1. **Giới thiệu  
   1.1 Lý do chọn đề tài**

**1.1.1 Tầm quan trọng của ngành công nghiệp game**:

* Ngành công nghiệp game hiện nay đóng vai trò quan trọng trong nhiều khía cạnh của xã hội và kinh tế. Dưới đây là một số điểm nổi bật về tầm quan trọng của ngành công nghiệp này:
  + Kinh tế: Ngành công nghiệp game là một trong những ngành có tốc độ phát triển nhanh nhất và mang lại doanh thu khổng lồ trên toàn thế giới. Doanh thu từ game không chỉ đến từ việc bán game mà còn từ các dịch vụ liên quan như eSports, streaming, và các giao dịch trong game.
  + Công nghệ: Game thường là động lực thúc đẩy sự phát triển của công nghệ, đặc biệt trong các lĩnh vực như đồ họa, trí tuệ nhân tạo, thực tế ảo (VR), và thực tế tăng cường (AR). Các công nghệ tiên tiến được phát triển cho game thường được ứng dụng vào nhiều lĩnh vực khác nhau, từ y tế đến giáo dục.
  + Văn hóa: Game đã trở thành một phần không thể thiếu của văn hóa đại chúng. Nhiều tựa game nổi tiếng đã tạo ra các cộng đồng người chơi đông đảo và ảnh hưởng sâu rộng đến nhiều mặt của đời sống, từ ngôn ngữ, âm nhạc đến thời trang.
  + Giải trí và giáo dục: Game cung cấp một hình thức giải trí phong phú và đa dạng, giúp người chơi thư giãn và giải tỏa stress. Bên cạnh đó, nhiều game cũng mang tính giáo dục cao, giúp phát triển kỹ năng tư duy, chiến thuật, và hợp tác.
  + Xã hội: Game có thể tạo ra môi trường xã hội hóa, nơi người chơi có thể kết bạn, giao lưu và phát triển kỹ năng xã hội. Các sự kiện eSports và hội nghị game cũng là cơ hội để người chơi và nhà phát triển gặp gỡ, trao đổi kinh nghiệm.
  + Tạo việc làm: Ngành công nghiệp game tạo ra hàng triệu việc làm trên toàn thế giới, từ lập trình viên, nhà thiết kế, nghệ sĩ đồ họa, đến nhà sản xuất và nhà phát hành. Đây là một ngành công nghiệp đa dạng và đòi hỏi nhiều kỹ năng khác nhau.

**1.1.2 Sự phổ biến và tiềm năng của Unity trong lĩnh vực phát triển game:**

* **Sự phổ biến của Unity**
  + **Đa nền tảng:** Unity hỗ trợ xuất bản game trên nhiều nền tảng khác nhau, bao gồm PC, MAc, Android, iOS, Web v.v. Điều này giúp các nhà phát triển dễ dàng tiếp cận được nhiều người chơi hơn.
  + **Dễ tiếp cận và sử dụng:** Giao diện thân thiện và dễ sử dụng của Unity giúp cả những người mới bắt đầu lẫn các nhà phát triển chuyên nghiệp có thể nhanh chóng làm quen và bắt đầu phát triển game. Unity cũng cung cấp nhiều tài liệu hướng dẫn và cộng đồng hỗ trợ rộng lớn.
  + **Cộng đồng mạnh mẽ:** Unity có một cộng đồng lớn và năng động, nơi các nhà phát triển có thể chia sẻ kinh nghiệm, hỗ trợ lẫn nhau và cùng nhau giải quyết các vấn đề kỹ thuật. Nhiều khóa học trực tuyến, diễn đàn và sự kiện được tổ chức nhằm hỗ trợ và phát triển cộng đồng này.
  + **Tài nguyên phong phú:** Unity Asset Store cung cấp hàng ngàn tài nguyên (assets) bao gồm mô hình 3D, âm thanh, scripts và nhiều thứ khác, giúp tiết kiệm thời gian và công sức trong quá trình phát triển game.
* **Tiềm năng của Unity:**
  + **Độ linh hoạt và mạnh mẽ:** Unity không chỉ dành cho phát triển game mà còn được sử dụng trong nhiều lĩnh vực khác như kiến trúc, y tế, phim ảnh, và thực tế ảo. Điều này mở ra nhiều cơ hội cho các nhà phát triển mở rộng phạm vi ứng dụng của Unity.
  + **Cập nhật và cải tiến liên tục**: Unity Technologies liên tục cập nhật và cải tiến công cụ này, bổ sung các tính năng mới và tối ưu hóa hiệu suất. Điều này đảm bảo rằng Unity luôn theo kịp các xu hướng và yêu cầu của ngành công nghiệp game hiện đại.

**1.1.3. Mục tiêu của đồ án**

* **Mục tiêu chính:**
  + Tìm hiểu tổng quan về kiến trúc và cách thức hoạt động trong Unity.
  + Tìm hiểu về các vấn đề như game scene, gameobject, animation, game model, sound audio, các hiệu ứng particle,v.v. để từ đó đưa ra giải pháp.
  + Sử dụng C# và Unity để viết mã nguồn, giải quyết các vấn đề kỹ thuật và tối ưu hóa hiệu suất.
* **Mục tiêu cụ thể:** 
  + Xây dựng và phát triển game Sky Bridges bằng Unity.
  + Các chức năng trong game:
    - Người chơi sẽ được tham gia vào một hành trình vô tận.
    - Nhiệm vụ của người chơi là điều khiển các đoạn cầu xoay cho đến khi nhân vật đi đến cuối.
    - Điều kiện thua của trò chơi là khi nhân vật rơi khỏi cầu.
  + Người chơi phải thật cẩn thận để có thể ghi nhiều điểm nhất và kiếm thật nhiều tiền để mở khóa các nhân vật mới.
  + Sử dụng âm thanh và nhạc nền để tăng cường trải nghiệm người chơi
  + Đảm bảo rằng trò chơi chạy mượt mà trên nhiều nền tảng khác nhau, tối ưu hóa mã nguồn và sử dụng tài nguyên hợp lý.
  + Thực hiện kiểm thử liên tục trong suốt quá trình phát triển, tìm và sửa lỗi để đảm bảo trò chơi hoạt động ổn định và không gặp vấn đề kỹ thuật.
  + Chuẩn bị và xuất bản trò chơi trên các nền tảng mong muốn đồng thời tạo kế hoạch tiếp thị và phân phối để tiếp cận người chơi.
* **Mục tiêu học tập:**
  + Hiểu rõ và thành thạo các tính năng và công cụ của Unity, từ giao diện người dùng đến các kỹ thuật lập trình nâng cao.
  + Lên kế hoạch, phân chia công việc, theo dõi tiến độ và quản lý tài nguyên hiệu quả trong quá trình thực hiện đồ án.

**1.2. Mục tiêu nghiên cứu**

* **Khám phá và hiểu biết về công cụ Unity:** 
  + **Tìm hiểu sâu về Unity:** Nghiên cứu các thành phần và công cụ có sẵn trong Unity, như Editor, Asset Store, các công cụ tạo hoạt ảnh, và hệ thống vật lý.
  + **Phân tích tính năng mới**: Đánh giá các tính năng mới của Unity và tìm hiểu cách chúng có thể được áp dụng trong dự án.
* **Phát triển kỹ thuật lập trình và logic game**:
  + **Nghiên cứu về C#**: Nắm vững ngôn ngữ lập trình C# và cách áp dụng nó trong Unity để phát triển game.
  + **Phát triển logic game**: Nghiên cứu các thuật toán và cách áp dụng chúng để xử lý các hành vi nhân vật trong game.
* **Thiết kế trải nghiệm người dùng (UX) và giao diện người dùng (UI)**:
  + **Nghiên cứu UX/UI**: Hiểu rõ các nguyên tắc thiết kế UX/UI trong game, từ việc tạo menu, màn hình chọn nhân vật, đến các yếu tố HUD (Heads-Up Display).
  + **Tối ưu hóa trải nghiệm người chơi**: Tìm hiểu các phương pháp để cải thiện trải nghiệm người chơi thông qua thiết kế giao diện và tương tác.
* **Tạo và quản lý tài nguyên đồ họa**:
  + **Nghiên cứu kỹ thuật đồ họa**: Hiểu biết về các kỹ thuật tạo mô hình 3D, ánh sáng, và hiệu ứng đặc biệt trong Unity.
  + **Quản lý tài nguyên**: Tìm hiểu cách quản lý và tối ưu hóa các tài nguyên đồ họa để đảm bảo hiệu suất và chất lượng hình ảnh.
* **Tối ưu hóa hiệu suất và xử lý lỗi**:
  + **Nghiên cứu về tối ưu hóa**: Tìm hiểu các kỹ thuật tối ưu hóa hiệu suất game, từ việc tối ưu mã nguồn, quản lý bộ nhớ, đến sử dụng hiệu quả các công cụ profiling của Unity.
  + **Xử lý lỗi và kiểm thử**: Nghiên cứu các phương pháp kiểm thử game và xử lý lỗi để đảm bảo tính ổn định và chất lượng của sản phẩm.
* **Phân tích thị trường và nhu cầu người dùng**:
  + **Nghiên cứu thị trường game**: Tìm hiểu xu hướng thị trường, nhu cầu và sở thích của người chơi, từ đó đưa ra các quyết định thiết kế phù hợp.
  + **Phản hồi người dùng**: Thu thập và phân tích phản hồi từ người chơi để cải thiện và hoàn thiện game.

1. **Tổng quan về Unity**

**2.1. Giới thiệu về Unity**

**2.1.1. Unity là gì?**

Unity là một “cross- flatform game engine” tạm hiểu là công cụ phát triển game đa nền tảng được phát triển bởi Unity Technologies. Game engine này được sử dụng để phát trển game trên PC, consoles, thiết bị di động và trên websites.

**2.1.2. Quá trình phát triển:**

* Ra mắt đầu tiên vào năm 2005 tại sự kiện Apple’s Worldwide Developer Conference bởi nhà sáng lập David Helgason, trải qua gần 20 năm phát triển, nay Unity đã hoàn thiện hơn về rất nhiều mặt.
* Giai đoạn 2006-2010, Unity ra mắt liên tiếp 3 phiên bản 2.0, 2.5, 3.0 hỗ trợ phát triển trên nền tảng Windows , mở rộng phạm vi sử dụng của Unity và tạo cơ hội phát triển cho thị trường game di động.
* Giai đoạn 2011-2015, Unity phát triển mạnh mẽ với hàng loạt các kho tài nguyên và các hệ thống tích hợp và dần trở thành một trong những công cụ phát triển game phổ biến nhất trên thế giới.
* Giai đoạn 2016-2020, Unity tập trung cải thiện hiệu suất và ổn định, đồng thời thay đổi cách đặt tên phiên bản và chuyển sang mô hình phát triển theo năm, nhằm hoàn thiện và cải thiện các vấn đề thông qua từng phiên bản nhỏ, từ đó củng cố vị thế là một trong những công cụ phát triển game hàng đầu thế giới.
* Giai đoạn 2021 đến nay, Unity liên tục cái thiện hiếu suất, ra mắt nhiều công cụ mới cho nhà phát triển nhằm mở rộng ứng dụng trong nhiều lĩnh vực khác.

**2.1.3. Đặc điểm nổi bật:**

* Chức năng cốt lõi đa dạng bao gồm: cung cấp công cụ dựng hình (kết xuất đồ họa) cho các hình ảnh 2D hoặc 3D, công cụ vật lý (tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động, trí tuệ nhân tạo, phân luồng, tạo dò ng dữ liệu xử lý, quản lý bộ nhớ, dựng ảnh đồ thị và kết nối mạng. Nhờ có các engine mà công việc làm game trở nên ít tốn kém và đơn giản hơn.
* Hỗ trợ đa nền tảng: Một trong các thế mạnh của Unity3D chính là khả năng hỗ trợ gần như toàn bộ các nền tảng hiện có bao gồm: PlayStation 3, Xbox 360, Wii U, iOS, Android, Windows, Blackberry 10, OS X, Linux, trình duyệt Web và cả Flash. Nói cách khác, chỉ với một gói engine, các studio có thể làm game cho bất kỳ hệ điều hành nào và dễ dàng convert chúng sang những hệ điều hành khác nhau. Đồng thời, đây cũng là giải pháp cho các game online đa nền tảng – có thể chơi đồng thời trên nhiều hệ điều hành, phần cứng khác nhau như Web, PC, Mobile, Tablet….
* Dễ sử dụng: Unity3D được build trong một môi trường phát triển tích hợp, cung cấp một hệ thống toàn diện cho các lập trình viên, từ soạn thảo mã nguồn, xây dựng công cụ tự động hóa đến trình sửa lỗi. Do được hướng đến đồng thời cả lập trình viên không chuyên và studio chuyên nghiệp, nên Unity3D khá dễ sử dụng. Hơn nữa, đây là một trong những engine phổ biến nhất trên thế giới, người dùng có thể dễ dàng tìm kiếm kinh nghiệm sử dụng của “tiền bối” trên các forum công nghệ.

**2.2. Các thành phần trong Unity**

**2.2.1. Unity Editor**

* **Scene View**: Cung cấp một không gian làm việc 3D nơi các nhà phát triển có thể sắp xếp và chỉnh sửa các đối tượng trong cảnh.
* **Game View**: Hiển thị cách mà người chơi sẽ nhìn thấy và tương tác với trò chơi khi nó chạy.
* **Hierarchy Window**: Liệt kê tất cả các đối tượng trong cảnh hiện tại dưới dạng cây, giúp quản lý và sắp xếp các đối tượng dễ dàng.
* **Project Window**: Hiển thị tất cả các tài nguyên (assets) trong dự án, bao gồm mô hình, âm thanh, scripts, và các tài nguyên khác.
* **Inspector Window**: Cung cấp các tùy chọn và thông tin chi tiết về các đối tượng được chọn trong Scene hoặc Hierarchy.

**2.2.2. GameObjects và Components**

* **GameObjects**: Là các thực thể trong trò chơi, có thể đại diện cho các đối tượng vật lý, các thành phần UI, hoặc bất kỳ thực thể nào khác trong game.
* **Components**: Là các khối xây dựng thêm vào các GameObjects để xác định hành vi, hình dáng và tính chất của chúng. Ví dụ, một GameObject có thể có một Component RigidBody để thêm tính chất vật lý hoặc một Component AudioSource để phát âm thanh.

**2.2.3. Assets và Asset Store**

* **Assets**: Bao gồm tất cả các tài nguyên cần thiết để xây dựng trò chơi như mô hình 3D, texture, âm thanh, scripts, và animations.
* **Asset Store**: Là một cửa hàng trực tuyến nơi các nhà phát triển có thể mua hoặc tải miễn phí các tài nguyên và công cụ do cộng đồng và các nhà phát triển khác cung cấp.

**2.2.4. Scripting**

* **C#**: Unity sử dụng ngôn ngữ lập trình C# để viết scripts, giúp điều khiển hành vi của các GameObjects, quản lý logic trò chơi, và tạo ra các hệ thống phức tạp.
* **MonoBehaviour**: Là lớp cơ bản mà hầu hết các scripts trong Unity kế thừa, cung cấp các phương thức sự kiện như Start(), Update(), FixedUpdate(), và nhiều hơn nữa để quản lý vòng đời của các đối tượng trong trò chơi.

**2.2.5. Physics Engine**

* **Rigidbody**: Cung cấp các thuộc tính vật lý cho các GameObjects, bao gồm khối lượng, trọng lực, và các lực tác động.
* **Colliders**: Xác định hình dạng vật lý của các GameObjects để chúng có thể tương tác với nhau trong không gian 3D.
* **Physics Materials**: Xác định các thuộc tính như ma sát và độ nảy cho các bề mặt vật lý.

**2.2.6. Animation System**

* **Animator**: Quản lý các animations và trạng thái của chúng, cho phép chuyển đổi giữa các animations một cách mượt mà.
* **Animation Clips**: Định nghĩa các chuỗi animations, chẳng hạn như chạy, nhảy, hoặc tấn công.
* **Mecanim**: Hệ thống animation mạnh mẽ của Unity, cho phép tạo và quản lý các animations phức tạp và tương tác.

**2.2.7. Rendering**

* **Camera**: Xác định góc nhìn của người chơi trong game.
* **Lights**: Chiếu sáng các đối tượng trong cảnh, bao gồm các loại đèn Directional, Point, và Spot.
* **Shaders** và **Materials**: Xác định các bề mặt của các đối tượng sẽ được render, bao gồm texture, màu sắc, và các hiệu ứng đặc biệt.

**2.2.8. UI System**

* **Canvas**: Là khu vực chứa các thành phần UI như text, buttons, và images.
* **UI Components**: Bao gồm các thành phần như Text, Button, Slider, và các loại hình ảnh, giúp tạo và quản lý giao diện người dùng.

### **2.2.9. Audio**

* **Audio Source**: Phát âm thanh trong game, có thể được gắn vào các GameObjects.
* **Audio Listener**: Thường được gắn vào Camera chính để thu nhận âm thanh trong cảnh.
* **Audio Mixer**: Quản lý và điều chỉnh âm thanh từ nhiều nguồn khác nhau.

### **2.2.10. Networking**

* **UNet (Unity Networking):** Hỗ trợ phát triển các trò chơi nhiều người chơi qua mạng, cung cấp các công cụ và APIs để quản lý kết nối mạng, đồng bộ hóa trạng thái và dữ liệu.

### **2.2.11. Profiler**

* **Profiler:** Cung cấp công cụ theo dõi hiệu suất của game, giúp các nhà phát triển xác định và tối ưu hóa các vấn đề về hiệu suất.

1. **Thiết kế game**

**3.1. Ý tưởng và cốt truyện**

* **Mô tả ý tưởng game**
  + **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.

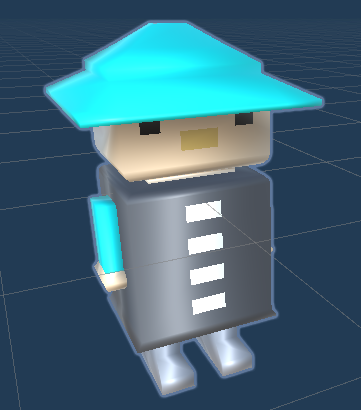
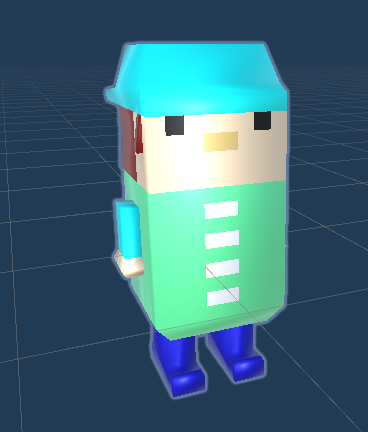
**3.2. Thiết kế nhân vật và môi trường**

**3.2.1. Thiết kế nhân vật**

* **Ý tưởng nhân vật:** 
  + Nhân vật được lấy cảm hứng trên Kogama.com.
  + Nhân vật được thiết kế sẽ là một thanh niên cố gắng thoát khỏi các đoạn đường rời rạc nối với nhau bằng những cây cầu xoay.
* **Thiết kế chi tiết :**

****

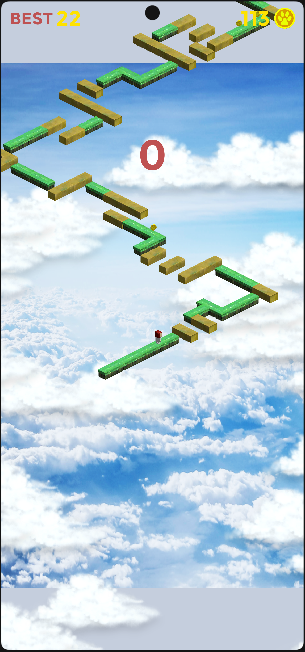
*Nhân vật chính*

******

*Các nhân vật phụ*

**3.2.2. Thiết kế môi trường**

* **Ý tưởng:** Bối cảnh địa hình trong game lấy cảm hứng từ vườn địa đàng với những cung đường rời rạc và những đoạn cầu xoay nối liền; xung quanh là khung cảnh những đám mây bồng bềnh trôi nổi.
* **Thiết kế chi tiết**

****

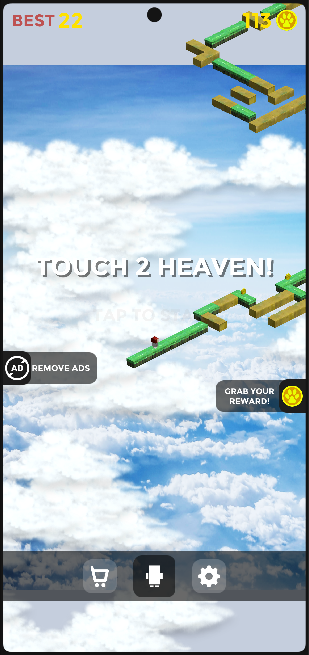
*Bối cảnh địa hình*

**3.3.3. Gameplay và cơ chế**

* **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.
* **Cơ chế điều khiển:** 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.

**3.3.4. Giao diện người dùng**

* **Main menu**

****

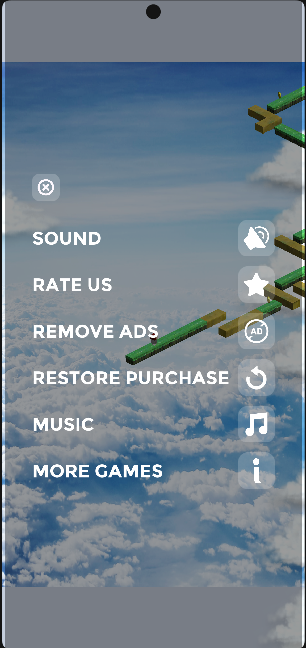
* **Buy gold menu**

****

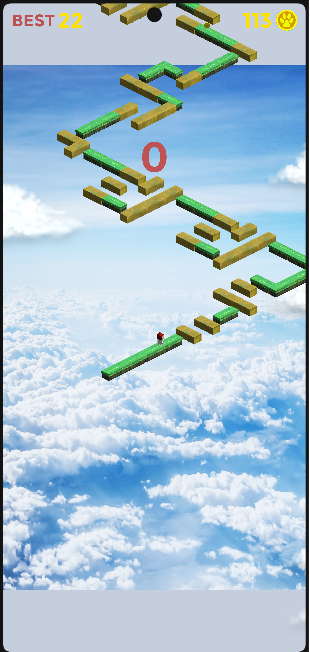
* **Select character menu**

****

* **Setting menu**

****

* **Gameplay Scene**

****

* **Game over**



1. **Phát triển game**

**4.1. Lập trình và triển khai các chức năng**  
**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.