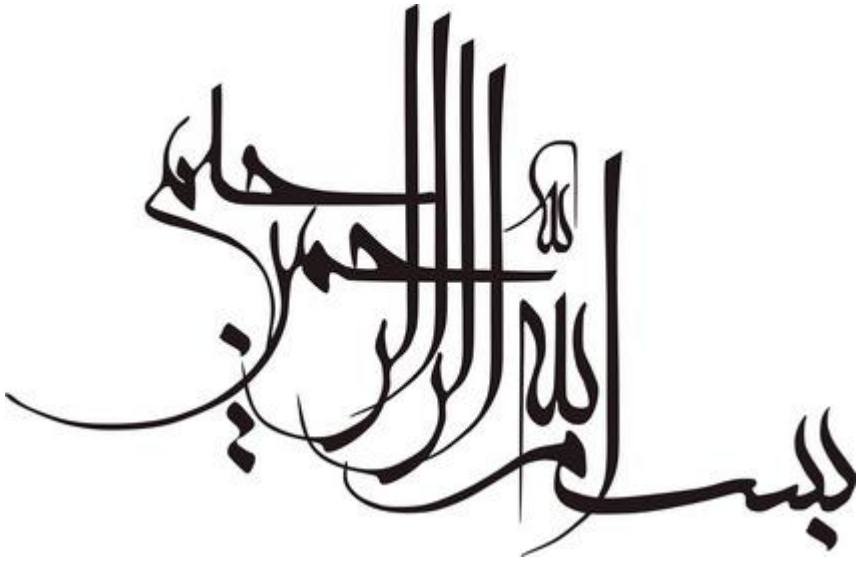


آموزش ساخت بازی اول شخص
با موتور قدرتمند و آسان Unity
قسمت مقدماتی

مترجم و مجری: علی زنجیران (AliyerEdon)

مرداد -شَهریور 1388

مقدمه

سلام. سلامی گرم به همه ی عزیزان بازی ساز و علاقه مند به این صنعت لذت بخش. امیدوارم در کارهاتون موفق باشید و امیدوار! موفق برای اتمام کار و امیدوار برای داشتن پشتکار کافی. چون همونطور که می دونید این صنعت سختی های زیادی داره ولی با کمی علاقه همش ناپدید می شه. من خودم این آموزش ها رو خوندم و اجرا کردم پس بهتون این اطمینان رو می دم که حتما می تونید بفهمید. فقط هر کدی که نوشته شده رو چند بار خودتون بنویسید و تمرین کنید تا حفظ بشید. کدهای هر درس نیم تا یک ساعت زمان برای حفظ شدن نیاز داره. چرا می گم حفظ کنید؟ اول بگم که منظورم از حفظ کردن فقط حفظ کردن نیست بلکه اول فهمیدن و بعد حفظ کردنه!! یعنی وقتی شما فهمیدی و حفظ هم هستی در قسمت های بعدی آموزش (دوره ی متوسطه و پیشرفته) خیلی راحت کدهای پیچیده تر رو می فهمید. به هر حال این تجربه ی من بود. حالا در مورد موتور هم یه حرفی بزنم. این موتور چرا شناخته شده نیست؟ واقعا چرا؟ خوب معلومه چون چند ماهه رو ویندوز عرضه شده و قبلا رو سیستم عامل مک او اس بود. البته اونجا بشدت طرفدار داشت!! و تازه اینکه این موتور از سال 2005 به بعد ساخته شده و به همین دلیل از ابتدای ساختش با تکنولوژی های روز ساخته شده. به نظر من که با ده ها موتور دیگه کار کردم این رو بهترین می دونم از هر نظر. تو موتور های دیگه شما باید 80 در صد کد بدید ولی تو این موتور به گفته ی سازندگان باید 80 درصد وقتتون رووی ساخت بازی به صورت ویژوال بزارید!! هدفشون همین بوده.

خب بقیه ی اطلاعات رو خودتون می تونید از سایت این موتور قدرتمند و دوست داشتنی پیدا کنید:

www.Unity3d.com

مقدمه ای بر ساخت بازی های اول شخص با استفاده از موتور Unity

این آموزش توضیح دهنده ی چگونگی ساخت بازی های اول شخص یا FPS می باشد. این آموزش مقدمه ایست بر پایه های مفهوم برنامه نویسی سه بعدی و همچنین نشان دادن این موضوع که چگونه مثل یک برنامه نویس بازی فکر کنید.

زمان برای تکمیل : 3 تا 4 ساعت

نویسنده: گراهام مک آلیستر

فهرست:

فایل های مورد نیاز را می توانید از لینک زیر دریافت کنید:

www.otee.dk/tutorials/fps_assets.zip

1.هدف های این آموزش

این آموزش دو چیز را فرض می کند. اول اینکه شما علاقه به بازی هـای کـامپیوتری داریـد. حال در زمینه ی بازی کردن، طراحی یا مطالعه در مورد آنها. همچنین فرض می کند که شـما مقداری تجربه برنامه نویسی یا اسکریپت نویسی برنامه های کامپیوتری را داریـد. سـرانجام اینکه شما قصد ساخت بازی های بزرگی را دارید!!

این آموزش چگونگی ساخت بازی اول شخص را توسط Unity به شما آموزش می دهد.

2.پیش نیازها

شما باید قبلا با واسط کاربری Unity و مفاهیم اسکریپت نویـسی آن آشـنایی داشـته باشـید. اگر ندارید می توانید آموزش هایی در مورد آنها را مطالعه بفرمایید.

ابزارهای زیر مورد نیاز برای بازی سازی هستند:

1.برنامه ی مدل ساز سه بعدی

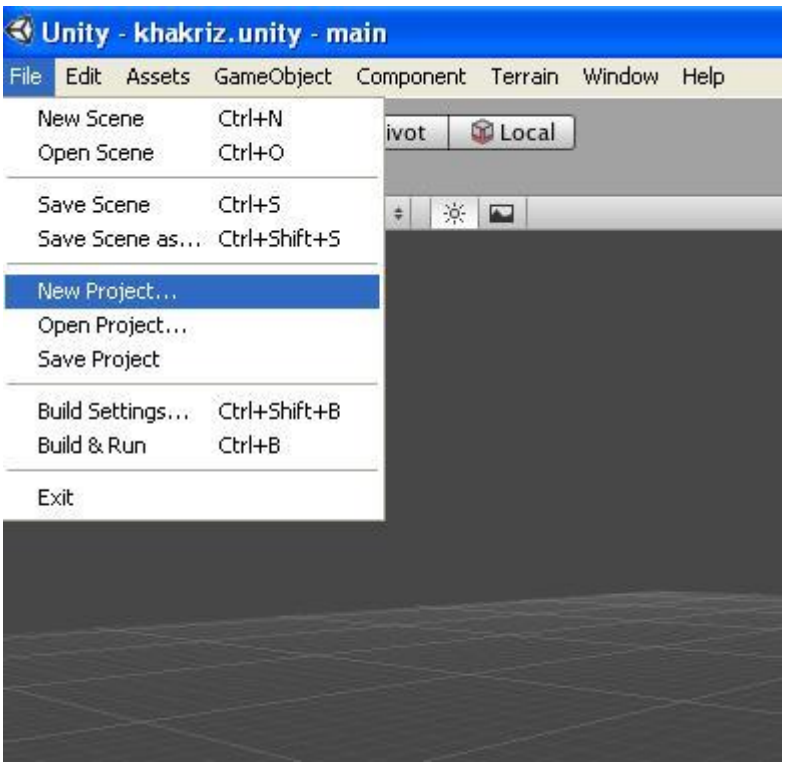
2.فتوشاپ

توجه: هر خطی که با "– "شروع می شود، یعنی باید یک کاری را انجام دهید.

3.شروع پروژه ی جدید

-Unity را اجرا کنید. این ایده ی خوبیست که آیکون آن را در دسکتاپ خود داشته باشید.

-پروژه ی جدیدی بسازید.



در پنل Project شما محتوا(Asset) هـای داخلـی Unity را مـی بینیـد. StandardAssets و Pro Standard Assets . وقتی که ما محتوای جدیدی را ایجـاد مـی کنـیم، بهتـرین راه ایـن است که آنها را براساس کارایی شان در پوشه هایی با نام مرتبط قرار دهیم. به عنوان مثال : Rocket-Explosion-Audio

-فایل زیر را دانلود کنید و با اجرای آن تمام اسکریپت ها و گرافیک هـای اسـتفاده شـده در این آموزش را به پروژه تان اضافه کنید.

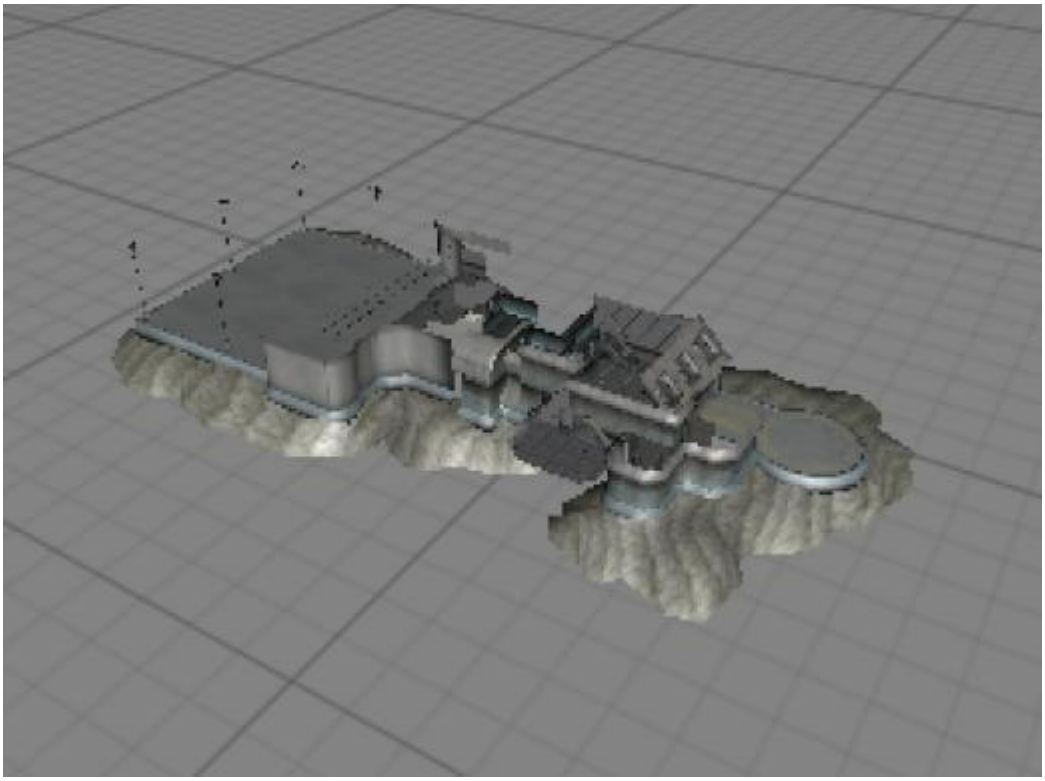
fps_assets.zip

4.آماده سازی محیط بازی

بعد از اینکه محتواها وارد پروژه شدند، شما می بینید که تعداد زیادی پوشه(فولدر در ویندوز) در پنل Project **اضافه شده است.**

- محتوای mainLevelMesh **را از مسیر** Objects\mainLevelMesh\mainLevelmesh **به داخل صحنه بکشید و رها کنید.**

-درحالی که mainLevelMesh **انتخاب شده است، از پنل** Project **در قسمت بالای آن** Setting **را انتخاب کنید و در پنجره ای که باز می شود مطمئن شوید که گزینه ی** Meshes have Coliders **فعال است. اگر این گزینه فعال نباشد، بازیکن شما به سادگی صقوط کرده و این شی را برای برخورد با آن تشخیص نمی دهد.**



نیازی به اضافه کردن هیچ نوری به صحنه ندارید چون این صحنه قبلا به صورت کامل لایت مپ شده است. با اضافه کردن صحنه ی آماده به بازی که قبلا لایت مپ شده است باعث می شود که کارایی محیط و بازی به شدت بالا رود و این مورد برای صحنه های با نورپردازی پیچیده خیلی مفید است. لازم به ذکر است که لایت مپ به معنی ساخته شدن سایه ها در

محیط است و این سایه ها به صورت پیش فرض در شی لایت مپ شده ذخیره می شوند که در این صورت این سایه ها ریل تایم نیستند.

خب حال شما آماده اید تا شخصیت خود را به صحنه وارد کنید.

5.اضافه کردن شخصیت اصلی

حال ما می رویم تا شخصیتی برای کنترل بازی کن اضافه کنیم. Unity یک Prefab مخصوص برای کنترل اول شخص در خود دارد. که در قسمت Project Panel در Standard Assets\Prefab یافت می شود.

-از پنل پروژه پوشه ی Standard Assets را باز کنید و در داخل آن Prefabs را پیدا کنید و از داخل آن First Person Controller را به صحنه درگ کنید. حال آن در صحنه نشان داده می شود.

شما باید شی سیلندری را ببینید که بازیکن را نشان می دهد به علاوه ی سه جهت که نشان دهنده ی مکان بازی کن در صحنه است و همچنین شی زرد رنگی که فضای دید (ViewPort) بازیکن را نشان می دهد. FPS Controller در بردارنده ی یک فضای دید است که این فضای دید زیرمجموعه ای از شی دوربین آن است. FPS Controller الان به عنوان دوربین پیش فرض بازی شماست و با حرکت آن، دوربینی که صحنه ی بازی را نشان می دهد (در قسمت GameView) نیز حرکت می کند. همچنین توجه کنید که یک آیکون دوربین در بالای FPS Controller می باشد که با بالا بردن آن، دید بازیکن نیز بالاتر از سطح زمین می رود.

-چون شما دیگر نیازی به شی MainCamera در صحنه ندارید، می توانید آن را پاک کنید. البته دقت کنید که شی MainCamera مربوط به خود صحنه و نه MainCamera ایکه در داخل FPS Controller است.

-حال دکمه ی Playرا در بالای محیط ادیتور بزنید. شما خواهید دید که می توانید با کلید های W.S.A.D و همچنین ماوس درمحیط بازی گردش کنید.

حال شما یک بازی اول شخص ساده ساخته اید! بیایید به بازیکن یک اسلحه بدهیم.

6.اضافه کردن اسلحه

حال ما می خواهیم که به بازیکن یک شی شبیه نارنجک بدهیم تا بتواند آن را به داخل محیط پرتاب کند. برای این منظور، شما باید مقداری کد جاوااسکریپت بنویسید تا به Unity رفتار اسلحه ی خود را شرح دهید.

بنابراین ما چه کار می خواهیم بکنیم؟ ما می خواهیم به بازیکن اجازه دهیم که به نقطه ای که دوربین بازی نگاه می کند شی ای را پرتاب کند. هرچند باید ابتدا در مورد شخصیت بازی و اسلحه های آن توضیح دهیم. شخصیت بازی ما از نمای اول شخص نگاه می کند که دوربین در مکان چشم های آن قرار دارد. هنگامی که بازیکن با اسلحه ای شلیک کرد، آن اسلحه باید از جایی که دست شخصیت قرار دارد شلیک کند نه از جایی که نگاه می کند. این یعنی اینکه ما می خواهیم شی ای بسازیم که به عنوان نارنجک انداز عمل کند و آن را جایی قرار دهیم که شخصیت دستش در آنجا قرار دارد. این کار باعث می شود که مطمئن شویم اسلحه از مکان مناسب شلیک می کند.

ساختن نارنجک انداز

ابتدا بیایید شی ای به عنوان نارنجک انداز به صحنه اضافه کنیم. شی بازی آیتمی است که در دنیای سه بعدی قرار می گیرد(مثل مدل ها و صدا و غیره) و کامپوننت نیز رفتاری است که به شی بازی تعلق می گیرد. بنابراین شما کامپوننت ها را به اشیاء بازی اضافه می کنید.

- **از منوی اصلی گزینه ی Game Object>Create Empty را انتخاب کنید و نام آن را در قسمت Hierarchy View به Launcher تغییر دهید. دقت کنید که این شی در صحنه نمایان نیست چون یک شی خالی است و فقط برای نگه داری مکان نارنجک انداز از آن استفاده خواهیم کرد.**

حال بیایید برویم سراغ FPS Controller تا ببینیم شی Launcher ما در کجای آن قرار می گیرد.

- شی FPS Controller را انتخاب کرده و مطمئن شوید که نشانگر ماوس بر روی قسمت Scene View قرار دارد. کلید F را بزنید تا به سمت شی FPS Controller زوم شود.
- حال شی Launcher را انتخاب کرده و سپس از منوی اصلی گزینه ی Game Object>Move to view را بزنید. دقت کنید که چگونه شی Launcher در نزدیکی شی FPS Controller قرار می گیرد. حال ما می توانیم کنترل کنیم که شی Launcher چگونه در قسمت تقریبا دست شخصیت قرار گیرد. همچنین شما می توانید با تغییر مکان Launcher تعیین کنید که شخصیت شما چپ دست یا راست دست است که نیازی به کد نوشتن نیست.
- مطمئن شوید که طرح بندی پنجره ی GUI محیط Unity در حالت Split 2 است(از منوی اصلی گزینه ی Window>Layouts>2 Split را انتخاب کنید. سپس بازی را اجرا کنید. مطمئن شوید که شی Launcher در قسمت Hierarchy View انتخاب شده است و سپس در حالی که به نمای صحنه (Scene View) نگاه می کنید، شخصیت بازی را حرکت دهید. می بینید که شی Launcher ما با شخصیت ما حرکت نمی کند.(بازی را متوقف کنید).
- برای حل این مشکل شی Launcher را از قسمت Hierarchy View به داخل شی Main Camera در شیFPS Controller درگ کنید. حال بازی را اجرا کنید. شخصیت را حرکت داده و به نمای صحنه نگاه کنید که چگونه شی Launcher نیز با حرکت دوربین حرکت می کند.

ساختن شی نارنجک

حال بیایید شی نارنجک را بسازیم تا وقتی که بازیکن کلید شلیک(کلیک چپ ماوس) را زد، آن را به جلو پرتاب کنیم.

- برای این آموزش ما از شی ساده ای مثل یک گوی استفاده می کنیم. یک Prefab جدید ساخته (از منوی اصلی گزینه یAssets>Create>Prefab را انتخاب کنید) و نام آن را به Missile تغییر دهید.
- حال یک گوی بسازید.(از منوی اصلی گزینه ی Game Object>Create Other>Sphere را انتخاب کنید).

- حال شی Sphereرا از قسمت Hierarchy View درگ کرده و بر داخل Prefab ه Missile در Project Panel بیاندازید. دقت کنید که آیکون Prefab تغییر می کند. حال می توانید شی Sphere را از صحنه یا Hierarchy View پاک کنید.

نکته: هر شی ای که می خواهید در هنگام اجرای بازی ساخته شود، قانونا باید یک Prefab باشد.

نوشتن کد نارنجک انداز

شی FPS Controller، Prefab ای است که از چندین شی بازی و کامپوننت تشکیل شده است. خود FPS Controller شی سیلندری (Cylinder) است که بر روی محور Y می چرخد. پس اگر ما بخواهیم کد نارنجک انداز خود را به این شی اضافه کنیم، در آن حالت نمی توانیم به سمت بالا و پایین پرتاب کنیم. بنابراین ما کد خود را به شی Man Camera اضافه می کنیم که می تواند به هر جهت نگاه کند.

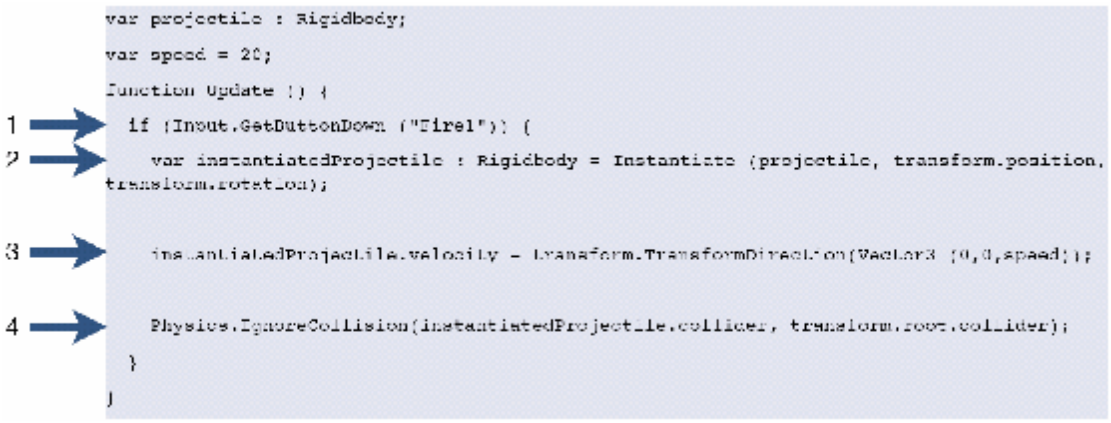
حال بیایید اولین کد جاوااسکریپت خود را ایجاد کنیم که تعیین کننده ی رفتار نارنجک انداز است.

- از منو اصلی گزینه ی Assets>Create>Javascript را انتخاب کنید. این کار فایل جاوا اسکریپت خالی ای را ایجاد می کند. در قسمت Project Panel نام این فایل NewBehaviorScript است. نام آن را به MissileLauncher تغییر دهید.

نکته: شما می توانید مسیر ادیتور کد دلخواه خود را به Unity معرفی کنید. برای این کار از منوی اصلی گزینه ی Edit>Pereferences را انتخاب کرده و در پنجره ی باز شده در قسمت External Script Editor، مسیر فایل اجرایی ادیتور خود را وارد کنید.

- پوشه ی جدیدی در قسمت Project View ایجاد کرده و نام آن را به Weapons Script تغییر دهید. از این به بعد تمام اسکریپت های مربوط به اسلحه را در این پوشه قرار می دهیم. حال اسکریپت MissileLauncherو همچنین Prefab ه Missile را به داخل این پوشه درگ کنید.

حال بیایید نگاهی به کد کامل MiissileLauncher بیاندازیم:



خب. چه کار می خواهیم بکنیم؟ خوبه. ما ابتدا می خواهیم بفهمیم بازیکن چه زمانی کلید شلیک را می زند تا اول یک شی نارنجک را بسازیم و سپس به سمت جایی که دوربین نگاه می کند به آن نیرویی بدهیم. بیایید بیشتر توضیح بدهیم. خط های بالا به ترتیب در زیر توضیح داده می شوند:

1. این خط می فهمد که چه زمانی بازیکن کلید شلیک را زده است. کلمه ی Fire1 بر روی کلیک چپ ماوس تنظیم شده است. برای تغییر تنظیم و تنظیمات دیگر از منوی اصلی گزینه ی Edit>Project Setting>Input را انتخاب کنید.
2. برای ساختن شی ای در Unity ما از تابع Instantiate استفاده می کنیم که سه پارامتر دارد: 1. شی ای که می خواهیم ساخته شود 2. مکان شی ای که قرار است ساخته شود 3. میزان چرخشی که می خواهیم داشته باشد.

اولین پارامتر این تابع کاملا منطقی است. ما آن را با یک متغیرعمومی پر کردیم تا بتوانیم در آینده شی دلخواه را در آن قرار دهیم. نوع این شی از نوع RigidBody است. زیرا که ما می خواهیم شی مورد نظر دارای رفتار فیزیکی باشد.

اولین پارامتر، projectile، شی ای است که ما می خواهیم بسازیم. پس آن شی چیست؟ خوب است. ما نوع این متغیر را عمومی تعریف کردیم تا بتوانیم به سادگی در محیط کاربری Unity هر شی ای که خواستیم را به عنوان این متغیر تعیین کنیم. با این روش هر وقت بخواهیم به سادگی می توانیم شی مورد نظر را عوض کنیم.

دومین پارامتر، transform.position، مکان شی ای است که اسکریپت به آن اضافه شده است. یعنی شی Launcher. سومین پارامتر نیز شبیه همین پارامتر است با این تفاوت که مقدار چرخش شی Launcher را برمی گرداند.

3. این قسمت شی نارنجک ما را به حرکت در می آورد. برای اینکار ما نیاز داریم تا به نارنجک سرعتی (Velocity) بدهیم. اما در کدام جهت؟ (X،Y یا Z؟). در قسمت Scene View بر روی شی FPS Controller کلیک کنید. نماد های حرکت ظاهر می شند (برای اطمینان کلید W را بزنید).یکی از آنها قرمز، یکی سبز و دیگری آبی است. قرمز محور X ا مشخص می کند. سبز محور Y را مشخص می کند و آبی هم محور Z را مشخص می کند. همچنین آبی نقطه ای که بازیکن به آن نگاه می کند را مشخص می کند. پس ما باید سرعتی در جهت محور Z به نارنجک بدهیم.

Velocity خاصیتی (propery) از InstantiatedProjectile است. ما این رو چطور باید بفهمیم؟ خوبه. InstantiatedProjectile از نوع RigidBody است پس اگر ما در API ی Unity نگاه کنیم خواهیم دید که Velocity از خواص (Property) آن است. همچنین به خواص دیگری که RigidBody دارد نگاه کنید. برای تعیین مقدار Velocity ما نیاز داریم تا سرعت را در هر جهت تعیین کنیم. البته این یک مقدار بحث نیاز دارد. اشیای سه بعدی از دو مدل مختصات استفاده می کنند: محلی (local) و جهانی (World). در فضای محلی (Local Space) مختصات وابسته به شی ای هستند که روی آن کار می کنید. در فضای جهانی (World Space) مختصات مستقل هستند. یعنی همیشه یک جهت برای تمام اشیاء دارند. RigidBody.velocity باید در فضای جهانی تعیین گردد. بنابراین وقتی مقدار Velocity را تعیین می کنید، شما نیاز دارید تا جهت آن در محور Z در فضای محلی را به جهت آن در فضای جهانی مربوط به آن تبدیل کنید. شما این کار را با استفاده از تابع transform.TransformDirection که یک پارامتر از نوع Vector3 می گیرد انجام می دهید. متغیر Speed هم عمومی تعیین شده است تا آن را در محیط GUI ه Unity تعیین کنید.

4. این خط برخورد (Collision) بین نارنجک و کاربر را غیر فعال می کند. اگر ما این کار را انجام ندهیم ممکن است نارنجک هنگاه ساخته شدن با بازیکن برخورد کند. این تابع را در API ه Unity در قسمت IgnoreCollision بررسی کنید.

- کد بالا را در فایل جاوا اسکریپت با نام MissileLauncher وارد کنید. فراموش نکنید که آن را ذخیره کنید.

- فایل MissileLauncher را به شی Launcher در زیر مجموعه ی FPS Controller اضافه کنید. شما می توانید برای اطمینان از اضافه شدن کد به شی Launcher، شی را انتخاب کرده و در قسمت Inspector View ببینید که اسکریپت نمایان شده است یا خیر.

همانطور که می بینید شی نارنجکی که ما قبلا ساخته بودیم به متغیر projectile در اسکریپتمان اضافه نشده است. این کار در محیط GUI ه Unity انجام می شود. متغیر projectile ما از نوع RigideBody است پس ما ابتدا باید مطمئن شویم که شی نارنجک ما دارای کامپوننت RigideBody است.

- برای این کار، ابتدا شی نارنجک (missile) را از قسمت ProjectView انتخاب کرده و از منوی اصلی گزینه ی Components>Physics>RigideBody را انتخاب کنید. این کار کامپوننت RigidBody را که می خواهیم، به شی نارنجک اضافه می کند. ما این کار را انجام دادیم تا نوع شی مان با نوع متغیری که در اسکریپت تعیین کردیم یکی باشند.
- حال برای وصل کردن شی نارنجک با متغیر projectile، ابتدا شی Launcher را انتخاب کرده و حال متغیر Projectile را در قسمت Inspector View در زیرمجموعه ی اسکریپت MissileLauncher می بینید. تمام کاری که باید انجام دهیم این است که Prefab نارنجک را از Project View انتخاب کرده و به داخل متغیر projectile درگ و رها کنید.
- حال اگر بازی را اجرا کنید می توانید با کلیک چپ ماوس کره ی کوچکی را به طرف جلو پرتاب کنید. این کار را تکرار کنید.

7. انفجارهای نارنجک

حال ما می خواهیم که وقتی که نارنجک با چیزی برخورد کرد ، انفجاری رخ دهد. برای این کار ما باید رفتاری را برای شی نارنجک تعریف کنیم. پس باید فایل اسکریپتی را ایجاد کنیم.

- فایل اسکریپتی را با انتخاب گزینه ی Assets>Create>JavaScript از منوی اصلی ایجاد کرده و نام آن را به Projectile تغییر دهید. فایل اسکریپت ایجاد شده را به داخل پوشه ی Weopens در قسمت Project View درگ کنید.

خب ما می خواهیم که اسکریپت Projectile چه کاری برای ما انجام دهد؟ ما می خواهیم که این اسکریپت تشخیص دهد چه زمانی نارنجک با چیزی برخورد کرده و سپس انفجاری را در آن نقطه ایجاد کنیم. به کد کامل آن دقت کنید:

```
var explosion : GameObject;

1 function OnCollisionEnter (collision : Collision) {
2     var contact : ContactPoint = collision.contacts[0];

      var rotation = Quaternion.FromToRotation(Vector3.up, contact.normal);
3     var instantiatedExplosion : GameObject = Instantiate (explosion, contact.point,
          rotation);
4     Destroy(gameObject);
}
```

1. هر کدی که در داخل تابع OnCollisionEnter قرار گیرید، وقتی که شی ای که اسکریپت به آن اضافه شده است با شی دیگری برخورد کند، اجرا می شود.
2. کار اصلی که ما می خواهیم در این تابع انجام دهیم این است که انفجاری در نقطه ای که شی با شی دیگری برخورد کرده است ایجاد کنیم. بنابراین برخورد کجا صورت می گیرد؟ تابع OnCollisionEnter کلاسی با نام Collision دارد که در بردارنده ی این اطلاعات است. نقطه ای که برخورد در آن صورت می گیرد در متغیر ContactPoint ذخیره می شود.
3. جایی که ما انفجار خود را در آن ایجاد می کنیم. ما می دانیم که برای ایجاد (Instantiate) هر شی نیاز به سه پارامتر داریم: 1. شی ای که باید ایجاد شود. 2. مکان شی در دنیای سهد بعدی. 3. میزان چرخش شی در دنیاس سه بعدی.

ما این را با شی ای از نوع Particle System بعدا تعیین می کنیم. همچنین ما می خواهیم مقدار آن را از محیط GUI ه Unity تعیین کنیم پس نوع آن را عمومی تعیین می کنیم.

دومین پارامتر، نقطه ی برخورد است که ما مقدار آن را در خط 2 تعیین کردیم.

سومین پارامتر چرخش شی ماست که نیاز به مقداری توضیح دارد. ما می خواهیم جهت محور Y ه شی انفجار ما در جهت نورمال سطحی که نارنجک با آن برخورد می کند باشد. این به این معنیست که برای اشیایی مانند دیوار به سمت خارج و اشیایی مانند کف زمین به سمت بالا باشد. پس کاری که ما می خواهیم انجام دهیم،

تغییر مکان محور Y ه شی انفجار در فضای محلی به نورمال سطح شی ای که نارنجک با آن برخورد می کند در فضای جهانی، است. این نکته ضروری است که چرخش محور Y ه انفجار با جهت نورمال شی برخوردی برابر باشد.

4. در نهایت می خواهیم وقتی نارنجک با شی ای برخورد کرد، از صحنه حذف شود. این کار را با استفاده از تابع Destroy() انجام می دهیم که یک پارامتر gameObject دارد که gameObject شی ای است که فایل اسکریپت به آن اضافه شده است.

- کد را در داخل اسکریپت Projectile وارد کنید و آن را ذخیره نمایید.
- اسکریپت Projectile را به شی نارنجک (Missile) اضافه کنید.

حال ما می خواهیم شی انفجاری که در زمان برخورد ایجاد می شود را بسازیم.

- ابتدا یک Prefab (با نام Explotion) برای نگه داری محتوی (Asset) انفجار بسازید. آن را به داخل Scene View درگ کنید تا بتوانید آن را ببینید. دقت کنید که سیستم ذرات فقط تکرار می شود و برای دیدن نمای نهایی باید بازی را اجرا کنید.
- محتویات استاندارد Unity (Standard Assets) در بردارنده ی یک Prefab انفجار خوب با نوری در اطراف آن است. از مسیر Standard Assets>Particles ، Prefab ه Explotion را به داخل Hierarchy View درگ کنید.
- بعد از اینکه Prefab ه Explotion را در Hierarchy Vew دیدید ، آن را به داخل Prefab ه خود در Project View درگ کنید.

حال ما می توانیم به شی نارنجک انفجار را اضافه کنیم.

- مطمئن شوید که ش نارنجک (Missile) انتخاب شده است. حال شی انفجاری که ساخته اید را به داخل متغیر Explotion در قسمت Inspector View درگ و رها کنید.

تعیین رفتار انفجار

حال ما یک اسکریپت جداگانه برای تعیین رفتار انفجار می سازیم.

- یک فایل اسکریپت جدید بسازید و نام آن را Explotion بگذارید و سپس آن را در فولدر Weopens در Project View قرار دهید. بر روی اسکریپت دوبار کلیک کنید تا برای ویرایش باز شود.

نوع رایج دیگری از توابع که در اسکریپت ها استفاده می شود، تابع Start() است. کدی که در این تابع قرار گیرد فقط یکبار در زمانی که شی ایجاد می شود، اجرا می شود. تنها کاری که ما فعلا می خواهیم انجام دهیم این است که انفجار را بعد از زمان خاصی از بین ببریم. برای این کار ما از تابع Destroy() استفاده می کنیم که دومین پارامتر آن میزان زمان را تعیین می کند.

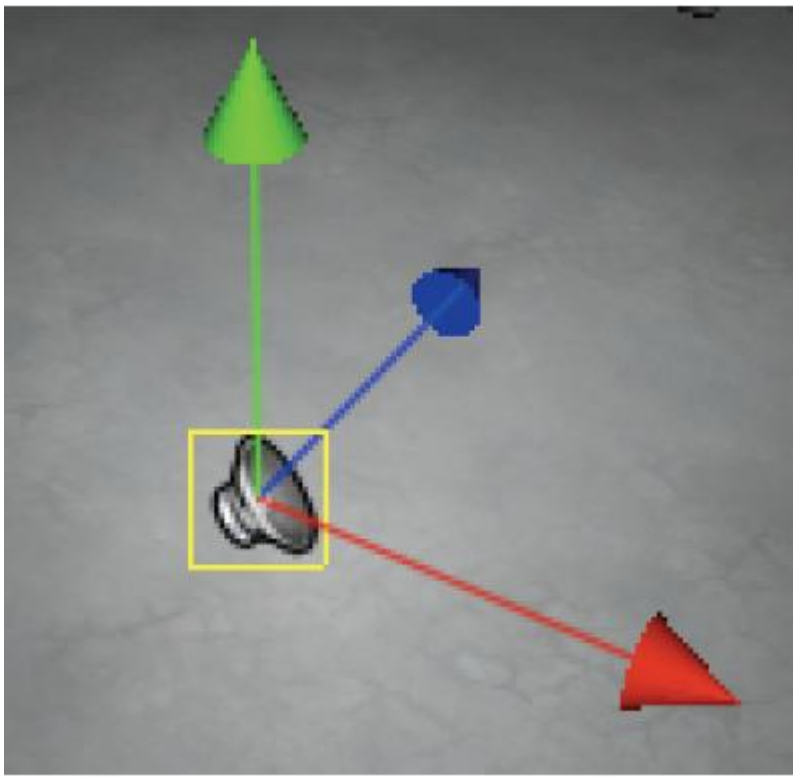
```
var explosionTime = 1.0;

function Start() {
    Destroy (gameObject, explosionTime);
}
```

متغیر explotionTime عمومی تعریف شده است تا بتوانید مقدار آن را در محیط کاربری Unity به سادگی تغییر دهید.

- کد بالا را در فایل اسکریپت Explotion وارد کنید و تابع Update() را حذف کنید.
- فایل اسکریپت بالا را به شی Explotion اضافه کنید. برای این کار ابتدا شی Explotion را انتخاب کرده و سپس از منوی اصلی گزینه ی Components>Scripts>Explotion را انتخاب کنید.
- بازی را اجرا کنید. حال بعد از پرتاب نارنجک، انفجاری در نقطه ی برخورد نارنجک ایجاد می شود و بعد از زمان تعیین شده از بین می رود.

8.افکت های صوتی

دنیای بازی ما تاکنون خیلی ساکت بوده است. بیایید مقداری افکت صوتی به آن اضافه کنیم.

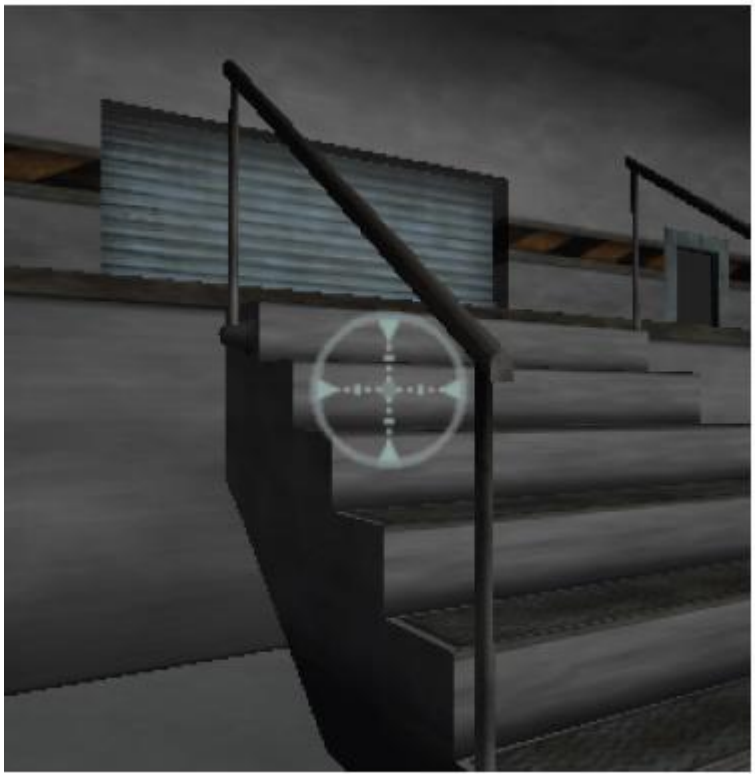


ایتدا بیایید یک کلیپ صوتی به Prefab ه انفجار اضافه کنیم.

- **برای اینکه بتوانیم فایل صوتی بـه شـی انفجـار اضـافه کنـیم، بایـد بـه آن کامپوننـت Audio Source اضافه کنیم. برای اینکار در حالی که شی انفجـار انتخـاب اسـت، از منوی اصلی گزینـه ی Components>Audio>Audio Source را انتخـاب کنیـد. حال می بینید که یکی از خواص (Property) این کامپوننت Audio Clip است.**
- **فایل صوتی Sounds>RocketLauncherImpact را از قسمت Project View بـه داخل متغیر Audio Clip مربوط به شی انفجار در قـسمت Inspector View درگ کنید. Unity می تواند فایل های صوتی aiff. و بیـشتر فرمـت هـای فـشرده شـنده ر پخش کند.**
- **بازی را اجرا کنید. حال می بینید که وقتی نارنجک منفجر می شود صـدای انفجـار بـه گوش می رسد.**

9. اضافه کردن GUI

حال بیایید به بازی مان یک GUI اضافه کنیم که بیشتر با نام HUD در بازی ها شـناخته مـی شود.



اضافه کردن مکان نمای ماوس

- از قسمت Project View فایل Textures>aim را انتخاب کرده و سپس از منوی اصلی گزینه ی Game Object>Create Other>GUI Texture را انتخاب کنید.

دقت کنید که تصویر aim در وسط صفحه ی بازی نمایان شده است و متغیر آن نیز در قسمت Inspector View اضافه شده است.

- بازی را اجرا کنید. اگر شی نارنجک از وسط aim پرتاب نمی شود، باید شی Launcher خود را مقداری جابجا کنید تا له مکان مناسب برود.

10. فیزیک

حال ما اشیایی در محیط خود می خواهیم که به صورت واقعیت رفتار کنند. این هدف با اضافه کردن فیزیک به اشیاء محقق می شود. حال ما اشیایی به صحنه اضافه خواهیم کرد که وقتی با نارنجک برخورد کردند پرتاب شوند. برای این کار نیاز داریم تا بعضی از اصطلاحات را توضیح دهیم.

Update

قبلا ما کدهای خود را در داخل تابع Update وارد می کردیم، پس ما می توانستیم کد خود را در هر فریم یکبار اجرا کنیم. برای مثال فهمیدن اینکه چه زمانی کاربر کلید آتش (Fire) را فشار می داد. اما سرعت فریم همیشه یکسان نیست و بستگی به پیچیدگی محیط بازی دارد. این بی قاعدگی سرعت فریم باعث ایجاد فیزیک ناپایداری می شود. بنابراین زمانی که شما بخواهید فیزیک شی خود را در صحنه آپدیت کنید، باید کدهای خود را در داخل تابع FixedUpdate() وارد کنید. متغیر deltaTime در unity برای به دست آوردن زمان بین دو فریم متوالی استفاده می شود.

تفاوت های عمومی بین تابع FixedUpdate و Update به شرح زیر است:

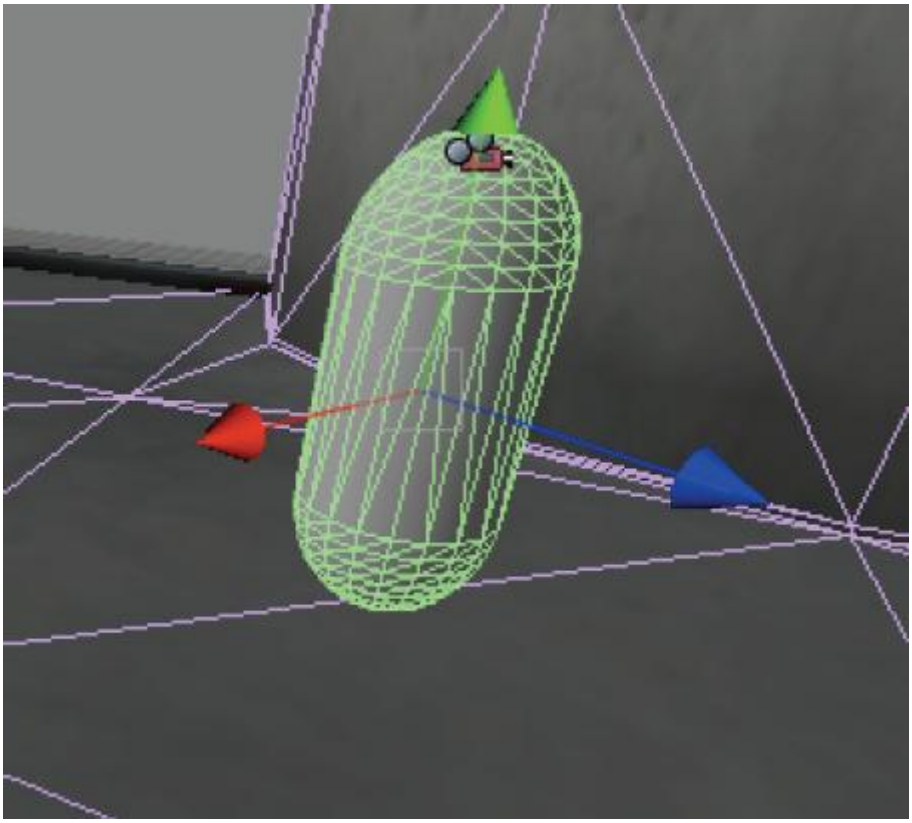
- Update() : کدهای داخل این تابع معمولا برای آپدیت کردن رفتار اشیاء و منطق بازی و غیره به کار می رود. مقدار deltaTime در داخل این تابع همیشه یکسان نیست.
- FixedUpdate() : کدهای داخل این تابع برای آپیدیت RigidBody و رفتار فیزیکی اشیاء به کار می رود. deltaTime همیشه مقدار ثابتی را در این تابع برمی گرداند.

مدار بسامد(تکرار) تابع FisexUpdate() با استفاده از خاصیتی به اسم Fixed Timestep تعیین می شود که در قسمت Edit>Project Setting>Time قابل دسترسی است.

همچنین می توان آن را تغییر داد. این خاصیت در بردارنده ی زمان در ثانیه ها است. برای بدست آوردن مقدار فریم بر ثانیه، مقدار معکوس این خاصیت را بگیرید.

زمانی که در حال تغییر مقدار Fixed Timestep هستید (عامل موثر فریم بر ثانیه)، مواظب باشید که بالانس را رعایت کنید. سرعت فریم بیشتر، فیزیک بهتری را به ارمغان می آورد ولی کارایی بازی پایین تر می رود. مطمئن شوید که بازی با سرعت مناسب و فیزیک به صورت واقعی اجرا می شود.

آخرین اصطلاحی که باید توضیح دهیم Yield است. فکر کنید که این همان کلمه ی کلیدی برای نگه داشتن (Pause) تابع است.

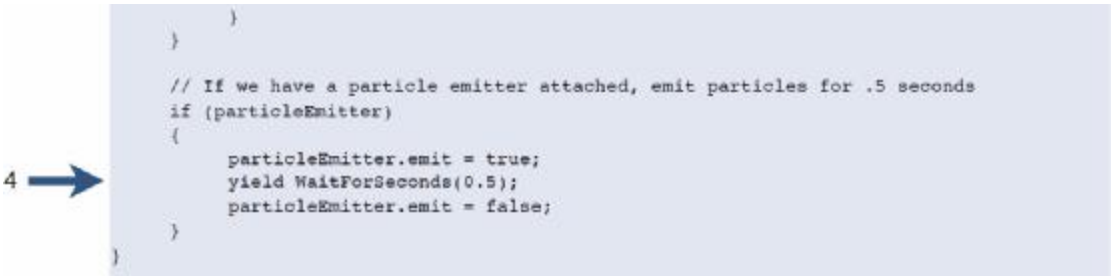


خب بیایید به بازی خودمان بازگردیم. چه کاری می خواهیم بکنیم؟

1. **به کاربر اجازه بدهیم تا نارنجکی را شلیک کند (که این کار را قبلا انجام داده ایک).**
2. **اگر شی نارنجک با** rigidbody **دیگری برخورد کرد، تصمیم بگیریم که کدام اشیاء ای که در اطراف آن وجود دارد دارای خاصیت** rigidbody **است.**
3. **برای هر** rigidbody **ای که در محدوده ی موج انفجار قرار دارد، به عنوان تاثیر نارنجک، به آن نیرویی در جهت بالا بدهیم.**

بیایید به کد اصلاح شده ی اسکریپت Explotion **نگاهی بیاندازیم:**

```
var explosionTime = 1.0;
var explosionRadius = 5.0;
var explosionPower = 2000.0;

function Start () {
     // Destroy the explosion in x seconds,
     // this will give the particle system and audio enough time to finish playing
     Destroy (gameObject, explosionTime);

     // Find all nearby colliders
1    var colliders : Collider[] = Physics.OverlapSphere (transform.position,
           explosionRadius);

     // Apply a force to all surrounding rigid bodies.
2    for (var hit in colliders) {
          if (hit.rigidbody)
          {
3              hit.rigidbody.AddExplosionForce(explosionPower, transform.position,
                    explosionRadius);
```



```
        }
    }

    // If we have a particle emitter attached, emit particles for .5 seconds
    if (particleEmitter)
    {
        particleEmitter.emit = true;
        yield WaitForSeconds(0.5);
        particleEmitter.emit = false;
    }
}
```


1. **این خط تصمیم می گیرد که کدام اشیاء ای که دارای** collider **هستند در اطاف شی نارنجک قرار دارند. تابع** Physics.OverlapSphere() **سه پارامتر می گیرد. یک مکان سه بعدی و یک شعاع. و آرایه ای از** Collider **هایی که در کره ی تعیین شده قرار دارند را بازمی گرداند.**
2. **بعد از اینکه آرایه به دست آمد، تمام** Collider **هایی که خاصیت** Rigidbody **دارند، در جهتی مشخص نیرویی بهشان وارد می شود.**
3. **این قسمت نیرویی (نیروی انفجار) در جهت بالا (محور** Y**) در مکانی که نارنجک فرود می آید ، وارد می کند. (**transform.position**). اما چون نیروی انفجار با دور شدن از آن نیروی کمتری را وارد می کند، پس در هر فاصله ای از انفجار نباید نیرویی یکسان وارد شود.** Rigidbody **هایی که در فاصله ی بیشتری از انفجار قرار دارند، نیروی کمتری را نسبت به** Rigidbody **های نزدیک به انفجار دریافت می کنند. این تاثیر در این تابع محاسبه و تعیین می شود. به دلیل اینکه دو متغیر** explotionPower **و** explotionRadius **به صورت عمومی تعریف شده اند، پس به راحتی می توان در محیط کاربری** Unity **میزان قدرت انفجار را تعیین کرد.**
4. **اگر سیستم ذراتی (**Particle System**) به شی انفجار ما اضافه شده باشد، شروع به ساتع شدن می کند (**emit**) و بعد از** 0,5 **ثانیه ساتع شدن، دوبازی خاموش می شود.**

- **فایل اسکریپت** Explotion **را با کدهای بالا آپدیت کرده و آن را ذخیره کنید.**
- **بیایید اشیایی برای پرتاب کردن بسازیم. یک مکعب بسازید و به آن** Rigidbody **اضافه کنید. چندین مکعب دیگر از روی این مکعب کپی گرفته و در محیط پخش کنید.**
- **بازی را اجرا کرده و نارنجک به سمت مکعب ها پرتاب کنید. حال می بینید که مکعب ها نیرویی به سمت بالا دریافت می کنند. نارنجک را دورتر از مکعب ها بیاندازید تا ببینید که نیرو را به یک اندازه و جهت دریافت نمی کنند.**

- **صحنه ی خود را ذخیره کنید.**

خلاصه

این آموزش نحوه ی ساختن یک بازی اول شخص ساده را به شما آموزش داد. حال شما باید با اصطلاحات زیر آشنا شده باشید.

- Asset
- Camera
- Light
- Viewport
- Prefab
- Collider
- Rigidbody
- Javascript
- GameObject
- Update\FixedUpdate
- Local View\World View

قسمت های بعدی این آموزش به شما نحوه ی استفاده از تکنیک های پیشرفته تری از جمله هوش مصنوعی (AI) و چندین اصلحه را آموزش خواهد داد.


یا مهدی ادرکنی


www.hayat-montazaran.blogfa.com
vahid-a