Tài liệu học Unity 3D

Unity là engine mạnh trong lập trình game 3d, và mô phỏng hệ thống 3d, đồng thời Unity hỗ trợ xuất ra trên mọi hệ điều hành di động với các phiên bản khác nhau. Nhóm phát triển game, và đào tạo lập trình viên unity và đồ họa 3d tại 3dvietpro giới thiệu với các bạn bước đầu tiên làm quen với Unity:

I. Cài đặt và sử dụng các đối tượng trong Unity

Để cài đặt các bạn vào: http://unity3d.com/ để download phiên bản free

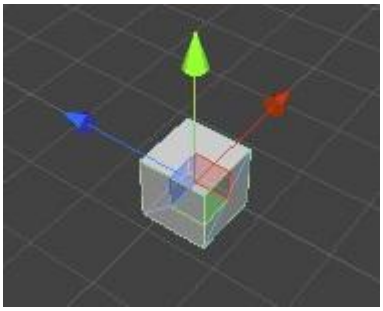
Sau khi cài dặt xong các bạn mở chương trình unity và chú ý các hình sau:

Hình:

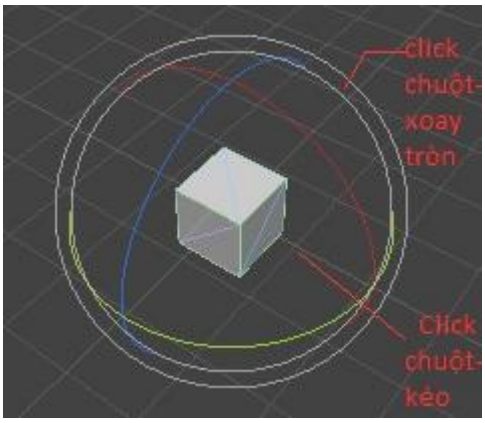
+ Scene: là giao diện thiết kế game unity
+ Game: là giao diện hiển thị chương trình game unity

Hình:

Tương ứng với 4 phím (Q,W,E,R), trong đó:
+ Q có biểu tượng bàn tay giúp bạn dịch chuyển vị trí màn hình
+ W dùng để lựa chọn các đối tượng, dịch chuyển theo 3 hướng x (*màu đỏ*), y(*xanh lam*), z(*xanh da trời*). Bạn muốn đối tượng dịch chuyển theo 3 chiều khác nhau chỉ việc click vào mũi tên và kéo theo chiều đó.

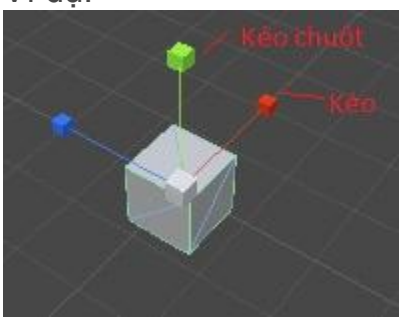


+ E dùng để xoay các đối tượng theo các hướng khác nhau

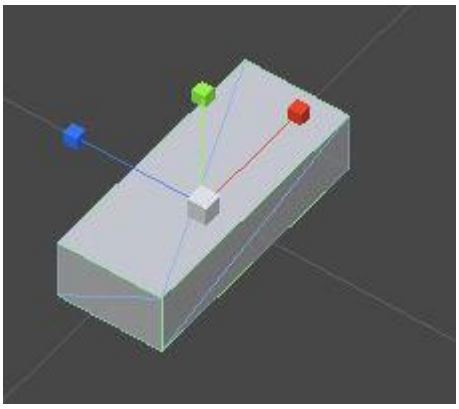


+ Phím R đùng để kéo đối tượng to hơn hoặc nhỏ hơn so với kích thước

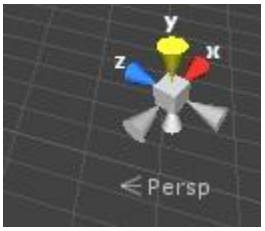
Ví dụ:



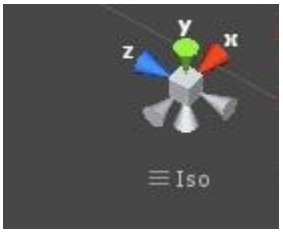
Thành



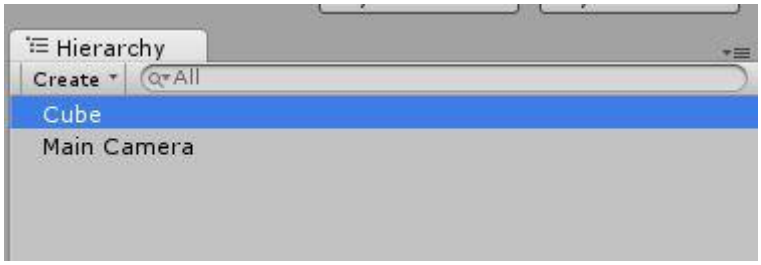
Các bạn làm quen với các trục tọa độ như hình dưới:



Ở đây là tọa độ theo 3d, nếu bạn muốn chuyển về tọa độ 2d theo chiều chọn bạn chị việc nhấn vào từ “Persp” phía dưới thành.

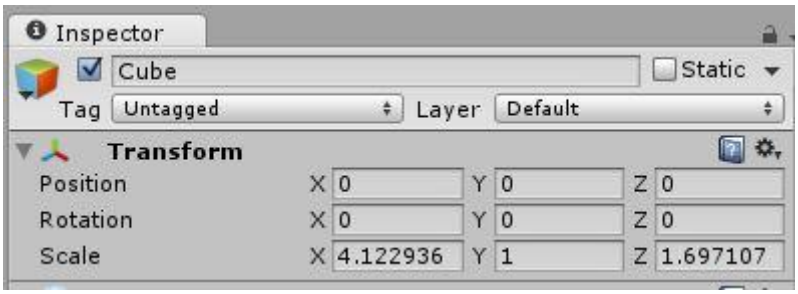


Giao diện tiếp theo của Unity đó là Hierarchy: dùng để chứa các đối tượng cần lập trình trong Unity.



Bạn có thể tạo ra đối tượng bằng cách nhấn vào Create, ở trên màn hình đang chỉ vào đối tượng “Cube”. Muốn tìm đối tượng này trong màn hình thiết kế (Scene) bạn chỉ cần nhấn phím “F”.

Giao diện Inspector:

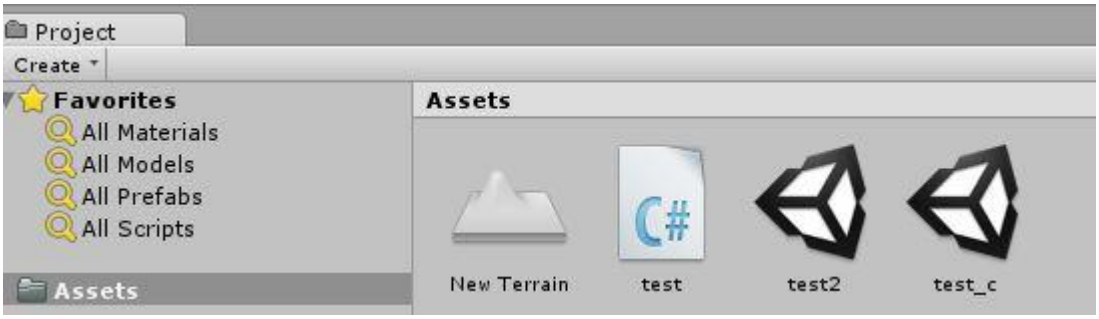


Đó chính là thuộc tính của đối tượng unity, với hình trên trong bài này 3dvietpro chỉ cho các bạn biết về:

+ Position: vị trí của đối tượng theo 3 chiều x, y, z mặc định là 0
+ Rotation: Vị trí xoay của đối tượng theo 3 chiều x, y, z, mặc định là 0
+ Scale: phóng to thu nhỏ đối tương theo các chiều x, y, z

Nếu bạn thay đổi lộn xộn các vị trí các đối tượng unity bạn chỉ cần nhấn vào biểu tường bánh răng bên phía tay phải chọn “reset”, tức là đối tượng sẽ trở về trạng thái ban đầu.

Giao diện Project:



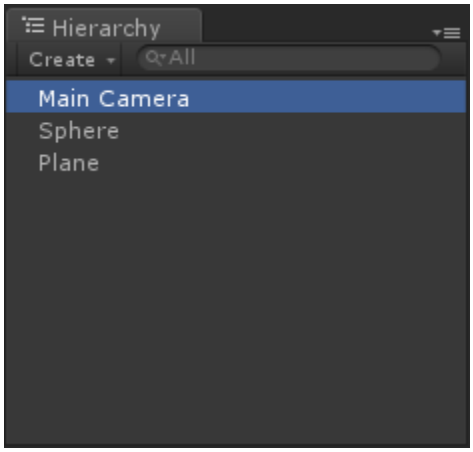
Dùng để chứa các file lập trình, âm thanh, hình ảnh, hiệu ứng, texture trong unity.

Hướng dẫn làm đối tượng di chuyển và nhảy bằng unity.

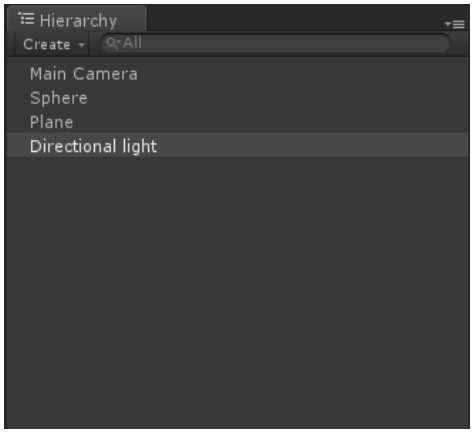
Hôm nay mình sẽ hướng dẫn các bạn mới học unity một bài khá đơn giản và cơ bản trong viêc học unity, đó là điều khiển một trái bóng di chuyển các hướng và nhảy lên bằng bàn phím.

Bước 1: các bạn tạo ra 1 quả bóng và 1 mặt phẳng để quả bóng có thể di chuyển trên đó

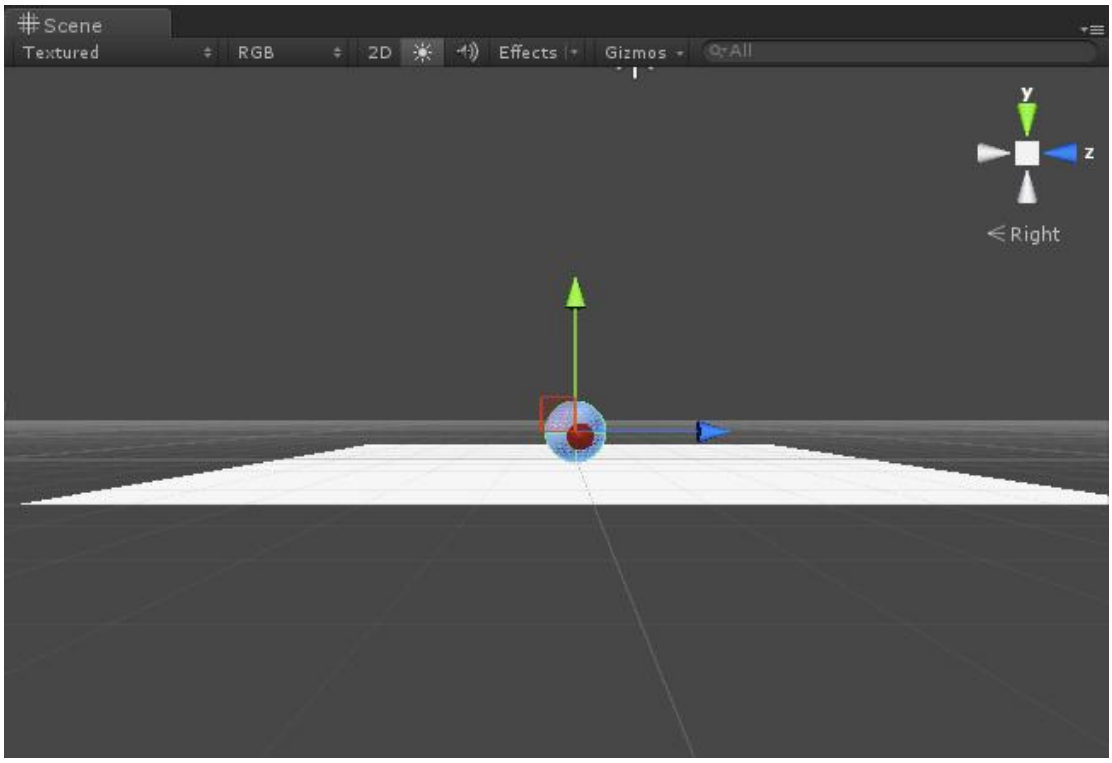
Các bạn nhìn trên thẻ Hierarchi có Create, các bạn vào đó và lần lượt chọn Sphere(khối cầu) và Plane (mặt phẳng).



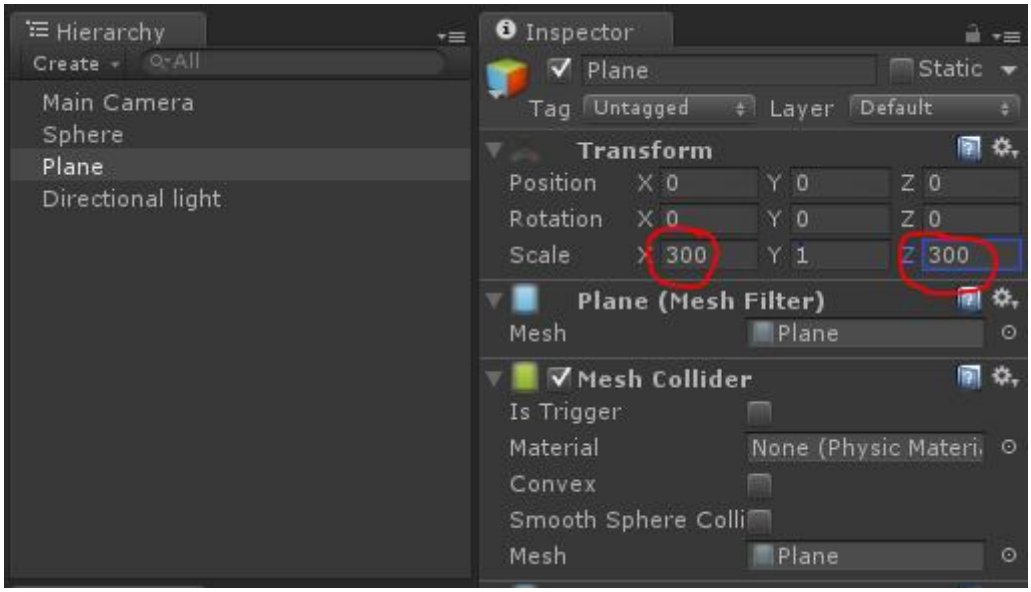
Các bạn có thể Create thêm Direction light để không gian sáng hơn .



Các bạn kéo trục tọa độ Y để quả cầu nằm trên mặt phẳng.

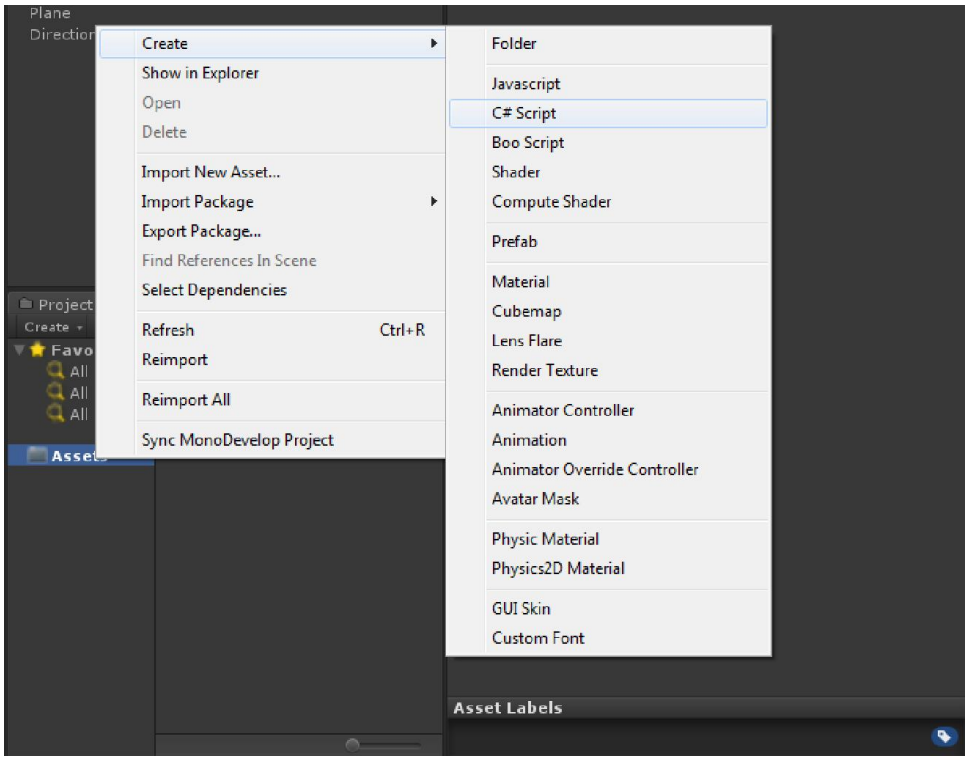


Sau đó các bạn kéo chiều dài và chiều rộng của mặt phẳng ra lớn hơn để quả bóng có thể di chuyển thoải mái bằng cách chỉnh thông số Scale của Plane trong thẻ Inspector.



Vậy là các bước chuẩn bị đã hoàn thành.

Bước 2: Để quả bóng di chuyển được, chúng ta cần viết script cho nó, các bạn nhìn trên thẻ Project, chuột phải vào Assets -> Create -> C# Script (ở đây mình dung ngôn ngữ C#).



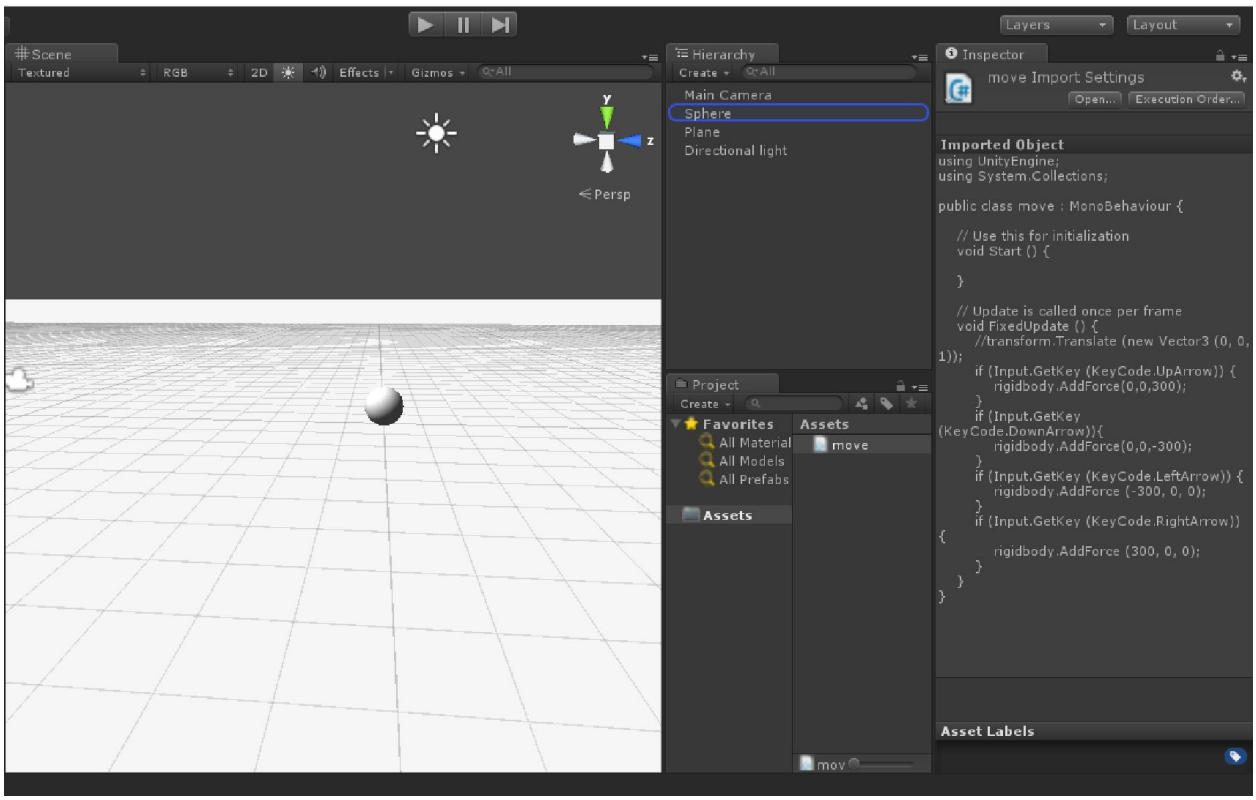
Một Script mới sẽ được tạo ra, các bạn đặt tên cho nó. Mình đặt tên là “move”.Sau đó các bạn nháy đúp vào script, cửa sổ MonoDevelop sẽ được mở ra để chúng ra code trong đó.

Các bạn code như sau:

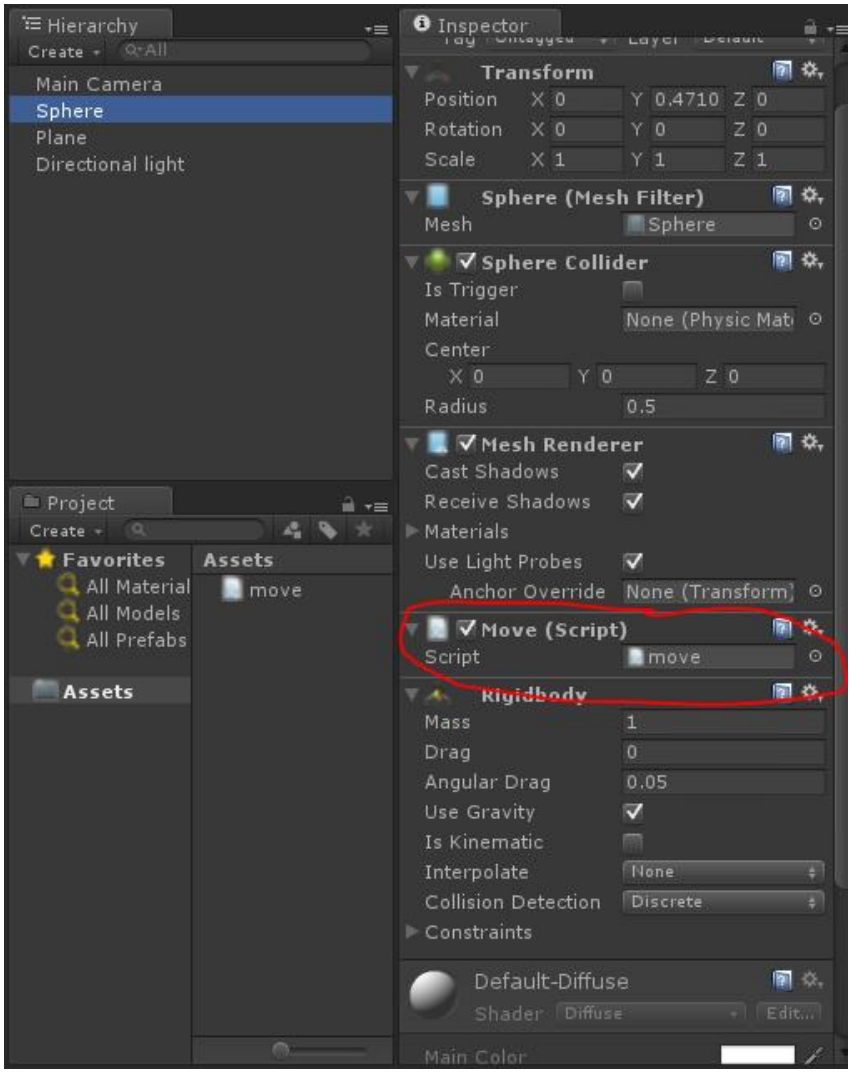

```
using UnityEngine;
using System.Collections;

public class move : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void FixedUpdate () {
        //transform.Translate (new Vector3 (0, 0, 1));
        if (Input.GetKey (KeyCode.Space)){
            rigidbody.AddForce(0,50,0);
        }
        if (Input.GetKey (KeyCode.UpArrow)) {
            rigidbody.AddForce(0,0,35);
        }
        if (Input.GetKey (KeyCode.DownArrow)){
            rigidbody.AddForce(0,0,-50);
        }
        if (Input.GetKey (KeyCode.LeftArrow)) {
            rigidbody.AddForce (-50, 0, 0);
        }
        if (Input.GetKey (KeyCode.RightArrow)) {
            rigidbody.AddForce (50, 0, 0);
        }
    }
}
```

Câu lệnh “rigidbody.Addfore” giúp ta đẩy quả bóng đó đi với 1 lực theo 1 vector nào đó.

Sau khi code xong các bạn ấn Ctrl+S để lưu lại rồi quay lại cửa sổ Unity, dùng chuột kéo file script (của mình là file move) vào đối tương Sphere (quả bóng)

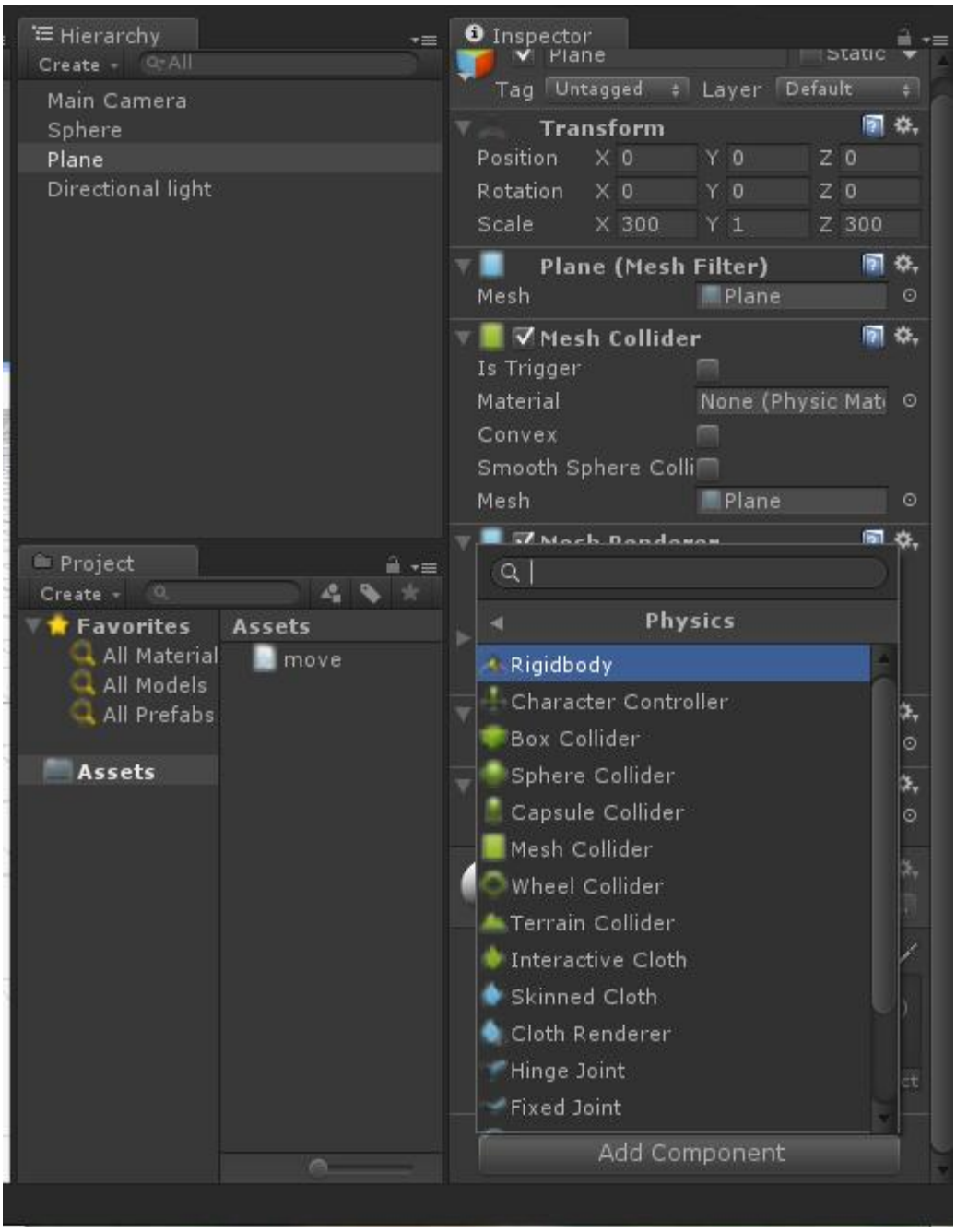


Bây giờ thì đối tượng Sphere đã được điều khiển bằng script “move”.

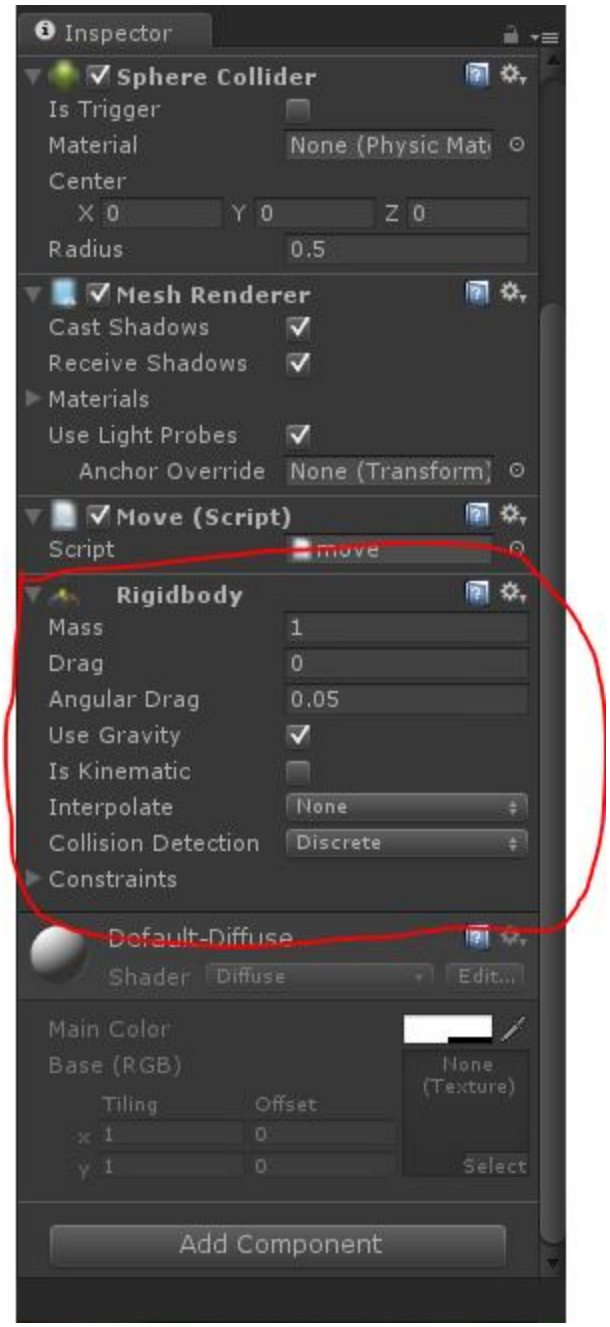


Vậy là ta đã xong phần viết lệnh điều khiển trái bóng.

Bước 3. Để quả bóng có thể nhận lực tác dụng vào thì nó phải có khối lượng, để tạo khối lượng cho quả bóng, các bạn chọn vào “Add component” trong thẻ Inspector của Sphere. Các bạn chọn Physics -> Rigidbody.



Bây giờ các bạn có thể tăng giảm khối lượng quả bóng.

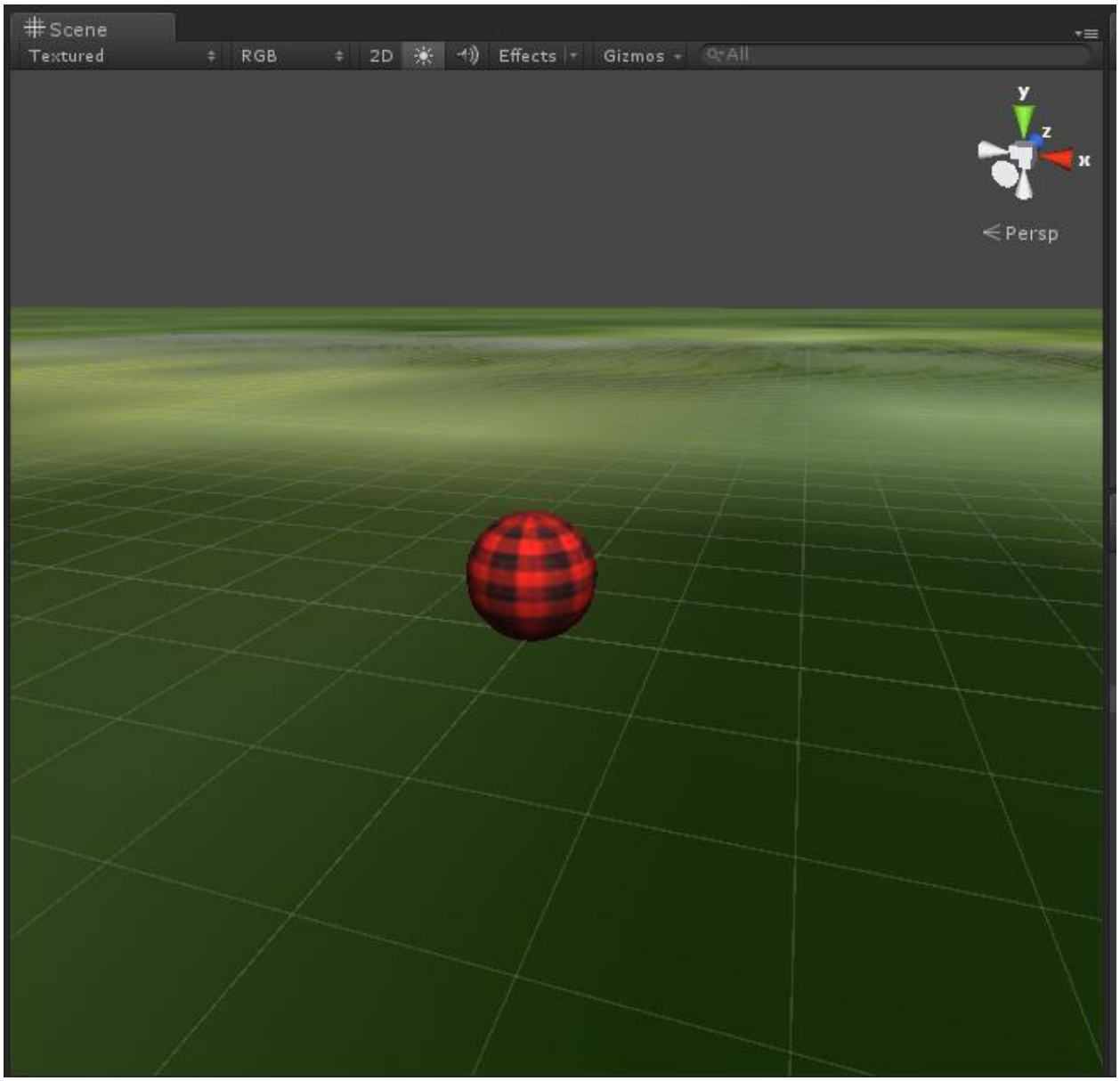


Bây giờ bạn thử ấn play và sử dụng phím lên xuống trái phải để di chuyển quả bóng, phím cách để nhảy :D

Bước 4: Để quả bóng đẹp hơn thì ta nên tạo cho nó 1 màu sắc hay họa tiết gì đó.

Các bạn lấy ảnh của 1 họa tiết bất kì mà bạn muốn tô cho quả bóng, sau đó kéo nó vào ô Assets của thẻ Project. Mình chọn họa tiết caro.

Sau đó bạn dùng chuột kéo file đó thả vào quả bóng, vậy là quả bóng đã được tô màu, các bạn có thể làm tương tự với Plane để tạo mặt đất đẹp hơn.



Vậy là các bạn đã hoàn thiện việc điều khiển quả bóng chạy theo các hướng và nhảy.

2.3. Tạo đối tượng cha, đối tượng con (giải thích Position, locaposition), cách lấy đổi tượng con trong cha

Trong unity với các các bạn có thể tạo thành nhóm các đối tượng bằng cách sau:

Bước 1: Click đúp vào đối tượng cần tạo đối tượng

Lưu ý: Hành động click đúp giúp bạn chọn đúng đối tượng đó, và thao tác tại vị trí đối tượng được chọn.

Hình 2.3.1. Click đúp vào đối tượng ParentCam

Bước 2: Nhấp phải chuột chọn một đối tượng, như hình 2.3.1 tôi đã tạo đối tượng con là Main Camera.

Bước 3: Thực hiện lệnh sau:

```
// Khai báo biến
protected Transform pivot;
public Transform cam;

// Thực hiện trong start
```

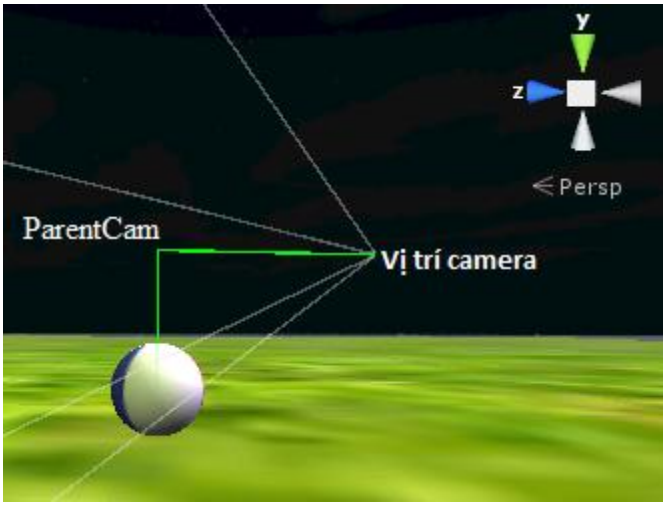
```
cam = GetComponentInChildren<Camera> ().transform;
pivot = cam.parent;
```

với dòng *"GetComponentInChildren<Camera> ().transform;"* để lấy đối tượng camera, và từ đối tượng camera chúng ta lấy được đối tượng cha của nó là ParentCam.

Bước 4: Kẻ đường kẻ màu xanh từ đối tượng cha tới đối tượng con

```
void OnDrawGizmos()
{
    if (pivot != null && cam != null)
    {
        Gizmos.color = Color.green;
        Gizmos.DrawLine(pivot.position,
cam.transform.position);
    }
}
```

Như hình:



Hình 2.3.2. Vị trí từ điểm cha tới điểm con

Tiếp theo chúng ta tìm hiểu về **Position, LocalPosition**, các bạn để ý với đối tượng "ParentCam" ở trên thì câu lệnh "transform.Position" thể hiện vị trí của đối tượng trong toàn bộ không gian 3D

Ví dụ:

```
// thực hiện di chuyển đối tượng tới tọa độ (0, 0, 0)
transform.position = Vector3(0, 0, 0);
```

Với LocalPosition có điều đặc biệt, đó là chúng ta dùng để xác định vị trí của đối tượng đó trong không gian cha của đối tượng.

Ví dụ trong đối tượng “Camera” ở hình 2.3.2 thì:

Hình 2.3.3. Vị trí của Camera trong đối tượng cha “ParentCam”

Như vậy chúng ta đã hiểu được vị trí của đối tượng cha và con, bây giờ các bạn thử làm một bài tập ngược lại như sau:

Bài 2.3.1: Từ vị trí cha lấy tên của đối tượng con, cụ thể trong ví dụ 2.3.1 thì từ vị trí **ParentCam** các bạn tìm cách lấy tên của **Main Camera** hoặc bất kỳ một đối tượng nào đó trong **ParentCam**

2.8. Hàm xử lý thời gian

2.8.1. **Sử dụng timescale**

Trong quá trình xử lý game unity bạn không thể thiếu hàm xử lý thời gian, nhóm 3dvietpro giới thiệu các bạn xử lý thời gian bằng hàm timeScale

Bước 1: Các bạn tạo một cube có gắn rigidbody, và file lệnh có các dòng lênh bước 2

Bước 2: Code hiển thị

```
// Update is called once per frame
void Update () {

   //Nhấn nút Q để thực hiện timeScale
   if (Input.GetKeyDown(KeyCode.Q))
   {
       if (statusBol == false)
       {
           Time.timeScale = 0.7f;
           statusBol = true;
       }
       else if (statusBol == true)
       {
```

```
            Time.timeScale = 0.1f;
            statusBol = false;

        }

        Time.fixedDeltaTime = 0.02f * Time.timeScale;
        print("fixedDeltaTime:" + Time.fixedDeltaTime);
    }
    // Nhấn phím Space để di chuyển theo trục y
    if (Input.GetKeyDown(KeyCode.Space))
    {
        transform.Translate(0,Time.deltaTime * Speed,0);
    }

}
```

Khi bạn nhấn nút space khối box sẽ di chuyển theo trục y, trong lúc khối box rơi xuống bạn nhấn phím Q, bạn sẽ thấy khối box di chuyển chậm dần, như vậy hàm timeScale đã có tác dụng:

Với Time.timeScale = 0.1f; // là thời gian thực hiện tại
Với Time.timeScale = 0f; // Thời gian sẽ dừng lại
Với Time.timeScale = 0.xxxxf; // Thời gian sẽ chậm lại 2x so với thời gian thực

Tác dụng của việc sử dụng timeScale gồm: pause game, sử dụng trong hiệu ứng và di chuyển

2.8.1. Sử dụng timescale để thực hiện pause một game

```
void PauseGame (bool val)
{
	if (val == false)
	return;
	if (Time.timeScale == 0f) {
			Time.timeScale = 1f;
			pause = false;
			return;
		} else {
			Time.timeScale = 0f;
			pause = true;
			return;
	}
}
```

Với hàm trên chúng ta thực hiện ép vào đối tượng và gọi PauseGame(true) thông qua một nút lệnh thì đối tượng sẽ dừng lại do thiết lập timescale =0f, và bấm nút lại đối tượng tiếp tục được di chuyển.

2.9. Lập trình AI & hệ tọa độ

Với ví dụ sau các bạn có thể định hướng đường đi của một đối tượng qua các điểm được đánh dấu cho trước bằng cách sử dụng thuật toán AI:

Bước 1: Tạo 1 file ".js" ép vào đối tượng cần di chuyển

```
var spawnPoint:Transform;
//Mảng bao gồm các phần tử điểm được đánh dấu
private var waypoint=new Array();
private var dau:float;
private var j:int=0;
var Gtext:GUIText;
private var veloc;
private var ktr:boolean;
private var t:float=0.0;

var vt: float = 5f;

function Start(){
     var i=0;
    //Lấy tất cả các điểm con trong điểm cha
     for(var child:Transform in spawnPoint){
          waypoint[i]=child;
          i++;
```

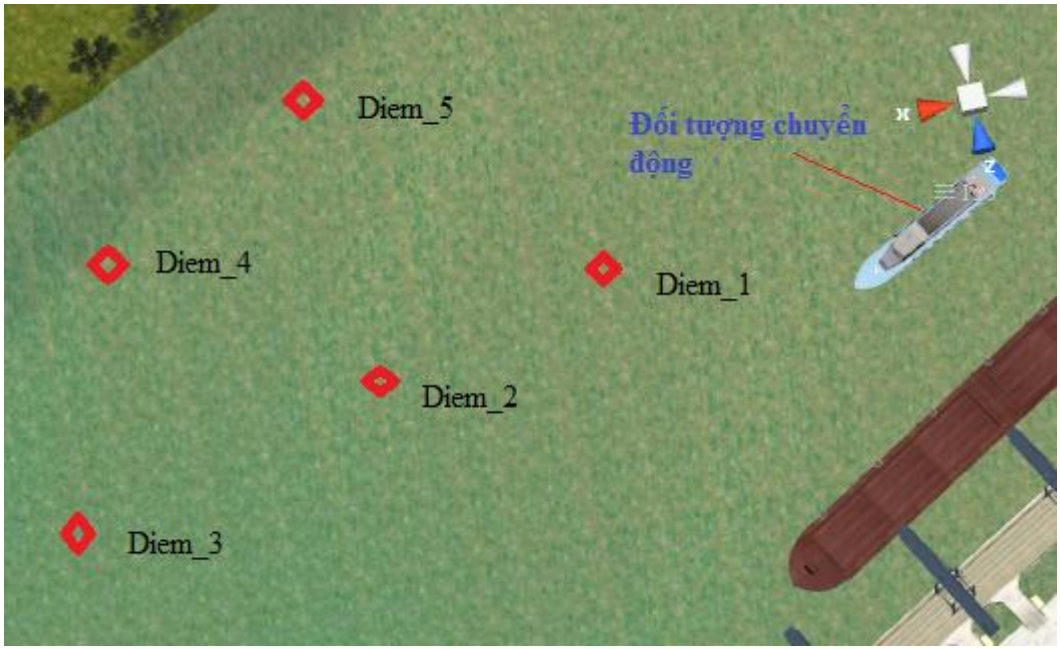
```
    }
}

function Update () {
    //lấy vị trí từ điểm đầu tới các điểm được đánh dấu
     dau=Vector3.Distance(transform.position,waypoint[j].position
);

     var relvitePos=waypoint[j].position-transform.position;
     var gtr :Vector3;
    //Xoay đối tượng tới vị trí điểm tìm thấy
     var rotation=Quaternion.LookRotation(relvitePos);
     transform.rotation=Quaternion.Slerp(transform.rotation,rotat
ion,Time.deltaTime*2);
    //Dịch chuyển đối tượng
     transform.Translate(Vector3.forward *Time.deltaTime * vt);
    //Lấy vị trí tiếp theo của điểm cần dịch tới
     if(dau<=1){
          j++;
     }
     //Kiểm tra xem điểm đã đã đánh dấu đã tới điểm cuối cùng
chưa thì quay lại điểm 0
     if(j>=waypoint.length){
          j=0;
     }
```

}

Bước 2: Tạo các điểm được đánh dấu

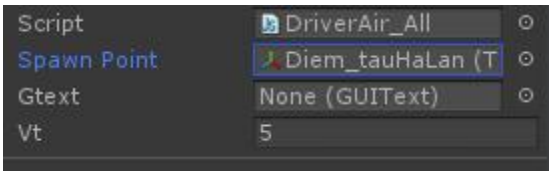
Hình 2.9.1. Tạo điểm định vị cho đối tượng chuyển động

Bước 3: Sắp xếp đặt điểm cho đối tượng cần di chuyển tới



Hình 2.9.2. Đánh dấu các điểm

Bước 4: Ép các điểm được đánh dấu vào mảng “waypoint” được khai báo ở trên



Hình 2.9.3. Ép điểm cha chứa các điểm đánh dấu vào tệp lệnh

Bước 5: Chạy chương trình

Thế vậy qua ví dụ này các bạn đã tạo được chương trình AI chuyển động cho bất kỳ đối tượng nào trên màn hình. Các bạn có thể phát triển thêm ý tưởng qua bài tập này nhé:

Bài 2.9. Thực hiện va chạm các đối tượng theo lệnh AI như trên, dừng lại di chuyển khi đối tượng va chạm và tạo hiệu ứng va chạm.

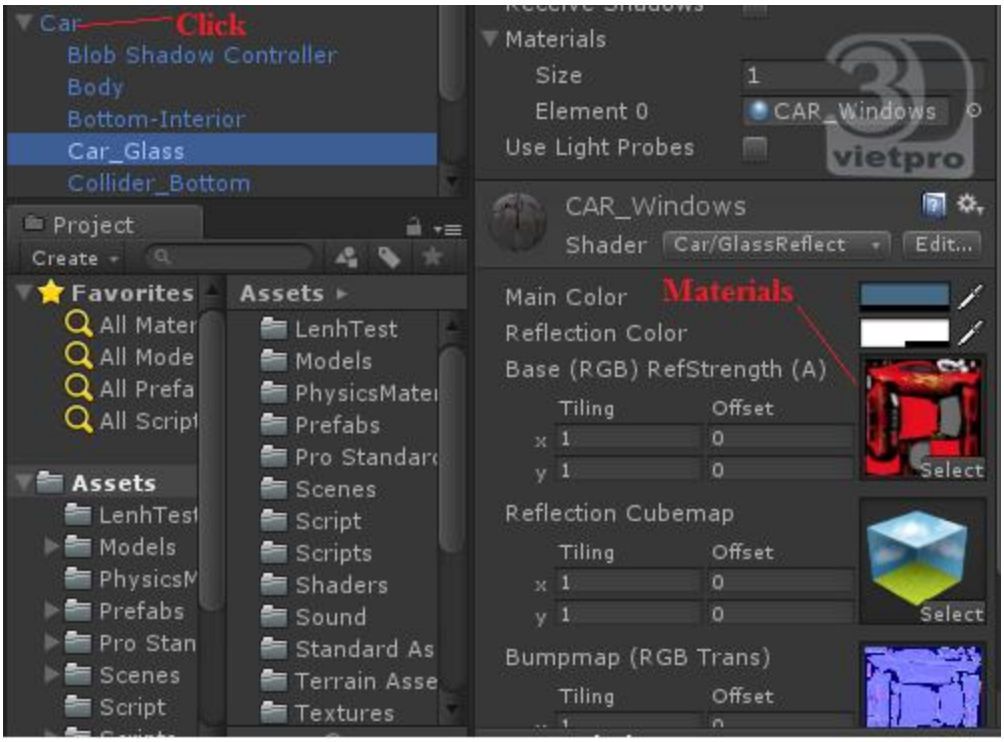
3. Hướng dẫn làm chuyển động Oto 3D

Chương trình làm ô tô 3D chuyển động được ứng dụng nhiều trong các game đua xe 3D, ngoài ra làm chuyển động ô tô 3d trên địa hình được ứng dụng vào trong lĩnh vực mô phỏng như các bài tập lái xe, kiểm tra các thiết bị của động cơ. Trong phần này nhóm 3DVietpro trình bày với các bạn các bước tạo một ô tô 3D chuyển động.

Bước 1: Các bạn vào phần Asset của unity tìm kiếm từ khóa sau “Toturial car” và download về máy của mình và chạy chương trình:

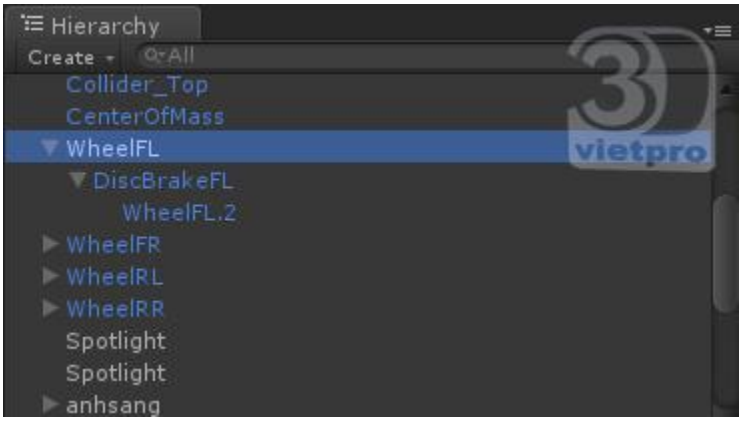
Hình 4.1.1. Giao diện chương trình đua xe ô tô được import từ Asset Unity

Bước 2: Các bạn tách mô hình Oto trong hình 4.1 ra một project riêng, để thực hiện tách các bạn lưu ý vấn đề sau:
- Vào thư mục gốc của project tìm đến \Assets\Models\Car
- Xuất hiện file “Materials” và “catamount.fbx”, các bạn thực hiện kéo vào cửa sổ Project của project mới tạo. Trong đó “Materials” là map của ô tô, còn fbx là mô hình 3D của ô tô.
Bước 3: Khi thực hiện kéo ô tô hiển thị không có màu sắc, các bạn click vào các phần của Oto sẽ hiện hình như dưới:



Hình 4.1.2. Thành phần của ô tô được gắn map

Nếu phần “Material” trong hình trên không có, thì các bạn tìm trong thư mục Materials vừa Import để phù hợp.



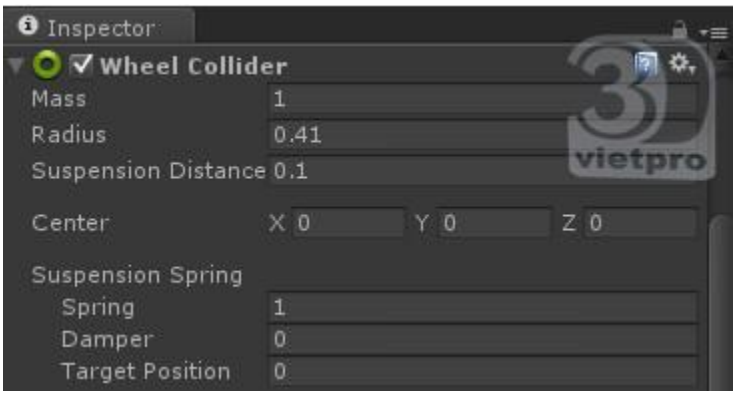
Hình 4.1.3. Lựa chọn bánh xe phía trước bên tay trái

Hình 4.1.4. Tách bánh xe

Bước 4: Click đúp vào bánh xe, sau đó vào Game Object chọn Create Empty, khi chọn xong bạn sẽ thấy xuất hiện một điểm nằm ở trung tâm bánh xe và đó chính là điểm Game Object vừa rồi bạn tạo.

Bước 5: Đặt tên cho đối tượng Game Object trong Bước 4 , như hình 4.1.3. Chúng ta đặt là WheelFL_col

Bước 6: Click vào “WheelFL_col” trong phần Inspector chọn Add Component tìm đến Wheel Collider, khi bạn lựa chọn sẽ hiển thị như sau:



Hình 4.1.6. Wheel Collider được lựa chọn

Hình 4.1.7. Hình sau khi được lựa chọn xong, vạch màu xanh chính là Wheel Collider

Bước 7: Bạn thực hiện tương tự như các bước 4,5,6 với các bánh xe còn lại

Bước 8: Viết lênh cho bánh xe chuyển động bằng cách tạo ra một file C#, sau đó bạn kéo vào chiếc xe.

- Chúng ta khai báo dòng lệnh sau là Public

```
// Biến để thực hiện làm chuyển động bánh xe
public Transform WheelFL;
public Transform WheelFR;
public Transform WheelRL;
public Transform WheelRR;

//Biến để thực hiện làm chuyển động Collider
public WheelCollider WheelFL_col;
```

```
public WheelCollider WheelFR_col;
public WheelCollider WheelRL_col;
public WheelCollider WheelRR_col;
```

- Khai báo tốc độ và góc quay bánh trước của xe

```
public float speed =40f;
public float angle=60f;
```

- Để thực hiện làm chuyển động, như thực tế đã cho thấy là chúng ta phải dùng lực đẩy 2 bánh sau đi và bánh trước chỉ có nhiệm vụ quay góc làm bánh lái cho xe chạy theo các hướng được điều khiển. Các bạn viết hàm sau trong Update:

```
WheelRL_col.motorTorque = speed * Input.GetAxis("Vertical");
WheelRR_col.motorTorque = speed * Input.GetAxis("Vertical");

//WheelFL_col.motorTorque = speed * Input.GetAxis("Vertical");
//WheelFR_col.motorTorque = speed * Input.GetAxis("Vertical");
```

- Tạo góc quay cho bánh trước

```
WheelFL_col.steerAngle = angle * Input.GetAxis ("Horizontal");
WheelFR_col.steerAngle = angle * Input.GetAxis ("Horizontal");
```

- Trong Update các bạn viết thêm hàm sau để quay bánh xe:

```
WheelRL.Rotate (speed*Input.GetAxis("Vertical"),0,0);
WheelRR.Rotate (speed* Input.GetAxis("Vertical"),0,0);
WheelFL.Rotate (speed*Input.GetAxis("Vertical"),0,0);
WheelFR.Rotate (speed * Input.GetAxis ("Vertical"), 0, 0);
```

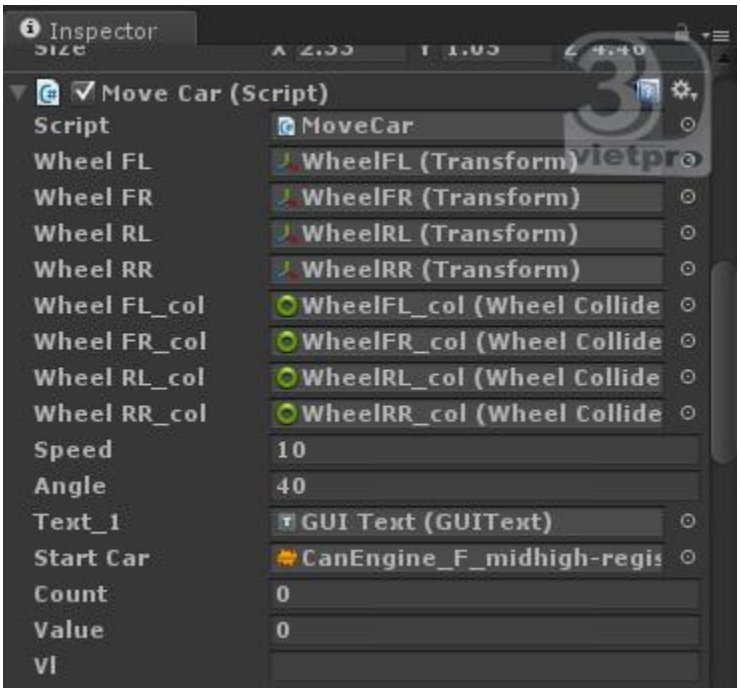
Và để chính xác với tốc độ quay chúng ta có thể thay biến speed bằng dòng lệnh sau:

“WheelFL_col.rpm/60*360”

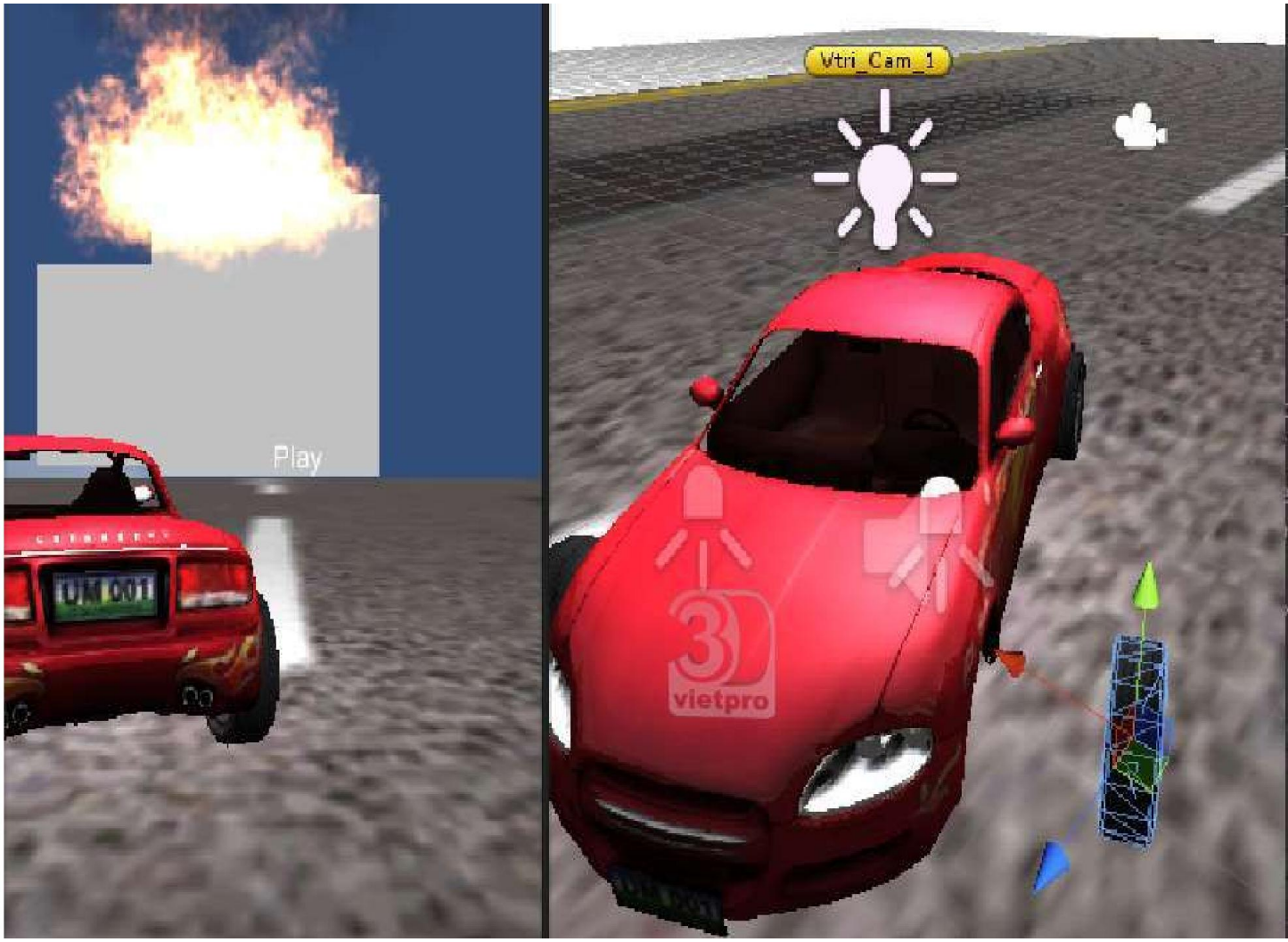
- Để bánh trước có thể làm bánh lái chúng ta dùng dòng lệnh sau:

```
WheelFL.localEulerAngles = new Vector3(WheelFL.localEulerAngles.x,
angle * Input.GetAxis("Horizontal"), WheelFL.localEulerAngles.z);

WheelFR.localEulerAngles = new Vector3(WheelFL.localEulerAngles.x,
angle * Input.GetAxis("Horizontal"), WheelFL.localEulerAngles.z);
```

Thế là chúng ta đã viết lệnh chuyển động cho bánh xe, bây giờ các bạn chỉ cần gán các đối tượng Public cho các biến được khai báo và chạy chương trình:



Hình 4.1.8. Kéo các đối tượng vào các biến của xe 3D



Bước 9: Kết quả chạy

Như vậy trong phần này nhóm 3dvietpro đã hướng dẫn các bạn làm xe chuyển động bằng cách sử dụng Wheel Collider. Các bạn lưu ý một số thuộc tính sau của Wheel Collider:
1. motorTorque : Tạo động lực cho bánh xe
2. steerAngle : Tạo góc cho bánh

Để lấy vận tốc của xe các bạn thực hiện lệnh sau:

“rigidbody.velocity.magnitude * 3.6f”

Vận tốc * thời gian (3.6f) = Số km/h

4. Sử dụng và di chuyển camera

Các hàm thường sử dụng:

+ LookAt: Hướng camera theo vị trí cần nhìn

LookAt(Transform target, Vector3 worldUp = Vector3.up);

VD:

```
public Transform target;
void Update() {
    transform.LookAt(target);
}
```

+ fieldOfView: Điều chỉnh tầm nhìn của Camera, để thực hiện điều chỉnh, chúng ta sử dụng dòng lệnh sau:

Camera.main.fieldOfView = <Biến kiểu số>;

VD:

Camera.main.fieldOfView = <Biến kiểu số>;

+ LookRotation: Dùng để xoay góc nhìn của camera tới một góc nhìn khác theo vòng quay được tính toán.

VD:

```
public Transform target;
void Update() {
    //xác định hướng cần xoay tới
    Vector3 relativePos = target.position - transform.position;
    // sử dụng hàm LookRotation để đưa ra vòng cần quay
    Quaternion rotation = Quaternion.LookRotation(relativePos);
    // di chuyển camera theo vòng quay được tính
    transform.rotation = rotation;
}
```

+ Slerp: di chuyển góc nhìn từ “From” tới “to” với vận tốc được đưa ra

public static Quaternion Slerp(Quaternion from, Quaternion to, float t);

VD:

```
public Transform from;
 public Transform to;
 public float speed = 0.1F;
 void Update() {
    transform.rotation = Quaternion.Slerp(from.rotation, to.rotation, Time.time *
speed);
 }
```

Chương trình trình bày:

Đưa đối tượng vào unity, thực hiện gắn map, lập trình đối tượng chuyển động và di chuyển camera:

5. Sử dụng chạm màn hình

TouchCount

GetTouch(0).phase

TouchPhase

+ Began: Ngón tay đã chạm vào màn hình

+ Moved: Di chuyển ngón tay trên màn hình

+ Stationary: Ngón tay chạm vào màn hình và không thực hiện di chuyển

+ Ended: Ngón tay rời khỏi màn hình, chạm kết thúc

+ Canceled: Hệ thống thoát chế độ chạm

VD 1:

```
for (var i = 0; i < Input.touchCount; ++i) {
        if (Input.GetTouch(i).phase == TouchPhase.Began)
  }
```

VD 2:

```
 int nbTouches = Input.touchCount;


  if(nbTouches > 0)
  {
    for (int i = 0; i < nbTouches; i++)
    {
      Touch touch = Input.GetTouch(i);


      TouchPhase phase = touch.phase;


      switch(phase)
      {
        case TouchPhase.Began:
          print("New touch detected at position " + touch.position + " , index " +
touch.fingerId);
```

```
            break;
        case TouchPhase.Moved:
            print("Touch index " + touch.fingerId + " has moved by " + touch.deltaPosition);
            break;
        case TouchPhase.Stationary:
            print("Touch index " + touch.fingerId + " is stationary at position " +
touch.position);
            break;
        case TouchPhase.Ended:
            print("Touch index " + touch.fingerId + " ended at position " + touch.position);
            break;
        case TouchPhase.Canceled:
            print("Touch index " + touch.fingerId + " cancelled");
             break;
        }
    }
}
```

Tính tốc độ của chạm

Sử dụng detatime để tính toán và di chuyển tốc độ của ngón tay

```
float touchSpeed = touch.deltaPosition.magnitude / touch.deltaTime;
```

Lấy vị trí của màn hình

```
Vector3 position = Camera.main.ScreenToWorldPoint(touch.position);
```

tapCount

```
if (Input.touchCount > 0) {
    Debug.Log("ONE TAP");
    if(Input.GetTouch(0).tapCount == 2)
    Debug.Log("DOUBLE TAP");
 }
```

Input.acceleration

Sử dụng để lắc màn hình theo chiều đã định (Chiều x, y) tương tự khi chúng ta sử dụng **Horizontal** và **vertical** trong unity thay cho chiều x, y khi chúng ta dùng acceleration

Các sử dụng:

VD 1:

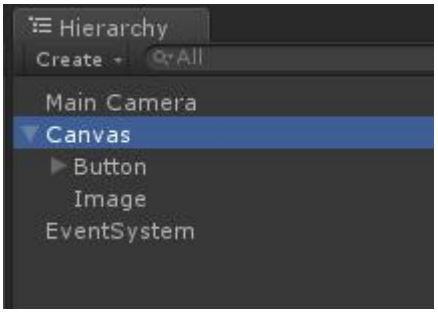
```
int tiltDirection = Input.acceleration.x > 0 ? 1 : -1;
thisTranfrom.position = new Vector3( Mathf.Clamp(thisPosition.x,-2.4f,2.4f),thisPosition.y,thisPosition.z);
```

với **Input.acceleration.x** các bạn lắc màn hình theo chiều ngang, và **Input.acceleration.y** dùng để lắc màn hình theo chiều dọc.

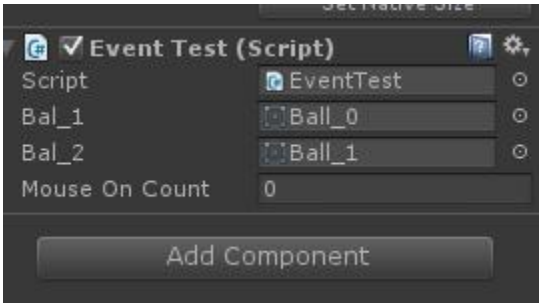
6. Sử dụng sự kiện cho UI

6.1. Sử dụng ép trực tiếp các sự kiện

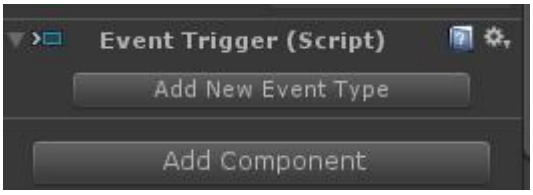
1. Tạo ra ra các đối tượng Button và Image nằm trong canvas



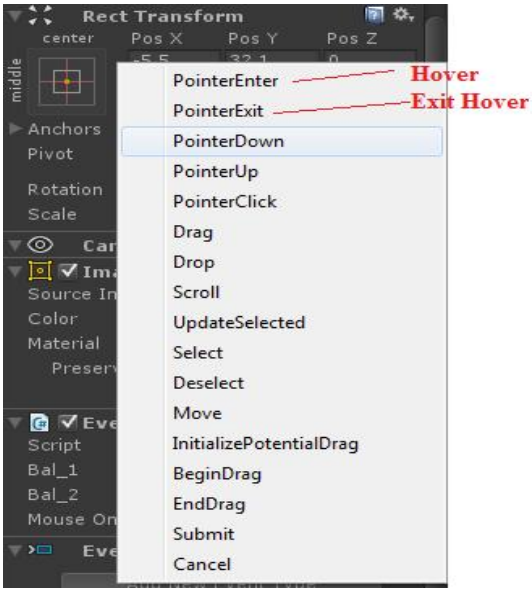
2. Đưa tệp lệnh vào đối tượng Image



3. Từ Component chọn Event Trigger



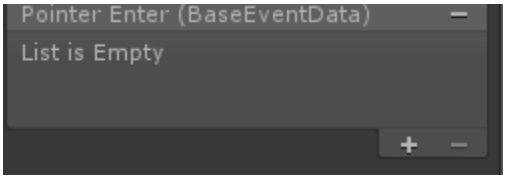
4. Chọn Add New Event Type> PointerEnter để thực hiện việc hover chuột, hay Chạm vào đối tượng



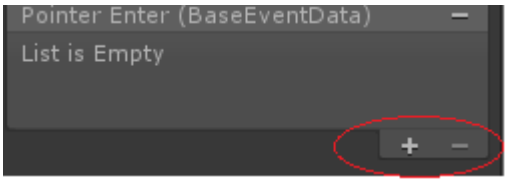
Trong đó:

PointerEnter: Kích hoạt khi chúng ta hover lên đối tượng

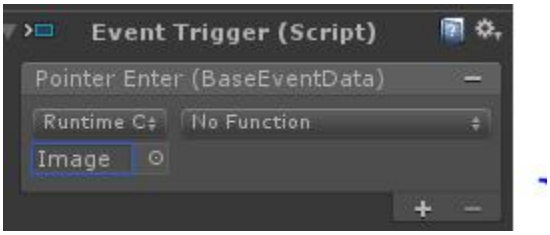
PointerExit: Kết thúc việc hover



5. Thêm sự kiện bằng cách clicks vào dấu (+), bỏ một sự kiện click vào dấu (-)



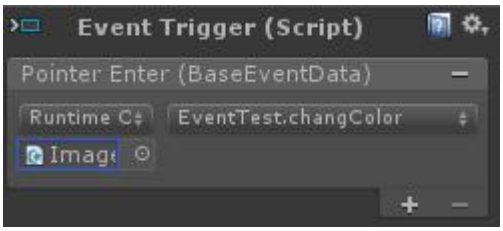
6. Kéo đối tượng Image có gắn tệp lệnh vào sự kiện



7. Viết lệnh code trên tệp lệnh

```
public void changColor()
{
    Debug.Log("1");
    //gameObject.GetComponent<Image>().sprite = bal_2;
}
```

8. Lựa chọn No Function trong phần 6 chọn phương thức ChangColor



Đó là phương pháp chúng ta xử lý trên đối tượng có gắn tệp lệnh để bắt các sự kiện, trong phần 6.2 các bạn biết thêm một phương pháp mới để sử dụng trực tiếp các sự kiện mà không cần thao tác bằng tay.

6.2. Sử dụng lệnh ép trực tiếp các sự kiện

Trong phần này các bạn tạo ra Tệp EventTest gắn vào đối tượng Image

1. Khai báo thư viện UnityEngine.EventSystem, thực hiện ép việc kế thừa đến lớp ,IPointerEnterHandler, IPointerExitHandler
Cho tệp các bạn cần bắt sự kiện

```
using UnityEngine.UI;
using UnityEngine.EventSystems;
public class EventTest : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler {
```

2. Trong phần khai báo các bạn khai báo như sau

```
private AudioClip audioclip;
private AudioSource audiosource;
```

3. Trong phần start chúng ta lấy đối tượng âm thanh

Trong đó `button_hover_sound` là tên file của âm thanh nằm trong thư mục “Resources”

```
void Start () {
        audioclip = Resources.Load<AudioClip>("button_hover_sound");
         gameObject.AddComponent<AudioSource>();
        audiosource = gameObject.GetComponent<AudioSource>();
}
```

3. Chúng ta viết ra 2 sự kiện sau

```
public void OnPointerEnter(PointerEventData evendata)
{
    mouseOnCount = mouseOnCount + 1;
    Debug.Log(mouseOnCount);
    audio.PlayOneShot(audioclip);
}
public void OnPointerExit(PointerEventData eventdata)
{
    Debug.Log("exitPointer");
}
```

Lưu ý `OnPointerEnter` và `OnPointerExit` các bạn phải viết chuẩn, biến mouseOnCount dùng để đếm số lần hover

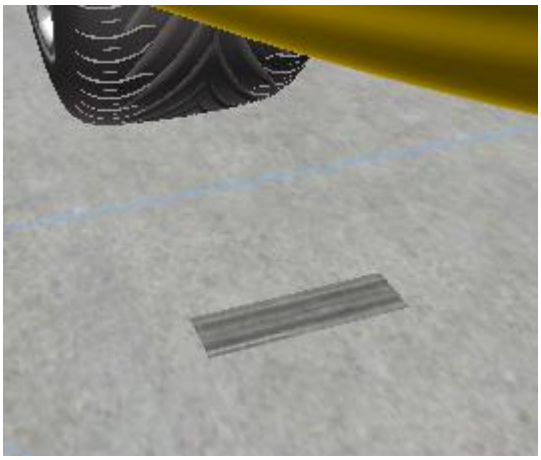
4. Chạy dương trình, tệp lệnh này các bạn gắn vào đối tượng cần thực hiện

6.3. Tạo Skidmark cho xe car

Bước 1: Tạo 2 biến sau

```
public ParticleEmitter Effect_L;
public ParticleEmitter Effect_R;
public Transform skidmark;
```

Với Effect_L, Effect_R là hiệu ứng khói
Skidmark là đối tượng vạch được tạo
Cụ thể như hình ảnh:



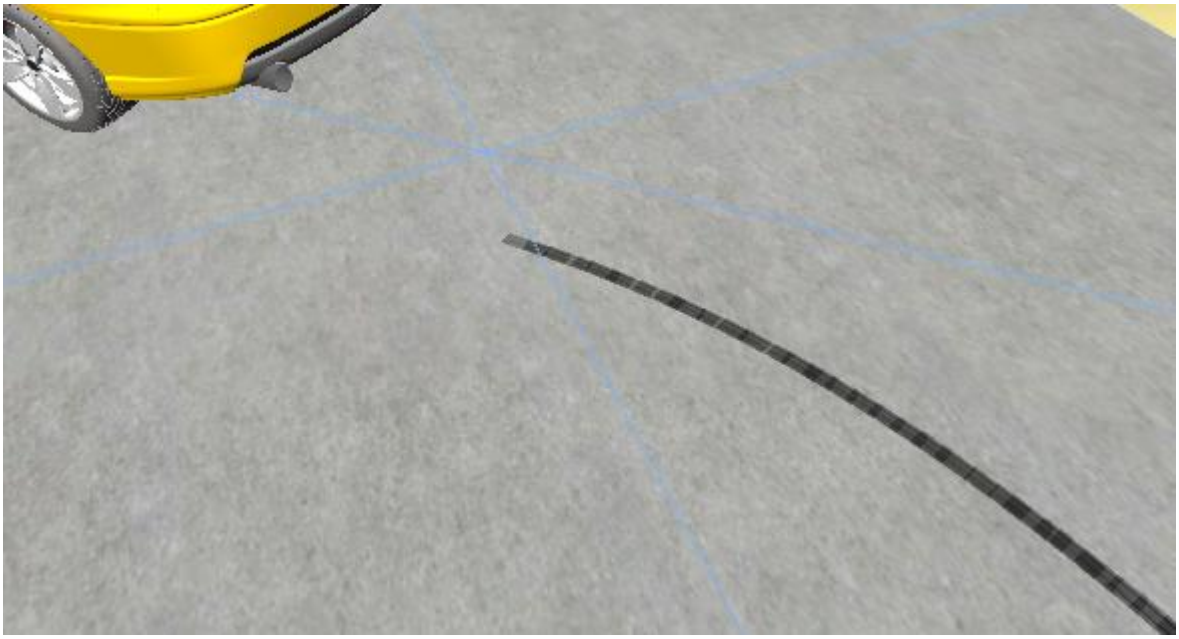
Bước 2: Tạo ra phương thức `BrakeSkidMark`
Với tham số truyền vào là Status

```
if (status == true)
{
    WheelHit hit;
    Vector3 pos;
    if (WheelRL_col.GetGroundHit(out hit))
    {
        Debug.Log("hit: True");
        //GameObject.Find("Smoke_L").particleEmitter.emit = true;
        Effect_L.particleEmitter.emit = true;
        Effect_R.particleEmitter.emit = true;
        pos = hit.point;

        pos.y +=0.2f;
        Instantiate(skidmark, pos, transform.rotation );

    }
}
else
{
    //GameObject.Find("Smoke_L").particleEmitter.emit = false;
    Debug.Log("hit: false");
    Effect_L.particleEmitter.emit = false;
    Effect_R.particleEmitter.emit = false;


}
```

Bước 3: Trong Upadate viết Phím ngắt (Space)

```
if (Input.GetKey(KeyCode.Space))
{
    BrakeSkidMark(true);


}
else
{
    BrakeSkidMark(false);
}
```

Bước 4: cho chạy chương trình



7. Cách biên dịch ra file APK trong unity

Để thực hiện biên dịch được các bạn làm theo các bước sau:

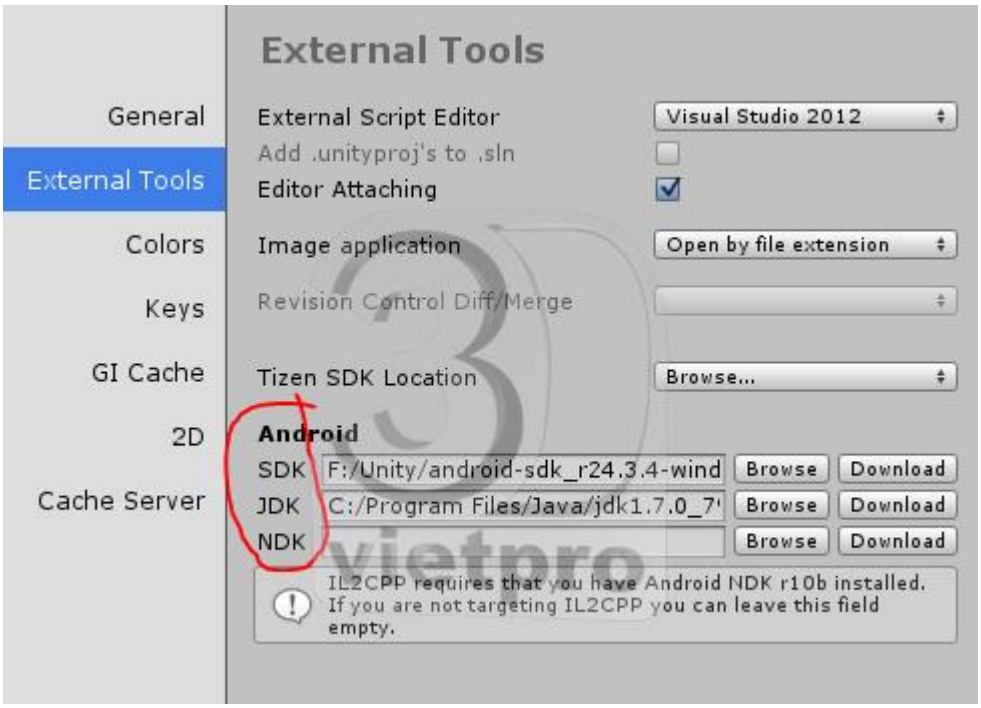
* Bước 1:

Download:

+ File android sdk tại đây

+ File JDK tại đây (Với jdk các bạn phải chọn phiên bản đúng với hệ điều hành của chúng ta là 32bit hay là 64bit)

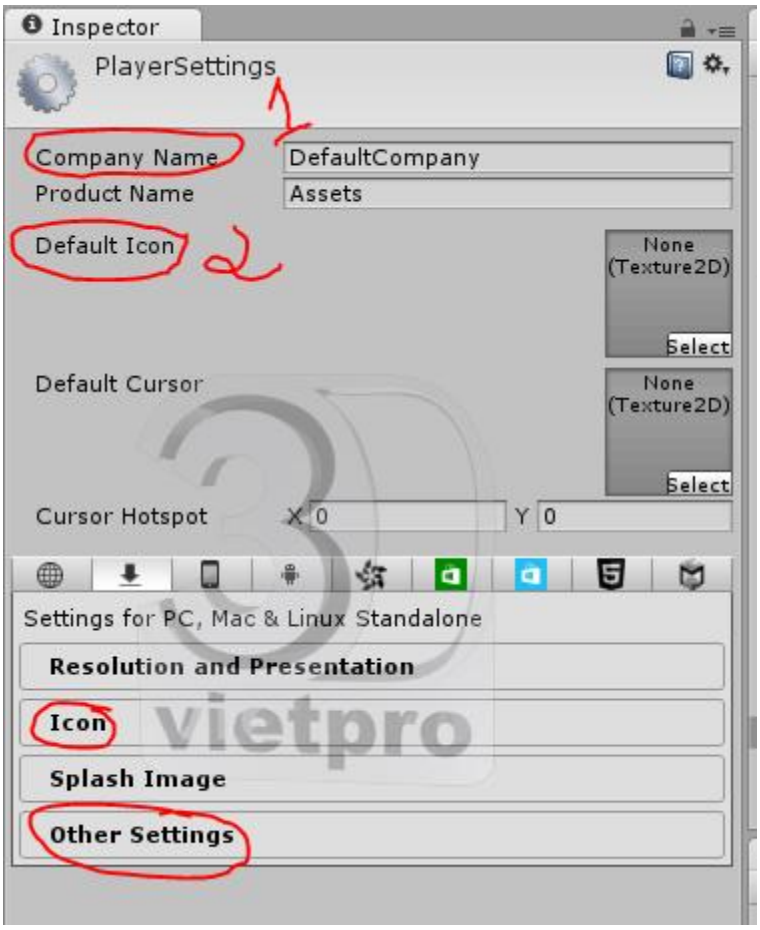
Sau khi down load xong các bạn giải nén và trỏ vào thư mục như hình ảnh phía dưới: (Nằm trong Menu>edit>References)



*Bước 2: Trong giao diện unity vào File>BuildSettings

Bản thân unity chúng ta có thể biên dịch ra rất nhiều môi trường khác nhau, nhưng trong bước này các bạn lưu ý phần android và các phân mà tôi bôi màu đỏ.

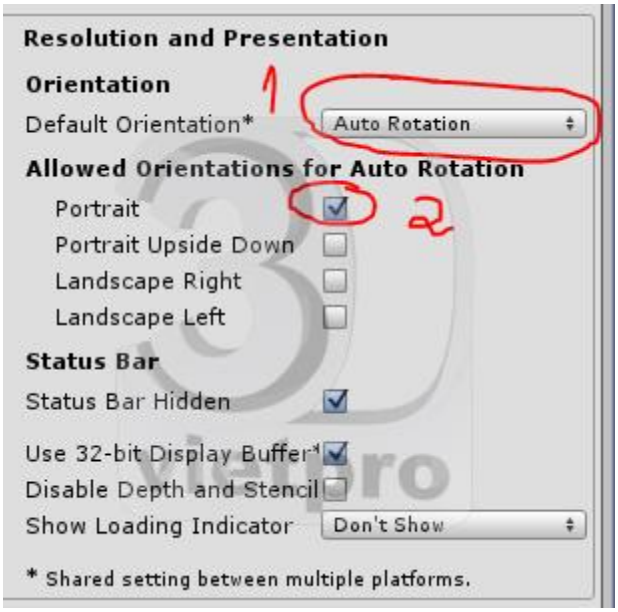
* Bước 3: Các bạn chọn Player Settings trong phần màu đỏ ở bước 2, sau khi chọn xong xuất hiện cửa sổ bên phía tay phải của các bạn như sau:



Trong phần hình ảnh của bước 3 này các bạn lưu lý phần tôi đánh dầu là phần 1, 2. Với phần 1 các bạn phải nhập tên công ty hay tập thể cá nhân của các bạn, và 2 các bạn phải đưa kích thước ảnh icon cho game (Lưu ý phải là hình vuông, thông thường tôi để 400x400)

Bước 4: Các thành phần trong cửa số player setting

- Với phần Resolution and presentation



Trong hình này phần 1 chúng ta thực hiện chọn chế độ tự động cho phép màn hình game của chúng ta tự động xoay hoặc đặt chế độ thẳng đứng hay nằm ngang.

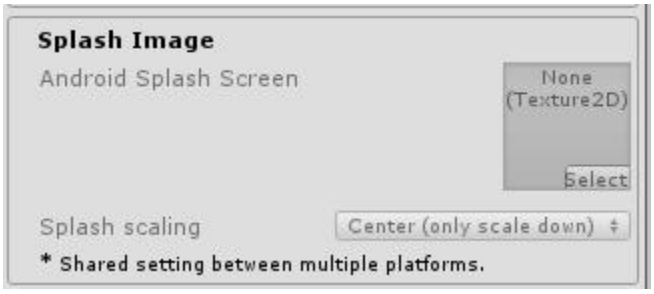
Màn hình thẳng đứng chúng ta để là: Portrait

Còn nằm ngang chúng ta để là: Landscape

- Với phần Icon

Là phần biểu tượng cho game, các bạn có thể thay đổi theo kích thước đối với mỗi độ phân giải của điện thoại.

- Với phần Splash Image



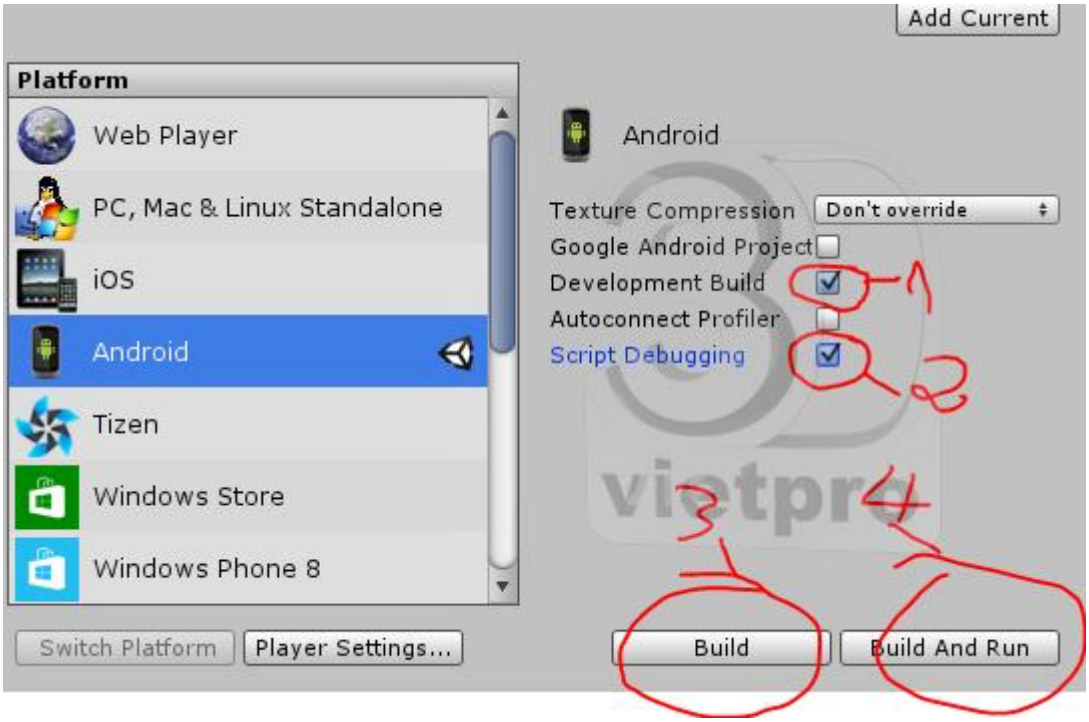
Là ảnh được hiển thị khi bắt đầu vào game của chúng ta, thông thường thì với bản unity free khi vào game các bạn sẽ thấy xuất hiện ảnh của Unity, còn phiên bản có bản quyền hay crack chúng ta có thể đặt ảnh của chúng ta tại đây khi bắt đầu mở game.

- Với phần Other setting

Các bạn chú ý 3 phần tôi đánh dấu, trong đó 1 chúng ta phải đặt tên identifier cho game chúng ta như tôi đặt ở trên trong đó các bạn không được đặt số ban đầu. Phần 2 là phiên bản game chúng ta biên dịch và phần 3 là thiết bị có hỗ trợ cho chíp ARMv7 hay x86 có thể cài đặt được không, lưu ý khi chúng ta chọn 2 chế độ này thì dung lượng game của chúng ta sẽ thay đổi. (Nếu armv7 dung lượng game sẽ cao hơn so với x86).

- Phần Publishing setings là phần liên quan tới bản quyền, tôi sẽ hướng dẫn các bạn ở phần sau

*Bước 5: Thực hiện Build trong cửa sổ build setting.



1. Chúng ta dành cho nhà phát triển

2. Dành cho debug game

3. build ra thư mục nào đó trên máy tính

4.Thực hiện build và chạy luôn, lưu ý các bạn phải cắm đây lối tới điện thoại

Đó là các phần tôi hướng dẫn các bạn build 1 game, chúc các bạn thực hiện thành công, mọi thắc mắc các bạn có thể comment và phản hồi tới admin.

Nguồn 3Dvietpro