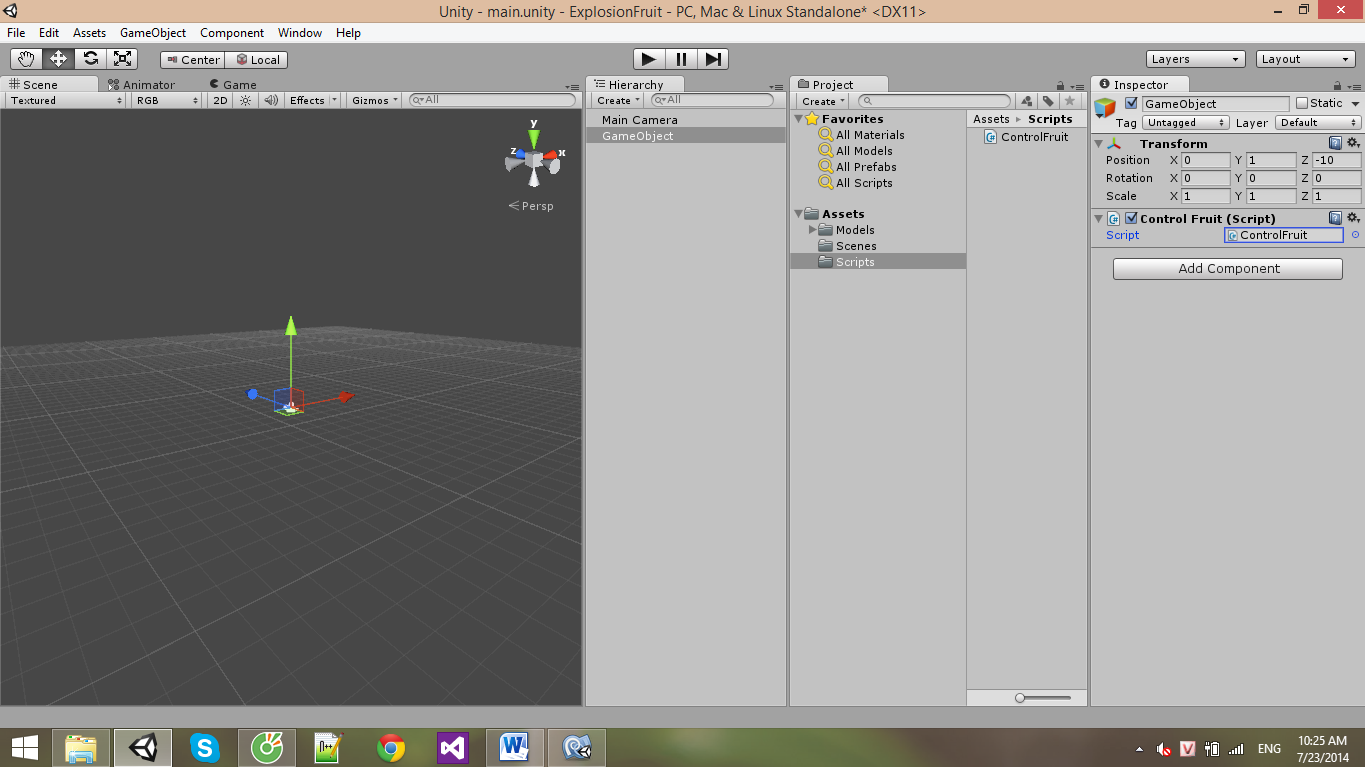
# Unity Basic

## Giao diện và nguyên lý làm việc của Unity Engine

### Giao diện

Unity là phần mềm tích hợp tất cả trong một để tạo một game 3d đa nền tảng. Khởi động Unity, giao diện chính bao gồm:



Các thành phần:

**Scene**: Là nơi dùng để thiết kế không gian game, game giống như một bộ phim, trước hết phải sắp xếp trường quay, sau đó di chuyển camera và các nhân vật bắt đầu diễn. Các nhân vật trong unity gọi là các GameObject.

**Hierarchy**: Quản lý các gameobject, mỗi gameobject được đưa vào scene sẽ được quản lý trên hierarchy.

**Project:** Quản lý các tài nguyên game, các tài nguyên này mới được đưa vào project còn muốn hiển thị trên scene cần kéo thả sang scene.

**Inspector:** Mỗi object sẽ bao gồm nhiều thành phần khác nhau, inspector có chức năng quản lý các components này. Ví dụ để ý trên hình ta đang chọn đối tượng gameobject, trên inspector của gameobjects có 2 thành phần chính là transform và ControlFruit. Transform là component gốc của mọi gameobject, ControlFruit là thành phần được code bằng C# có chức năng điều khiển đối tượng. Giống như kịch bản, khi game bắt đầu thì các diễn viên bắt đầu diễn theo kịch bản. Công việc chủ yếu của lập trình Unity là viết các kịch bản này.

Để bắt đầu game, bấm vào nút mũi tên (giống mediaplayer).

### Nguyên lý làm việc

Thành phần cơ bản nhất của game trong Unity là gameobject. GameObject đơn giản là một đối tượng chứa thành phần transform (tọa độ, góc xoay, tỉ lệ phóng to). Mọi đối tượng trong game (người chơi, quái vật…) đều là các gameobject. Để biến một gameobject thành các đối tượng khác nhau, ta thêm vào object các thành phần (components) khác nhau. Ví dụ nếu muốn biến object thành một cánh cửa có thể tự động mở ra vào, ta thêm component Mesh Render. Thành phần này sẽ có chức năng render ra một cánh cửa, khi thêm vào gameobject, kết hợp với transform (xác định vị trí, góc xoay, tỉ lệ) sẽ thành một đối tượng cụ thể trong game.

Công việc lập trình game Unity là viết các hành vi ứng xử (behaviour) của mỗi object. Ví dụ đối với cánh cửa, ta tạo một file C# DoorController.cs, trong file này viết sự kiện khi người chơi chạm vào cánh cửa, cánh cửa sẽ tự mở ra. Đoạn code ta vừa viết cũng chính là một component để thêm vào object cánh cửa. Một object có thể có nhiều component, nhiều script được thêm vào. Việc viết cụ thể các script sẽ được giải thích cụ thể ở phần sau.

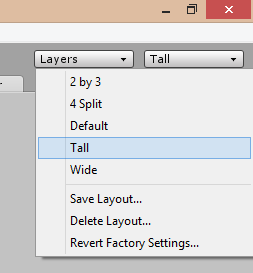
## Bắt tay làm game đầu tiên (FruitExplosion)

### Ý tưởng

Để làm quen, trước hết ta làm một game đơn giản đó là điều khiển một cái giỏ để hứng các quả rơi từ trên xuống. Nếu đỡ được thì sẽ tăng thêm một điểm, nếu không thì quả bị rơi ra ngoài sẽ nổ và gameover.

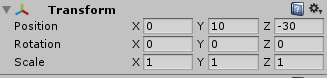
### Setup môi trường

Ở góc trên bên phải, chọn kiểu sắp xếp Tall, kiểu này sẽ dễ làm việc với game kiểu 3d.



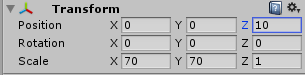
### Camera

Trong game camera không chuyển động, nên ta cố định camera. Chọn Main Camera, chỉnh sửa thông số như sau:



### Background

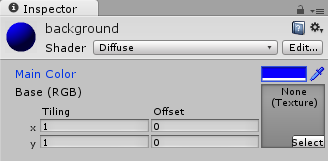
Chọn GameObject/Create Other/Cube. Một khối hộp sẽ được tạo trên scene. Bấm F để điều chỉnh vị trí nhìn để nhìn rõ khối hộp này. Điều chỉnh thông số như sau:



Khối này sẽ dùng làm nền cho game.

Trong phần Assets, nháy phải chọn Create/Folder. Đặt tên folder mới là Materials. Trong folder materials, nháy phải chọn Create/Material. Đặt tên cho material mới là background.

Chọn background, nhìn bên inspector, chọn Main Color là màu gì đó. Ví dụ:



Kéo thả background vào trong đối tượng Cube trên scene. Cả cube sẽ đổi thành màu vừa tạo. Bấm Play, kết quả nhận được sẽ là một màn hình hơi tối, lý do là chưa có ánh sáng.

### Light

Chọn GameObject/Create other/Directional Light. Xuất hiện đối tượng Directional Light trong Hierarchy. Có thể tùy chỉnh hướng của light bằng cách, chọn đối tượng Light. Ở phía trên bên trái có các phím chức năng như sau:



Thứ tự các nút lần lượt là xem, di chuyển, xoay và thu phóng. Tương ứng các phím tắt Q, W, E, R. Để xoay hướng ánh sáng, chọn light, bấm e, sau đó trên scene kéo thả các thanh để xoay ánh sáng đúng ý.

Bấm Play, giờ nhìn màn hình đã sáng sủa lên rồi.

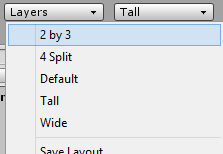
Ngoài việc sử dụng Directional Light, Unity còn cung cấp 3 kiểu light khác là Point Light, Spot Light và Area light, có thể thay bằng các kiểu khác đó để so sánh, lựa chọn loại light phù hợp nhất cho style của game.

### Thêm giá đỡ

Ta cần một đối tượng giá đỡ để đỡ người chơi điều khiển hứng các quả rơi từ trên xuống.

Bấm Ctrl+Shift+N (phím tắt tạo đối tượng mới). Một đối tượng mới sẽ được tạo, bấm vào đối tượng trên hierarchy, bấm F2 để đổi tên thành Player:

Lúc này, cần đổi lại cách bố trí của Unity để dễ làm, sửa kiểu Tall thành kiểu 2 by 3.



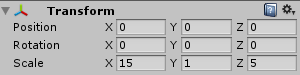
Ở giao diện mới ta sẽ thấy 2 phần rộng là Scene và Game. Scene là nơi thiết kế còn game là phần xem khi chơi game sẽ trông thế nào

Tạo một cube (xem phần trên). Đặt tên là model. Trong hierachy, kéo model vào đè lên player. Kết quả thu được như sau:

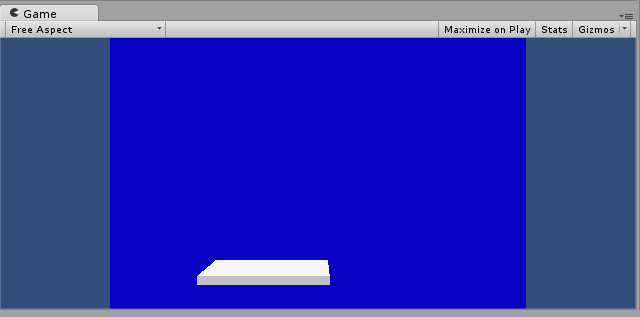


Ta vừa biến đối tượng model thành con của Player, thế có nghĩa là tọa độ của model giờ sẽ đổi theo tọa trục tọa độ mới lấy player là tâm (0,0,0), gọi là tọa độ tương đối. Khi đối tượng player thay đổi tọa độ, thì tất cả con của chúng sẽ di chuyển theo và tọa độ tương đối của các con không hề thay đổi. Trước mắt, việc tạo ra một đối tượng player rồi đưa model vào làm con chưa cần thiết nhưng đó sẽ là một thói quen tốt cho các project sau này.

Chọn đối tượng model (không phải player), chỉnh sửa các thông số như sau:



Trên phần Game, ta sẽ thấy:

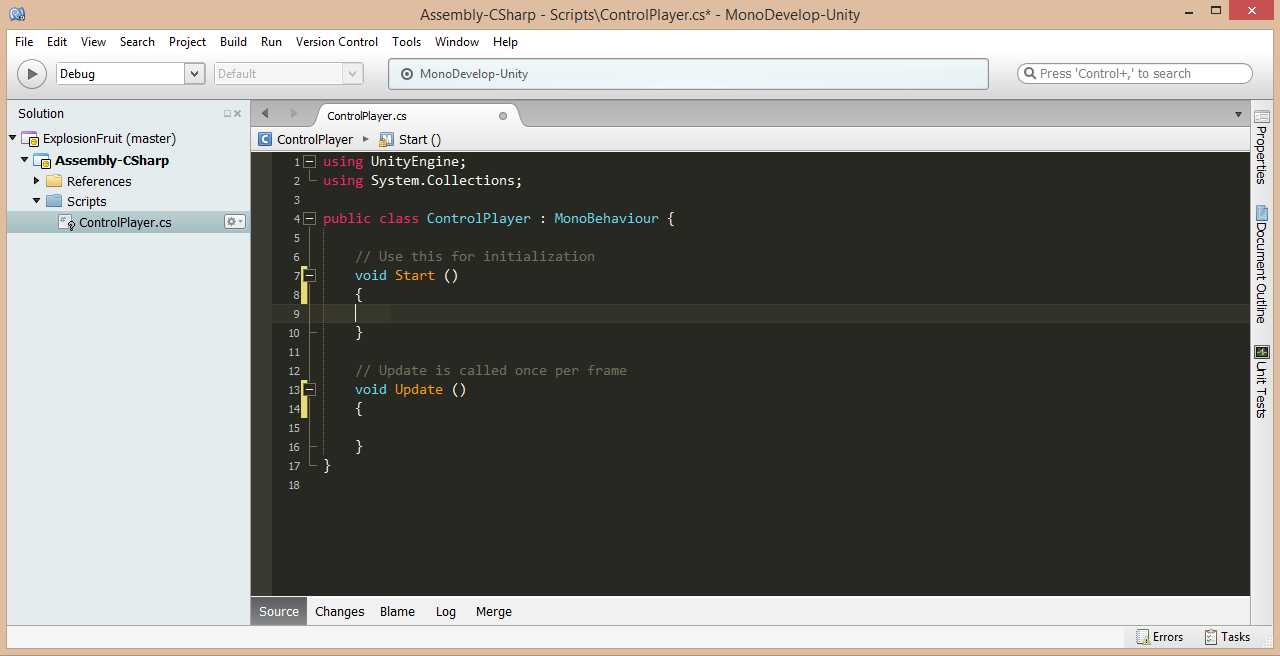


Vậy là ta đã có “diễn viên” chính, giờ viết kịch bản để “diễn”.

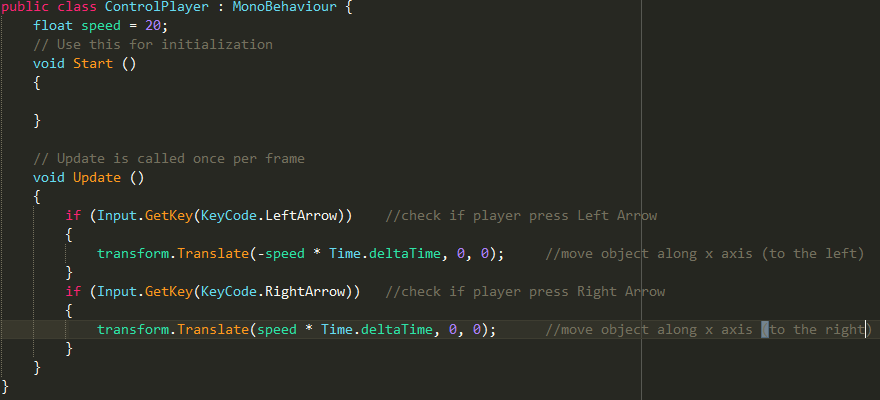
Chọn thư mục scripts, nháy phải vào phần bên cạnh chọn Create/C# Script

Đặt tên script mới được tạo ra là ControlPlayer.

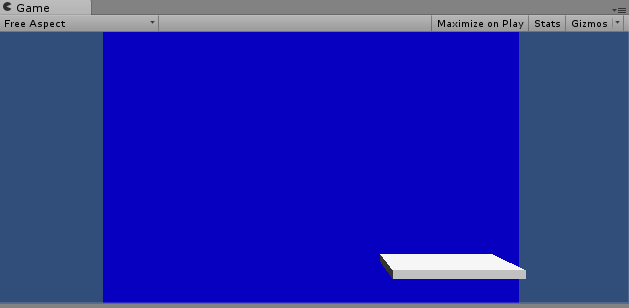
Nháy đúp vào scripts, trình soạn thảo MonoEditor sẽ được hiện ra (có thể sử dụng visual studio nếu đã cài sẵn).



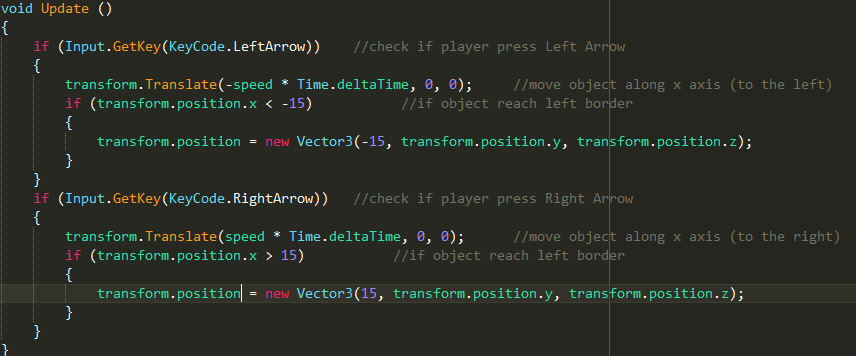
Một teamplate của file ControlPlayer có sẵn 2 hàm Start() và Update(). Hàm start sẽ được gọi 1 lần duy nhất khi đối tượng được khởi tạo. Hàm Update() sẽ gọi liên tục 60 lần/s để cập nhật các phần tính toán logic cho đối tượng (chỉ đạo diễn xuất). Ở đây ta không dùng hàm Start, viết trong hàm Update() nội dung như sau:



Quay trở lại Unity, chọn đối tượng Player, kéo file script vừa viết vào inspector của player. Chạy thử game, bấm 2 nút trái phải, nếu mọi thứ chính xác thì tấm màu trắng sẽ di chuyển theo điều khiển.



Một vấn đề là khi ta di chuyển ra hẳn bên ngoài thì khối kia cũng không bị giới hạn, giờ cần thêm một đoạn để giới hạn hai bên cho player. Sửa lại đoạn code như sau:



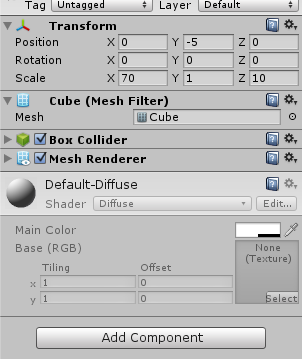
Ở đây không được dùng transform.position.x = 15 mà phải dùng như trong code mới. Cập nhật lại đoạn code chạy lại game sẽ thấy khối trắng bị giới hạn giữa 2 đường biên vô hình.

Phần tiếp theo sẽ thêm các loại hoa quả rơi tự do từ trên không xuống.

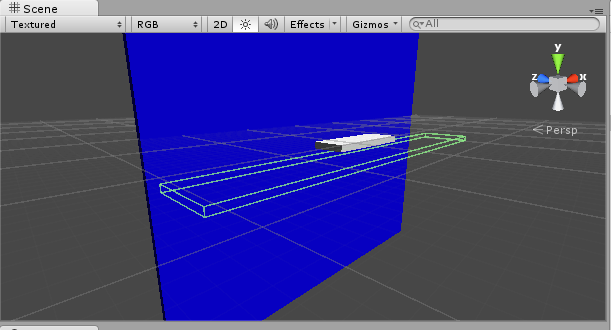
### Tạo hoa quả rơi xuống

Bấm Edit/Project Settings/Tag and Layers. Nhìn bên khung Inspector, xổ phần tag ra gõ thêm vào Element “ground” và “player” (viết chính xác).

Tạo một khối hộp, đổi tên thành “ground” đặt các thông số transform như sau:

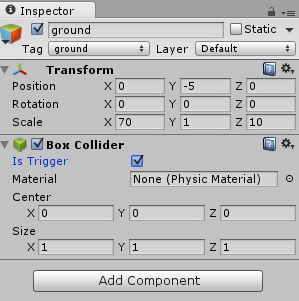


Nhìn trên hình ta sẽ thấy khối hộp này nằm ngay dưới player tạo ở phần trước. Khối hộp này có chức năng đỡ những quả rơi vào nó, nghĩa là người chơi đã ko đỡ được và quả rơi xuống từ đó tính số điểm và xác định gameover. Vẫn chọn khối ground, trong inspector, nháy phải vào mesh filter và mesh render chọn remove component. Vậy phần inspector chỉ còn transform và collider. Bấm vào mũi tên để hiển thị collider. Trong hình:



Hai thành phần mesh filter và mesh render dùng để hiển thị đối tượng lên màn hình đã bị remove, nhìn trên màn hình game ta sẽ không thấy đối tượng này.

Collider là thành phần tính toán vật lý có chức năng xác định va chạm vật lý rồi trả về sự kiện để developer tự xử lý trong script. Vẫn chọn đối tượng ground, trong phần tag, chọn “ground”, tick vào nút is trigger trong Box Collider. Sẽ giải thích sau ở phần script.



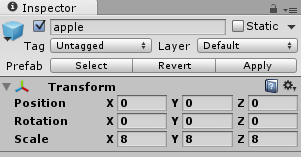
Tiếp theo, ta sẽ tạo quả táo bao gồm cả hình dạng (meshrender) và ứng xử (script).

Trong lập trình hướng đối tượng, ta biết lớp (class) có tính chất đóng gói. Nghĩa là đóng gói các thuộc tính, phương thức vào thành một lớp để khi sử dụng thì instance lớp đó ra. Unity hỗ trợ việc đóng gói một cách trực quan bằng các thao tác kéo thả. Các thành phần đóng gói trong Unity có tên là Prefab.

Bấm Ctrl+Shift+N để tạo một object mới, đặt tên là “pApple”. Trong thư mục Model, kéo hình apple từ trong đè lên object pApple mới tạo, kết quả là quả táo xuất hiện trên màn hình và là con của object apple.



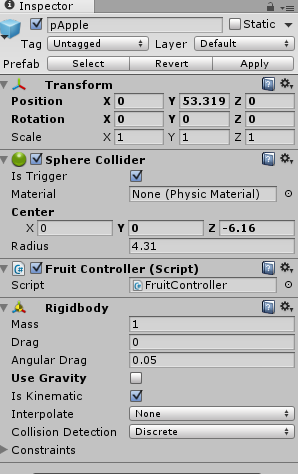
Chọn đối tượng apple, đặt các thuộc tính transform như sau:



Nháy chọn đối tượng pApple, ở phần inspector, chọn Add Component, gõ tìm kiếm “sphere collider”. Rồi chọn Sphere Collider từ menu xổ xuống.

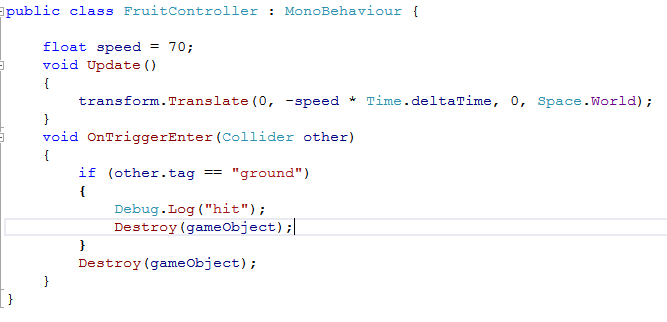
Làm tương tự add thêm thành phần RigidBody menu.

Điều chỉnh các thông số giống hình:



Giải thích các thành phần.

Sphere collider là đối tượng vật cản hình cầu, nó sẽ đại diện cho quả táo trong thế giới vật lý để tính toán. Điều chỉnh các thông số như trên để khối cầu sát với quả táo, sau này trong quá trình mô phỏng hiệu ứng vật lý, sẽ có cảm giác thật với quả táo. RigidBody là thành sẽ tìm hiểu sau này nhưng cần phải đưa vào. FruitController là đoạn script mà ta sẽ viết ngay sau đây. Tạo một file script FruitController.cs có nội dung như sau:

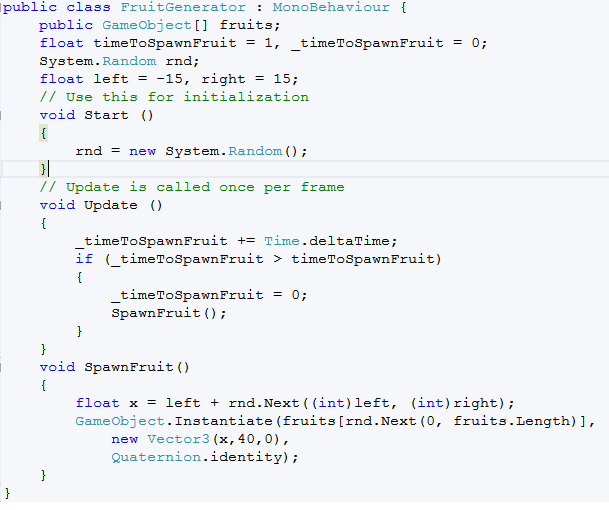


Hàm Update là hàm chạy liên tục 60 lần/s. lệnh translate sẽ dịch chuyển đối tượng theo các tham số truyền vào, như trong đoạn code ta thấy lệnh đó sẽ dịch quả táo theo trục y, tham số thứ 2 với hệ trục tọa độ chung (space.world). Nghĩa là khi vừa được tạo ra, quả táo sẽ có hành động duy nhất đó là di chuyển xuống dưới (rơi). Đó chính là kịch bản dành riêng cho quả táo.

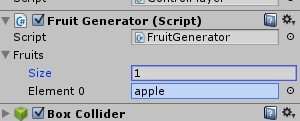
Hàm OnTriggerEnter() được gọi khi quả táo va chạm với một đối tượng vật lý khác (người chơi hoặc tấm chặn dưới). Hàm này sẽ kiểm tra tag của đối tượng mà nó va chạm, nếu là “ground”, ở phần trên ta đã đặt tag cho tấm đó là ground, thì sẽ log ra từ “hit”, rồi hủy bản thân, nếu không (chạm vào người chơi) thì đơn giản là hủy đối tượng. Sau này khi thêm phần tính điểm, đoạn này ta sẽ gọi thêm hàm cộng điểm, nhưng trước hết cần làm để xác định là mọi thứ đúng logic trước.

Code xong, và kéo file code vào đối tượng pApple. Kéo đối tượng pApple và thư mục Prefabs. Unity sẽ tự biến đối tượng đó thành prefab để sau này các đoạn code chỉ cần đưa prefabs đó ra, là tương đương với việc đưa mọi thành phần đã được sắp xếp sẵn trong đó ra.

Tạo file script FruitGenerator.cs. File này có chức năng sinh quả táo ở các vị trí ngẫu nhiên. Trong file, code đoạn script sau:



Sau mỗi khoảng thời gian “timeToSpawnFruit” script sẽ gọi SpawnFruit() để sinh một đối tượng ra. Chú ý phần khai báo public GameObject[] fruits; Việc khai báo kiểu public sẽ cho phép devloper cấu hình các biến bằng tay trực tiếp trên inspector. Code xong kéo file này vào đối tượng Player. Nhìn bên inspector, trong component Fruit Generator, xổ mục fruits, nhập 1 vào phần Size. Sau đó kéo prefab pApple vào Element.



Đến đây game đã có thể có những chức năng cơ bản, điều khiển giá đỡ, những quả táo rơi từ trên xuống. Số lượng những quả táo sẽ không bao giờ bị sinh ra vô hạn vì khi rơi xuống dù người chơi có đỡ được không thì nó cũng bị destroy. Trong phần hướng dẫn NGUI ta sẽ tạo phần giao diện cho game.