به نام خداوند خیلی خیلی مهربان

مقدمه ای بر اسکریپت نویسی Unity

نویسنده : **گراهام مک آلیستر**

ترجمه : **علی زنجیران**(AliyerEdon)

مرداد 1388 **شعبان**

پیشگفتار

سلام. سلام به همه. این هم دومین قسمت از آموزش مربوط به انجین Unity که در مورد اسکریپت نویسی اونه. با خوندن این آموزش شما آماده می شید برای خونـدن آمـوزش سـاخت بـازی هـای اول شـخص کـه آخرین گام برای بازی ساز شدنتونه!! یعنی بعد اون و با کمک گرفتن از خلاقیتتون می تونید بازی های اول شخص عالی رو بسازید. بازی هـایی کـه هـیچ کـم و کاسـتی بـا بـازی هـای خـارجی نداشـته باشـه. چـون از تکنولوژی یکسانی استفاده می کنید و این تکنولوژی به کمک Unity امکان پذیر شده. فقط پشتکار یادتون نره چون این صنعت بسیار مشکل و کار طلبیه و انگیـزه ی قـوی ای نیـاز داره. بـه رویاهـاتون فکـر کنیـد!! همین!!

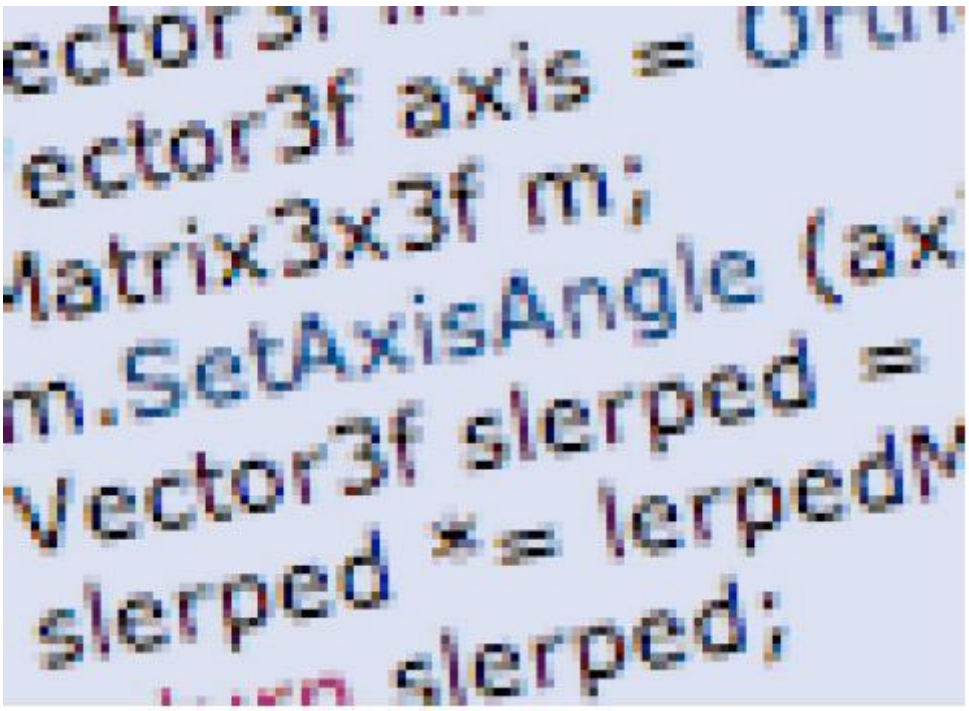
مستر AliyerEdon – مرداد 1388

مقدمه ای بر اسکریپت نویسی Unity

اسکریپت نویسی به عنوان بخشی حیاتی از Unity است چون تایین کننده ی رفتار بازی شماست. این آموزش مقدمه ای است بر پایه های اسکریپت نویسی جاوا در Unity. اطلاعات قبلی در مورد جاوا و Unity مورد نیاز نیست.

زمان برای تکمیل: دو ساعت

نویسنده: گراهام مک آلیستر



محتویات

1. هدف های آموزش

اسکریپت نویسی چگونگی تعیین رفتار یا نقش های بازی توسط برنامه نـویس در Unity مـی باشـد. زبـان اسکریپت نویسی پیشنهادی برای Unity جاوا اسکریپت می باشد. اگرچه می توان از سی شارپ و Boo نیز استفاده کرد. این آموزش پوشش دهنده ی پایه های اسکریپت نویسی در Unity و همچنین مقدمه ایست بر عناصر کلیدی API یه Unity. فرض کنید API قطعه کدهایی است که از قبـل بـرای شـما نوشـته شـده است تا بتوانید بر طراحی بازی تمرکز کنید و سرعت ساخت بازی تان بالا رود.

فهم خوب این پایه ها به شما کمک ی کند تا در آینـده از قـدرت کامـل Unity در پـروژه هایتـان اسـتفاده کنید.

---

2. پیش نیازها

این آموزش بر عنصر اسکریپت نویسی Unity تمرکز دارد. پس برای این آموزش باید بـا محـیط کـاربری Unity آشنایی داشته باشید.(می توانید از آموزش واسط کاربری Unity استفاده کنید).

نکته: هر متنی که با حرف "-" شروع می شود به معنی این است که باید کاری توسط شما انجام شود.

---

3. قراردادهای اسمی

قبل از اینکه شروع کنیم بهتر است که تعدادی از قراردادهای Unity را شرح دهیم:

1.متغیر ها (Variable) : از متغیر ها برای نوشتن اطلاعات در آنها استفاده می شود.

2. توابع (Function): توابع قطعه های کدی هستند که یک بار نوشته می شوند و چندین بار مورد استفاده قرار می گیرند.

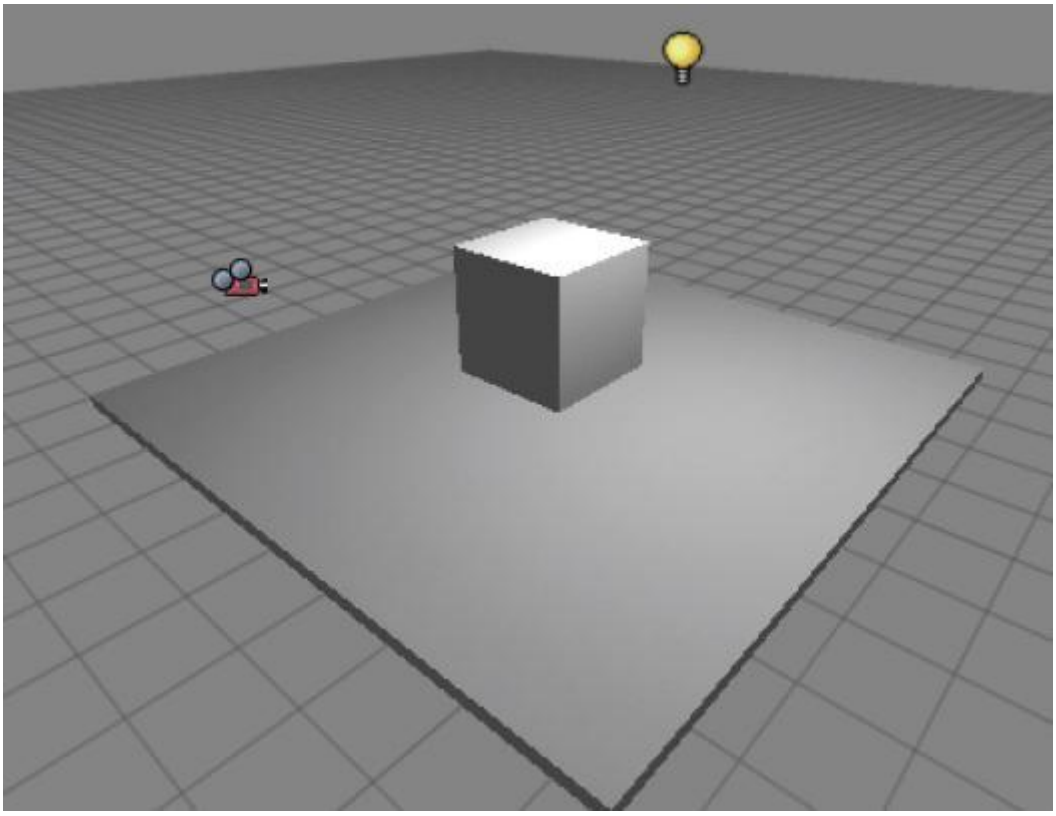
3. کلاسها (Class): کلاس ها دربرگیرنده ی تعدادی تابع در خود هستند.

نکته: وقتی که در حال مطالعه ی API یه Unity هستید به حروف اول کلمات دقت کنید. این کار باعث می شود که ارتباط بین اشیاء را بهتر درک کنید.

---

4. ورودی کاربر

در اولین پروژه ی خود ما به بازیکن اجازه می دهیم که در محیط ساده ی بازی حرکت کند.



آماده سازی صحنه

ابتدا بیایید تا یک سطح صاف بسازیم که بازیکن بتواند بر روی آن حرکت کند. ما از یک مکعب صاف شده برای این کار استفاده خواهیم کرد.

- **یک مکعب بسازید و اندازه های X Y Z آن را به ترتیب برابر 5 0,1 5 قرار دهید. حال باید مکعب ما شبیه یک سطح صاف و کشیده شده باشد. نام این مکعب را در Hierarchy View به Plane تغییر دهید.**
- **دومین مکعب را ساخته و آن را در مرکز شی Plane قرار دهید. اگر شی را در قسمت نمای بازی (Game View)نمی بینید، مکان دوربین خود را تغییر داده و آن را اصلاح کنید. نام این مکعب را به Cube1 تغییر دهید.**
- **همچنین شما باید یک نور نقطه ای ساخته و آن را در بالای مکعب دوم قرار دهید تا صحنه ی شما دید بهتری داشته باشد.**
- **بازی خود را ذخیره کنید. برای این کار از منوی اصلی گزینه ی File>Save As را انتخاب کرده و به بازی تان یک اسم دهید.**

اولین اسکریپت ما

حال ما آماده ایم تا برنامه نویسی بازی را شروع کنیم. ما می خواهیم به بازیکن اجازه دهیم تا با کنترل مکان دوربین در محیط بازی ما حرکت کند. برای این کار ما یک فایل اسکریپت ساخته و اسکریپت خود را در آن می نویسیم که در آن اسکریپت ورودی کیبورد را از کاربر می گیریم و در نهایت فایل اسکریپت را به دوربین بازی اضافه می کنیم.

- با ساختن اسکریپتی خالی شروع می کنیم. از منوی اصلی گزینه ی Assets>Create>Javascript را انتخاب کرده و نام اسکریپت ایجاد شده را در قسمت Project View به Move1 تغییر دهید.
- با دوبار کلیک کردن بر روی فایل اسکریپت آن را باز کنید. می بینید که به صورت پیش فرض تابع Update() در آن موجود است. هر کدی که در داخل این تابع نوشته شود در هر فریم یکبار اجرا می شود.

برای حرکت دادن هر شی در Unity ما نیاز داریم تا مکان آن شی را تکان دهیم که توسط خاصیتی از شی به نام transform قابل انجام است که Transform دارای تابعی به نام Translate در زیرمجموعه ی خود دارد که این کار را برای ما انجام می دهد. تابع Translate دارای سه پارامتر X، Y و Z برای حرکت است. چون ما می خواهیم که دوربین اصلی (Main Camera) خود را با استفاده از کلید های ماوس حرکت دهیم، پس از کلید های ماوس برای پارامتر های این تابع استفاده می یکنیم.

```
function Update () {
    transform.Translate(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
}
```

تابع Input.GetAxis() مقداری بین -۱ و ۱ را برمی گرداند. برای مثال بر روی محور افقی ، کلیک چپ ماوس برابر -1 و کلیک راست ماوس برابر 1 است.

دقت کنید که مقدار پارامتر محور Y به این دلیل صفر است که ما نمی خواهیم دوربینمان به سمت بالا حرکت کند. محورهای افقی (Horizontal) و عمودی (Vertical) به صورت پیش فرض در Unity تعیین شده است که با انتخاب گزینه ی Edit>Project Setting>Input قابل تغییر و تعیین است.

- فایل اسکریپت Move1 را باز کرده و کدهای بالا را در آن وارد کنید. به حروف بزرگ کلمات دقت کنید.

اضافه کردن اسکریپت

حال که اسکریپت ما کامل شد، چگونه به Unity بگوییم که کدام شی ما باید دارای این رفتار باشد؟ تمام کاری که ما می خواهیم بکنیم این است که تعیین کنیم کدام شی باید کدها ی نوشته شده در اسکریپت را اجرا کند.

- ابتدا شی ای که می خواهید اسکریپت را به آن اضافه کنید را انتخاب کنید. در این قسمت ما می خواهیم این اسکریپت را به شی Main Camera اضافه کنیم. پس از Hierarchy View شی Main Camera را انتخاب کنید.
- سپس از منوی اصلی گزینه ی Components>Scripts>Move1 را انتخاب کنید. این کار اسکریپت Move1 را به شی Main Camera متصل می کند. همانطور که می بینید در قسمت Inspector View مر بوط به شی Main Camera کامپوننت Move1 اضافه شده است.

نکته: همچنین شما می توانید با درگ کردن اسکریپت از Project View به داخل شی مورد نظر در Scene View این کار را انجام دهید.

- بازی را اجرا کنید. حال شما باید بتوانید با استفاده از کلید های W.S.A.D در محیط با استفاده از دوربین حرکت کنید.

اگر دقت کرده باشید، دوربین با سرعت زیادی حرکت می کند. حال بیایید راه مناسبی برای کنترل سرعت دوربین ارائه کنیم.

Delta Time

چون شما کدهای خود را در داخل تابع Update() وارد کرده اید، به همین دلیل دوربین شما با سرعتی بر اساس متر بر فریم حرکت می کند. برای بهبود این مشکل ما مقدار برگشتی از تابع Input.GetAxis()را در مقدار Time.deltaTime و همچنین میزان سرعت ضرب می کنیم.

```
var speed = 5.0;

function Update () {
     var x = Input.GetAxis("Horizontal") * Time.deltaTime * speed;
     var z = Input.GetAxis("Vertical") * Time.deltaTime * speed;
     transform.Translate(x, 0, z);
}
```

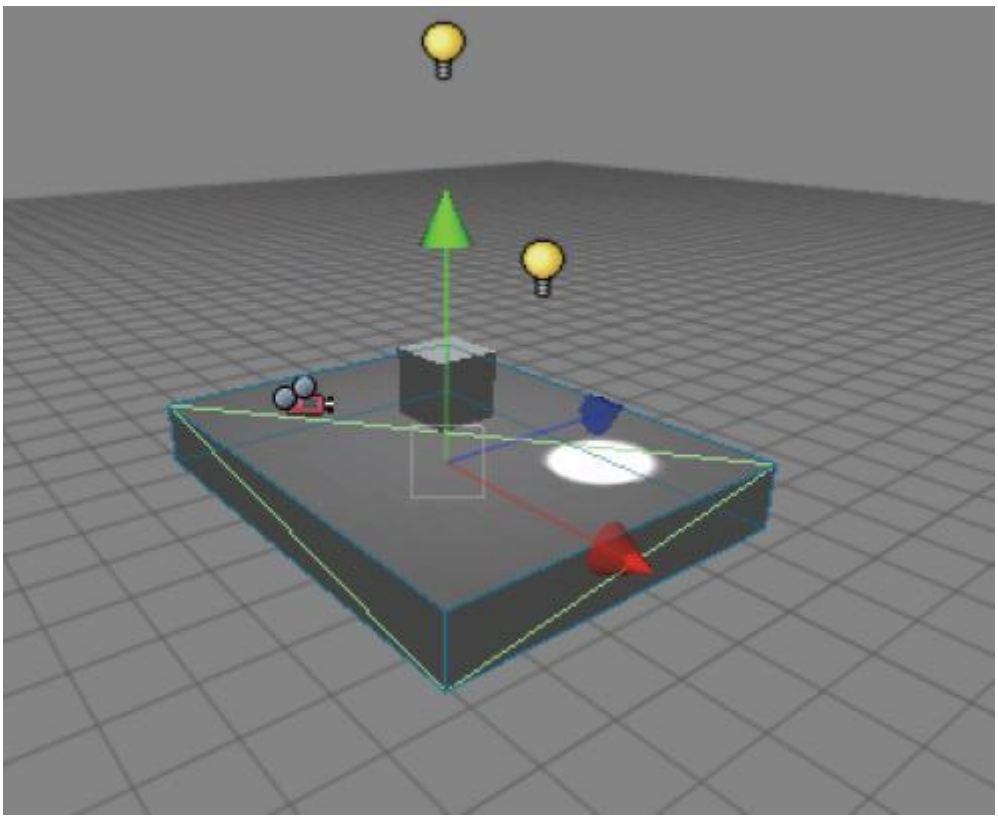
- اسکریپت Move1 را با کدهای زیر جایگزین کنید.

دقت کنید که متغیر speed بیرون از تابع Update() قرار دارد. چون در بیرون قرار دارد پس متغیر عمومی می شود و این یعنی اینکه مقدار این متغیر را می توان توسط واسط کاربری Unity در Inspector View مربوط به شی ای که اسکریپت به آن اضافه شده است، تغییر داد. این روش راه مناسبی برای امتحان کردن مقادیر مختلف برای سرعت به آسانی می باشد. حال شما می توانید به آسانی مقدار سرعت حرکت دوربین را حتی هنگام اجرای بازی تغییر دهید!!

5. متغیر های اتصال

اتصال متغیر ها توسط واسط کاربری Unity یکی از پیشرفته ترین امکانات آن است!! با این ویژگی شما می توانید مقادیر متغیر هایی که در کد نوشته اید را با استفاده از محیط Unity تعیین کنید. به عنوان مثال شما متغیر مورد نظر را در کدنویسی به صورت عمومی تعریف می کنید و توسط محیط کاربری Unity مقدار آن را با درگ کردن تعیین می کنید.

ما مفهوم اتصال متغیر ها را با مثالی که در آن یک نور اسپات لایت دوربین بازی را دنبال می کند، توضیح می دهیم.



- **یک نور اسپات لایت به صحنه اضافه کنید. آن را به مقدار لازم حرکت دهید تا شبیه تصویر بالا شود.**
- **فایل اسکریپت جدیدی ایجاد کنید و نام آن را Follow بگذارید.**

بیایید بررسی کنیم که چکار می خواهیم بکنیم. ما می خواهیم نور اسپات لایت ما به سمتی بتابد که دوربین ما در آنجا قرار دارد. برای انجام این کار تابعی در Unity به نام transform.LockAt() وجود دارد که این کار را برای ما انجام می دهد. اگر شما فکر می کنید که چگونه این کار را انجام دهیم و مقدار زیادی کد در ذهنتان ساخته اید، ارزشش را دارد که در مورد تابعی که Unity در API خود دارد، از قسمت راهنمای Unity تحقیق کنید. همچنین تحقیق کردن در مورد تابع transform نیز خالی از لطف نیست.

حال ما به قسمت اتصال متغیر ها بازمیگردیم. ما برای تابع LockAt() از چه پارامتر هایی استفاده می کنیم؟ خوبه. ما از یک شی برای پارامتر آن استفاده می کنیم. و چون می خواهیم این شی را در محیط کاربری Unity تعیین کنیم، پس متغیر آن شی را عمومی تعریف می کنیم.

کدهای نوشته شده در Follow.js:

```
var target : Transform;

function Update () {
    transform.LookAt(target);
}
```

- اسکریپت بالا را به شی نور اسپات لایت اضافه کنید. دقت کنید که متغیر target در قسمت Inspector View مربوط به نور اسپات لایت نمایان می شود.
- در حالی که شی نور اسپات لایت انتخاب شده است، شی Main Camera را از قسمت Hierarchy View به داخل منتغیر target در Inspetcor View درگ کنید. این کار مقدار متغیر target را تعیین می کند. اگر شما می خواهید شی اسپات لایت شی دیگری را دنبال کند، به جای شی Main Camera شی دیگری را به داخل متغیر target درگ کنید.(البته باید از نوع Transform باشد!).
- بازی را اجرا کنید. همانطور که در قسمت نمای صحنه (Scene View) می بینید با حرکت دوربین توسط شما جهت شی اسپات لایت هم به سمت دوربین تغییر می کند!

---

6. دسترسی کامپوننت ها

همانطور که یک شی می تواند دارای چندین اسکریپت(یا کامپوننت) اضافه شده به آن باشد، گاهی اوقات لازم است که به متغیرها و توابع کامپوننت های دیگر نیز دسترسی داشته باشیم. Unity این امکان را با استفاده از تابع GetComponent() فراهم آورده است.

حال ما اسکریپت دیگری به شی اسپات لایت اضافه می کنیم که تعیین کند که وقتی که کلید Space در کیبورد زده شد به شی Cube1 نگاه کند.

اول بیایید ببینیم چه کار می خواهیم بکنیم:

1. فهمیدن اینکه چه زمانی کلید Space زده شده است.
2. زمانی که کلید مورد نظر زده شد، به شی Cube1 نگاه کند.

چه طور این کار را انجام دهیم؟ خوبه. اسکریپت Follow دارای شی ای به نام target است که تعیین می کند اسپات لایت به چه شی ای نگاه کند. ما نیاز داریم تا مقدار این متغیر را تغییر دهیم. ما می توانیم شی مورد نظر دوم را با کد تعیین کنیم ولی همانطور که گفته شد با استفاده از واسط کاربری Unity بهتر است.

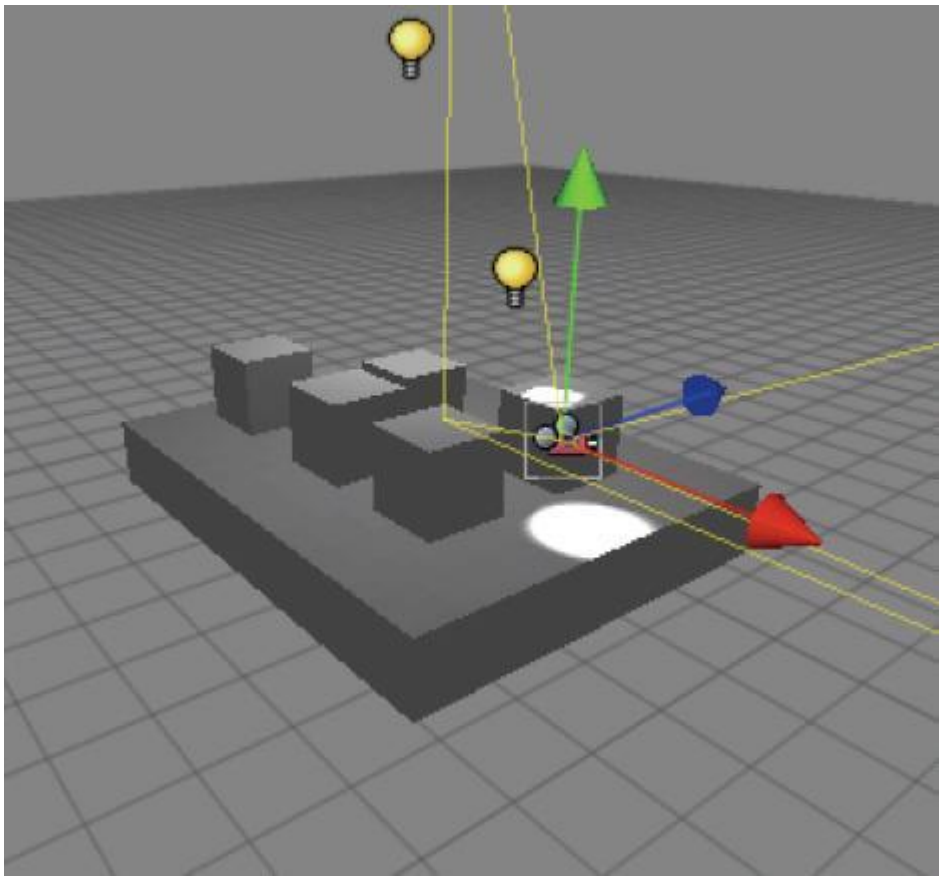
- فایل اسکریپت جدیدی ایجاد کرده و نام آن را به Switch تغییر دهید. کد های زیر را به ان اسکریپت اضافه کنید:

```
var switchToTarget : Transform;

function Update () {
     if (Input.GetButtonDown("Jump"))

          GetComponent(Follow).target = switchToTarget;

}
```

به قسمت GetComponent دقت کنید. ببینید که چگونه با این تابع به متغیر target موجود در فایل Follow دسترسی پیدا میکنیم.

- اسکریپت Switch را به شی اسپات لایت اضافه کنید و شی Cube1 را به عنوان متغیر SwitchToTarget در Inspector View تعیین کنید.
- بازی را اجرا کنید. دوربین را حرکت داده و ببینید چگونه اسپات لایت شما را دنبال می کند. سپس کلید Space را زده تا ببینید که اسپات لایت حال به شی Cube1 زوم کرده است!!

انجام اینکار با کد



یک مقدار قبل تر ما ذکر کردیم که می توان با کد نیز مقدارمتغیر را تعیین کرد. حال بیایید ببینیم چگونه این کار امکان پذیر است.

این نکته رو برای مقایسه به یاد داشته باشید: همیشه تعیین کردن مقدار متغیر توسط محیط کاربری Unity توصیه می شود.

مسئله ی ما قبل تر این بود که چگونه به اسپات لایت بگوییم که وقتی کلید Scape زده شد به شی Cube1 نگاه کن. راه حل ما این بود که با تعیین متغیری به صورت عمومی به نام switchToTarget و درگ کردن Cube1 در واسط کاربری Unity به این متغیر، این کار را انجام دهیم. برای اینکه در کدنویسی این کار را انجام دهیم دو راه داریم:

1. با استفاده از نام شی.
2. با استفاده از برچسب (Tag) شی

1. نام شی

نام شی مورد نظر را می توان درقسمت Hierarchy View مشاهده کرد.برای استفاده از این اسم در کد، ما آن را به عنوان پارامتر برای تابع GameObject.Find() استفاده می کنیم. پس کدی که برای پرش از Main Camera به شی Cube1 مورد استفاده قرار میگیرد، به شرح زیر است:

```
function Update () {
     if (Input.GetButtonDown("Jump"))
     {
          var newTarget = GameObject.Find("Cube").transform;
          GetComponent(Follow).target = newTarget;
     }
}
```

دقت کنید که هیچ متغیر عمومی ای تعریف نکرده ایم چون با استفاده ازنام شی مورد نظر مقدار متغیر را تعیین کرده ایم. در قسمت راهنمای API یه Unity در مورد تابع Find() بیشتر تحقیق کنید.

2. برچسب شی

برچسب یک شی، حاوی متن رشته ای است که برای شناختن یک کامپوننت می توان از آن استفاده کرد. برای دیدن برچسب های داخلی Unity بر روی دکمه ی Tag موجود درقسمت Inspector View کلیک کنید. دقت کنید که شما می توانید برای خود نیز بسازید. تابع مخصوص برای پیدا کردن یک کامپوننت از روی برچسب آن،GameObject.FindWithTag() می باشد که یک مقدار متنی به عنوان پارامترمی گیرد. کد کامل به شرح زیر است:

```
function Update () {
     if (Input.GetButtonDown("Jump"))
     {
          var newTarget = GameObject.FindWithTag("Cube").transform;
          GetComponent(Follow).target = newTarget;
     }
}
```

7. ساختن

بعضی موقع ها این موضوع لذت بخشه که بخواهیم در زمان اجرای بازی شی ای را بسازیم. برای این کار ما از تابع Instantiate استفاده می کنیم.

حال بیایید برای اینکه چگونگی کار این تابع را شرح دهیم مثالی از ساختن یک شی در حین اجرای بازی را بدهیم. ساختن یک شی زمانی که کاربر کلید چپ ماوس را فشار داد.

خب. باید ببینیم چه کاری می خواهیم بکنیم. ما می خواهیم که وقتی بازیکن درحال قدم زدن در محیط است بعد از زدن کلیک چپ ماوس شی ای ساخته شود. چند چیز را باید توضیح دهیم:

1. کدام شی را می خواهیم بسازیم؟
2. کجا آن را بسازیم؟

در مورد اینکه کدام شی را بسازیم، بهترین راه این است که یک متغیر عمومی برای آن تعریف کنیم. به عبارت دیگر می خواهیم شی ای که ساخته شود را با استفاده از درگ کردن در محیط کاربری Unity تعیین کنیم.

در مورد اینکه کجا شی مورد نظر را بسازیم، می خواهیم که شی در جایی که کاربر قرار دارد و کلیک چپ ماوس زده شده، ساخته شود.

تابع Instantiate دارای سه پارامتر است. پارامتر اول شی ای است که می خواهیم بسازیم. پارامتر دوم مکان (Position) شی ای است که می سازیم و پارامتر سوم چرخش (Rotation) شی ای است که می خواهیم بسازیم.

کد کامل برای این کار به شرح زیر است:(Create.js)

```
var newObject : Transform;

function Update () {
     if (Input.GetButtonDown("Fire1")) {
        Instantiate(newObject, transform.position, transform.rotation);
     }
}
```

فراموش نکنید که توابع transform.position و transform.rotation مکان و چرخش شی ای است که اسکریپت به آن اضافه می شود. در اینجا مربوط به Main Camera است.

مناسب است که هر شی ای که در زمان اجرا ایجاد می شود یک Prefab باشد. حال ما شی Cube1 را به داخل یک Prefab می اندازیم.

- ابتدا یک Prefab خالی می سازیم. از منوی اصلی گزینه ی Assets>Create>Prefab را انتخاب کنید و نام آن را به Cube تغییر دهید.
- شی Cube1 را از قسمت Hierarchy View به داخل Cube در Project View درگ کنید. دقت کنید که آیکون Prefab مربوط به Cube تغییر می کند.

حال ما می توانیم کد جاوا اسکریپت خود را بسازیم:

- فایل جاوا اسکریپتی بسازید و نامش را به Create تغییر دهید. کدهای بالا را به آن اضافه کنید.

- اسکریپت ساخته شده را به شی Main Camera درگ کنید و سپس به متغیر موجود در قسمت Inspector View مربوط به Main Camera، Prefab ه Cube را اضافه کنید(با درگ کردن Prefab به داخل آن).
- بازی را اجرا کنید. درمحیط حرکت کرده و کلیک چپ ماوس را بزنید. خواهید دید که شی مکعبی در زمان اجرا نمایان می شود.

---

8. اشکال زدایی (دباگ کردن)

اشکال زدایی یا دباگ کردن مهارتی برای پیدا کردن خطاهای انسان در کد است. (بیایید بهشون بگیم دردسر انسانی!!). Unity برای کمک کردن کلاسی به اسم Debug را فراهم کرده ایت که تابعی به نام Debug.Log() دارد.

Log

تابع Log() به کاربر اجازه می دهد تا به کنسول (میز فرمان) Unity پیامی را بفرستد. دلایل این کار به شرح زیر است:

1. مطمئن شدن از اینکه قسمتی از کد در هنگام اجرای بازی، اجرا می شود.
2. گزارش دادن از وضعیت یک متغیر.

حال ما از تابع Log() برای فرستادن پیامی به کنسول Unity استفاده می کنیم. هنگامی بازیکن کلیک چپ ماوس را زد.

- اسکریپت Create را باز کرده و کد زیر را بعد از تا بع Instantiate در بلوک if وارد کنید.

```
        Debug.Log("Cube created");
```

- بازی را اجرا کنید. کلیک چپ ماوس را بزنید. حال پیامی را با مضمون "Cube Created" در بالای محیط کاربری Unity خواهید دید.

Watch

یکی دیگر از خاصیت های اشکل زدایی یا دباگ کردن، قابلیت تفسیر یک متغیر خصوصی (Private) است. این باعث می شود هنگامی که حالت Debug فعال است، متغیر در قسمت Inspector View نمایان شود. اما قابل ویرایش نیست.

برای فهماندن این موضوع ما متغیرخصوصی ای را تعریف می کنیم که تعداد مکعب های ساخته شده را شمارش می کند.

- اسکریپت Create را باز کرده و دو خط را اضافه کنید.

1. یک متغیر خصوصی با نام cubeCount اضافه کنید.
2. هر موقع که مکعبی ساخته شد این متغیر را افزایش دهید.

کد کامل به شرح زیر است (Create.js):

```
var newObject : Transform;
private var cubeCount = 0;

function Update () {

     if (Input.GetButtonDown("Fire1")) {
        Instantiate(newObject, transform.position, transform.rotation);
        Debug.Log("Cube created");
        cubeCount++;
     }

}
```

- بازی را اجرا کرده و کلیک چپ ماوس را چند بار بزنید تا چندین مکعب ایجاد شود. به Inspector View دقت کنید که چگونه با هر بار ایجاد یک مکعب متغیر cubeCount افزایش پیدا می کند. همچنین دقت کنید که رنگ آن خاکستری است. این به این دلیل است که غیر قابل ویرایش است.

---

9. انواع اسکریپت های رایج

هر وقت که یک فایل جاوا اسکریپت می سازید، دارای یک تابع Update() به صورت پیش فرض است. در این قسمت در مورد انواع دیگر توابع رایج بحث می کنیم. به سادگی تابع Update() را با یکی از توابع ذکر شده در زیر جایگزین کنید.

FixedUpdate()

کدهای موجود در این تابع در بازه ی زمانی ثابتی اجرا می شوند(فریم ریتی ثابت). این تابع استفاده ی رایجی دارد هنگامی که به یک شی Rigidbidy نیرویی وارد می کنید.

```
// Apply a upwards force to the rigid body every frame

function FixedUpdate () {
    rigidbody.AddForce (Vector3.up);
}
```

Awake()

کدهای درون این تابع وقتی اسکریپت ایجاد می شود اجرا می شود.

Start()

این تابع قبل از تابع Update() ولی بعد از Awake() اجرا می شود. فرق این تابع با Awake() در این است که این تابع وقتی اجرا می شود که اسکریپت به شی ای اضافه شده است و نه وقتی که اسکریپت ایجاد می شود.

OnCollisionEnter()

کدهای این تابع وقتی اجرا می شود که شی مورد نظر با شی ای دیگر برخورد کند.

OnMouseDown()

کدهای این تابع وقتی اجرا می شود که ماوس بر روی شی ای که دارای خاصیت GUIElement **یا** Collider **است، برود و کلیک کند.**

```
// Loads the level named "SomeLevel" as a response
// to the user clicking on the object

function OnMouseDown () {
    Application.LoadLevel ("SomeLevel");
}
```

OnMouseOver()

کد های این تابع وقتی اجرا می شود که ماوس بر روی شی ای که دارای دو خاصیت ذکر شده در بالا باشد، برود.

```
// Fades the red component of the material to zero
// while the mouse is over the mesh

function OnMouseOver () {
    renderer.material.color.r -= 0.1 * Time.deltaTime;
}
```

API **یه** Unity **را برای تحقیق بیشتر بر روی این توابع بررسی کنید.**

سرانجام

این قسمت از آموزش در مورد پایه های اسکریپت نویسی در Unity **بود و حال شما می توانید هر آموزش یا نمونه کدی را برای افزایش مهارت و تجربه ی خود بخوانید.**