọn ngẫu nhiên số lượng đoạn đường cho chuỗi tiếp theo:**  
pathCounter = 0;

* pathCounter được đặt lại về 0.
* pathNumber được chọn ngẫu nhiên trong khoảng từ minPathNumber đến maxPathNumber.
* turn được đổi dấu để thay đổi hướng đường (nếu đang đi thẳng thì sẽ rẽ phải, nếu đang rẽ phải thì sẽ rẽ trái).

**Xác định hướng và góc quay của đoạn đường tiếp theo:**

* pathCreationDirection và pathCreationRotation được xác định dựa trên giá trị của turn.

**Tạo đoạn đường rẽ và đặt vị trí cho nó:**

* Đoạn đường rẽ (trái hoặc phải) được tạo ra và đặt tại vị trí nextPathPosition.
* CreateGold(turnPath) được gọi để tạo các vật phẩm vàng trên đoạn đường.

**Cập nhật vị trí cho đoạn đường tiếp theo:**

* fixedPosition được cập nhật dựa trên vị trí của đoạn cuối của đoạn đường hiện tại.
* nextPathPosition được cập nhật dựa trên fixedPosition và hướng pathCreationDirection.
* Đoạn đường rẽ vừa tạo được thêm vào danh sách listPath.

**Nếu không phải đoạn đường cuối cùng trong chuỗi:**  
Tạo một đoạn đường ngẫu nhiên từ pathArray.

* Đặt đoạn đường tại nextPathPosition.
* Gọi CreateGold(path) để tạo các vật phẩm vàng trên đoạn đường.
* Cập nhật fixedPosition và nextPathPosition như trong bước trên.
* Thêm đoạn đường vào listPath.

**Tạo các đối tượng vàng trên một đoạn đường cụ thể:**

**Bước 1: Xóa danh sách các vị trí vàng**

* Xóa danh sách listGoldPosition để chuẩn bị cho việc lưu trữ các vị trí mới của các đối tượng không phải là cầu (Bridge) trên đoạn đường (path).

### **Bước 2: Duyệt qua tất cả các con của đoạn đường và lưu trữ vị trí của chúng nếu chúng không phải là cầu**

* **Duyệt qua tất cả các con của đối tượng path**:
  + Lấy từng con và kiểm tra xem nó có phải là đối tượng có tag Bridge hay không.
  + Nếu không phải là cầu (Bridge), lưu trữ vị trí của nó vào danh sách listGoldPosition.

### **Bước 3: Tạo vàng tại các vị trí đã lưu trữ**

**Duyệt qua từng vị trí trong listGoldPosition**:

* Sử dụng một giá trị ngẫu nhiên để quyết định có đặt vàng tại vị trí này hay không (Random.value <= goldFrequency).
* Nếu điều kiện trên đúng:
  + Gọi hàm GetGold() để lấy hoặc tạo một đối tượng vàng.
  + Đặt vị trí của đối tượng vàng ở tọa độ (pos.x, 1, pos.z) (giả sử y = 1 là độ cao cố định của vàng).
  + Đặt đối tượng vàng làm con của path.transform.