شروع بازی سازی با موتور بازی یونیتی

نویسنده : دکتر اِدوارد لاویِری

به کوشش : عباسعلی طهماسبی

www.bazi-dev.ir

آذر ۹۵

درباره نویسنده

دکتر اِدوارد لاویِری یک طراح و سازنده بازی باتجربه است که دارای سابقه علمی طولانی می باشد . او دکترای علوم کامپیوتر را از دانشگاه Colorado Technical گرفته است و دارای مدارک علمی در زمینه سیستم های مدیریت اطلاعات از دانشگاه Bowie State است . او دوره های مربوط به کامپیوتر را از سال ۲۰۰۲ تدریس می کند و در حال حاضر مشغول تدریس در دانشگاه Southern New Hampshire است . او موسس و رئیس استادیو three19 است و همیشه در حال توسعه نرم افزار بوده است و از یونیتی به عنوان یکی از اولین ابزارهای ساخت و توسعه بازی ها استفاده کرده است . علاقه او ساختن بازی های آموزشی و نرم افزارهای موبایل می باشد .

مقدمه

با همه گیرشدن بازی ها و سود سرشار موجود در این صنعت نوپا ، تمایل به کشف چگونگی استفاده از نرم افزارهای توسعه مدرن و به روز ، زیاد دور از انتظار نیست . تعداد زیادی از ابزارهای نرم افزاری در دسترس هستند که به سازندگان و توسعه دهندگان کمک می کنند ؛ بازی های شگفت انگیزی را برای کنسول ها ، وب ، کامپیوترهای دسکتاپ و دستگاه های موبایل تولید کنند . در این میان ، موتورهای بازی یکی از قدرتمندترین ابزارهای موجود هستند . موتور بازی Unity 3D یکی از این موتورهای بازی مدرن است . بسیاری از استادیوهای بازی سازی و نیز سازندگان مستقل بیشماری ، از این موتور برای تولید بازی های محبوب دوبعدی و سه بعدی استفاده می کنند . با استفاده از ورژن رایگان این موتور و انتشار Unity 5 ؛ بهترین زمان برای شروع استفاده از Unity فرارسیده است .

این کتاب ، یکی از محبوب ترین موتورهای بازی در دسترس را بررسی می کند . در اینجا ، شما را از میان پروسه ساختن یک بازی سه بعدی راهنمائی می کنیم . این پروسه از دانلود موتور بازی Unity شروع می شود و تا انتشار و عرضه بازی شما ؛ ادامه خواهد داشت . مطمئنم از مطالعه مباحث جذابی مثل کاراکترهای قابل کنترل و انیمیشن ها ، لذت خواهید برد . چه شما به عنوان یک سازنده بازی ؛ به تازگی شروع به کار کرده باشید یا اینکه تجربه استفاده از موتور بازی Unity یا موتورهای دیگر را داشته باشید ؛ این کتاب برای شما یک تور آموزشی از ساختن بازی ها به وسیله Unity 5 ، فراهم کرده است . همراه با توضیحات واضح ، نکته ها و تصاویر کمکی فراوان ، یک گام به گام مفصل در زمینه ساختن بازی ها ؛ خواهید داشت .

این کتاب از یک شیوه تمرینی همراه با فعالیت های آماده ، برای آموزش Unity 5 استفاده می کند . با پیشرفت هر فصل ، ساختن یک بازی سه بعدی به نام Little Farmer Colt را تکمیل خواهید کرد . همراه با تولید این بازی ؛ خصوصیات کلیدی Unity 5 مثل ایجاد محیط بازی ، انیمیت کردن کاراکترها ، اسکریپت نویسی و ... را یاد می گیرید . تمام مِش ها (mesh) ، مدل ها ، بافت ها ، انیمیشن ها و منابع دیگر ؛ در سایت کتاب در دسترس هستند .

با تکمیل درس های موجود در کتاب ، اطمینان کامل به دست می آورید تا خودتان با استفاده از Unity 5 ، شروع به ساختن بازی کنید .

این کتاب چه مطالبی را بررسی می کند ؟

فصل ۱ : آشنائی با یونیتی - موتورهای بازی را بررسی کرده و آنها را با یونیتی مقایسه می کند تا با امکانات موتور بازی یونیتی آشنا شوید . با نحوه دانلود و نصب یونیتی ورژن ۵ آشنا خواهید شد . درخصوص محیط کاری یونیتی و ابزارهای اصلی آن مثل ابزارهای تغییر شکل و دوربین ها نیز آموزش خواهید دید .

فصل ۲ : ایجاد محیط بازی - طراحی بازی را به شما یاد می دهد و در ادامه به طراحی بازی *Little Farmer Colt* که در طول این کتاب ساخته می شود ، می پردازد . شما با ساختن زمین ، اضافه کردن کوه ها ، درخت ها ، دریاچه و آسمان ؛ شروع به ایجاد محیط بازی خواهید کرد .

فصل ۳ : مدیریت منابع بازی - درباره منابع مورد استفاده برای ساختن بازی ها در یونیتی است . شما در زمینه منابع ، محل به دست آوردن آنها ، نحوه ایجاد آنها و نحوه وارد کردن آنها به داخل بازی ؛ آموزش می بینید . همچنین درباره یک نرم افزار رایگان ایجاد اشیاء سه بعدی ، آموزش می بینید که با آن می توان اشیاء بازی مناسب برای استفاده در یونیتی را ساخت .

فصل ۴ : متحرک سازی کاراکترهای بازی - درخصوص انیمیشن های یونیتی و کنترل کننده های بازیکن است . خواهید دید که کاراکترهای بازی ما جان می گیرند و در ضمن ، به کاربر بازی توانائی کنترل بازیکن را خواهید داد .

فصل ۵ : اسکریپت نویسی بازی ـ به شما نحوه اسکریپت نویسی را آموزش می دهد . فصل با مقدمه ای به C# و MonoDevelop آغاز می شود . در طول فصل ، تجربه خوبی در زمینه ایجاد اسکریپت های C# برای بازی هایتان ، کسب خواهید کرد .

فصل ۶ : اضافه کردن رابط کاربری گرافیکی ـ شامل اطلاعاتی کلیدی درباره رابط های گرافیکی و اهمیت آنها برای بازی هاست . شما با سیستم رابط کاربری یونیتی آشنا می شوید و درخصوص ایجاد HUD ، نقشه های کوچک روی صفحه نمایش و منوهای بازی ؛ آموزش می بینید .

فصل ۷ : بهینه سازی بازی ـ به شما چند راه نشان می دهد تا به وسیله آنها ، بازیتان زیباتر شده و بهینه تر کار کند . با اهمیت جلوه های ویژه صوتی و تصویری و نحوه ایجاد آنها آشنا می شوید . با چگونگی ایجاد افکت های صوتی ، سایه ها ، نورپردازی و رندرینگ دوربین آشنا می شوید . در ضمن نحوه بهینه سازی اسکریپت ها نیز بررسی می شود .

فصل ۸ : بهبود بازی و مدیریت پروژه ـ آخرین فصل است و در اینجا پیشنهاداتی برای بهترشدن بازی Little Farmer Colt ارائه می شود . همچنین درخصوص مدیریت پروژه های یونیتی و چند مبحث پیشرفته مثل سیستم های ذره ای صحبت خواهیم کرد .

ابزارهای مورد نیاز کتاب

برای دنبال کردن مثال های کتاب ، به یک کپی از Unity 5 یا بالاتر نیاز دارید . این نرم افزار به صورت رایگان در آدرس زیر در دسترس است :

http://unity3d.com/get-unity/download

می توانید از سیستم عامل های ویندوز یا Mac استفاده کنید . در ضمن برای دانلود منابع گرافیکی ارائه شده در کتاب ، به اینترنت نیاز خواهید داشت .

این کتاب مخصوص کیست ؟

این کتاب برای افرادی نوشته شده که به تازگی به سراغ Unity آمده اند یا تجربه کار با ورژن های قبلی Unity 5 را دارند . اگر می خواهید Unity 5 را یاد بگیرید یا اینکه می خواهید ببینید بازی ها چطور با استفاده از یک موتور بازی پیشرفته ، ساخته می شوند ؛ این کتاب مخصوص شماست .

برای دانلود کدها و گرافیک های استفاده شده در کتاب به سایت www.bazi-dev.ir مراجعه کنید .

فهرست مطالب

فصل ۱ : آشنائی با یونیتی

هدف این فصل ، آشنائی شما با Unity و محیط کاری آن است . ما با بحث درباره موتورهای بازی و مقایسه Unity شروع می کنیم . درخصوص نحوه دانلود Unity و فرایند نصب آن ، آموزش خواهید دید . سپس درباره پروژه Unity و ساختار فایل آن در داخل و خارج از Unity بحث خواهیم کرد .

همچنین نگاهی به چهره و محیط Unity نیز خواهیم داشت . ما طرح بندی و چینش های مختلف موجود در یونیتی را تست خواهیم کرد . من هر یک از نماهای موجود در یونیتی را توضیح خواهم داد و هدف هریک از آنها را بررسی خواهم کرد . ما با استفاده از ابزارهای تغییرشکل ، زمان بیشتری را روی نمای Scene صرف می کنیم تا شما در هنگام استفاده از آنها ، راحت تر باشید . این فصل با بحث بر روی نحوه استفاده از دوربین ها در یونیتی ، به پایان خواهد رسید .

در طول این فصل :

با موتور یونیتی آشنا خواهید شد .

قادر به دانلود و نصب یونیتی خواهید شد .

با پروژه های یونیتی آشنا می شوید .

با طرح بندی و چینش های یونیتی آشنا می شوید .

با هر یک از نماها ، آشنا می شوید .

با استفاده از نمای Scene ، قادر به گشت و گذار در بازی خواهید شد .

با ابزارهای تغییر شکل ، آشنا می شوید .

با دوربین ها ، آشنا می شوید .

چرا یونیتی ؟

برای سازندگان بازی های دوبعدی و سه بعدی ، چندین موتور بازی مختلف در دسترس است . پس چرا یونیتی را انتخاب کنیم ؟ ابتدا اجازه دهید درباره بعضی از رقبای این موتور ، صحبت کنیم .

موتورهای Unreal و CryEngine از جمله موتورهای بازی قدرتمند و در دسترس هستند . بسیاری از استادیوهای بازی سازی ، در طول سالیان متمادی ، با استفاده از این موتورها ، بازی های سه بعدی ساخته اند . قیمت و پیچیدگی استفاده از اینها کافیست تا باعث شود ، سازندگان بازی مستقل و استادیوهای بازی سازی کوچک ، به دنبال راه های دیگری باشند .

اما نظرتان درباره موتورهای بازی ساده چیست ؟ Game Salad ، Game Maker و Construct 2 ابزارهائی هستند که یادگرفتن و استفاده از آنها آسان است . درواقع ، با استفاده از این موتورها می توانید بدون نیاز به برنامه نویسی یا حتی نوشتن یک خط اسکریپت ، بازی های ساده ای را بسازید . همه اینها ، موتورهای بازی دوبعدی هستند و امکانات موتورهای حرفه ای را ندارند .

البته آنچه گفتیم باعث نمی شود یونیتی در وسط موتورهای ساده و موتورهای حرفه ای قرار گیرد . بلکه با توجه به طیف امکانات یونیتی ، این موتور درکنار موتورهای حرفه ای جای می گیرد . شکل زیر را ببینید :

اجازه دهید جدول زیر را بررسی کنیم تا این موتورها را بر حسب امکانات ، منحنی یادگیری و زبان های برنامه نویسی مورد استفاده برای ساختن بازی ، مقایسه کنیم :

موتور بازی	منحنی یادگیری	زبان	2D/3D
Unity	۴ از ۵	C#, JS, Boo	هر دو
Unreal	۵ از ۵	C++	3D
CryEngine	۵ از ۵	C++, Lua	هر دو
Game Salad	۱ از ۵	GML	2D
Game Maker	۱ از ۵	N/A	2D
Construct 2	۲ از ۵	JS	2D

موتور بازی یونیتی ، در نقطه وسط امکانات پیشرفته و سختی یادگیری قرار می گیرد . این باعث شده است تا این موتور ، انتخاب بسیاری از سازندگان باشد . همانطور که بعدا در این فصل خواهید دید ، هنگام ایجاد یک پروژه بازی ؛ می توانیم دوبعدی یا سه بعدی بودن را انتخاب کنیم . دلایل دیگری برای استفاده از یونیتی درمقایسه با موتورهای دیگر وجود دارد :

۱. امکان برنامه نویسی با C# ، جاوااسکریپت ، Boo یا هر ترکیبی از این زبان ها در یک بازی .

نکته

هرچند یونیتی ، اسکریپت نویسی در یک بازی را با استفاده از زبان های مختلفی (C# ، جاوااسکریپت و Boo) پشتیبانی می کند ؛ اما این کار را توصیه نمی کنیم . بهترین شیوه ؛ انتخاب یک زبان برنامه نویسی و استفاده از آن در سراسر بازی است .

۲. امکان تست بازی ها در نماهای (پنجره های) مجزا بدون اینکه مجبور به خروج از محیط موتور باشیم .

۳. امکان انجام تغییرات در بازی در حالیکه بازی در حال اجراست .

۴. یک بار ساختن بازی و سپس عرضه و انتشار آن بر روی پلتفرم های موبایل (iOS ، اندروید ، Windows Phone 8 و BlackBerry 10) ، کامپیوترهای دسکتاپ (Mac ، ویندوز و لینوکس) ، وب (سافاری ، فایرفاکس ، کروم و اینترنت اکسپلورر) و کنسول های بازی (ایکس باکس وان ، ایکس باکس 360 ، پلی استیشن 3 ، پلی استیشن 4 ، PlayStation Vita و Wii U)

علاوه بر دلایل ذکر شده ، موتور بازی یونیتی رایگان است . البته یک ورژن Pro وجود دارد که استفاده از آن هزینه بر است اما اگر می خواهید بدون پرداخت هیچ هزینه ، بازی سازی را با یک موتور بازی شروع کنید ، یونیتی مخصوص شماست !

دانلود یونیتی

با استفاده از مراحل زیر ، یونیتی را دانلود و نصب کنید :

۱. به آدرس http://www.unity3d.com بروید .

۲. در قسمت بالای صفحه لینک دانلود (Download) را می بینید . با کلیک این لینک ، به صفحه دانلود می روید :

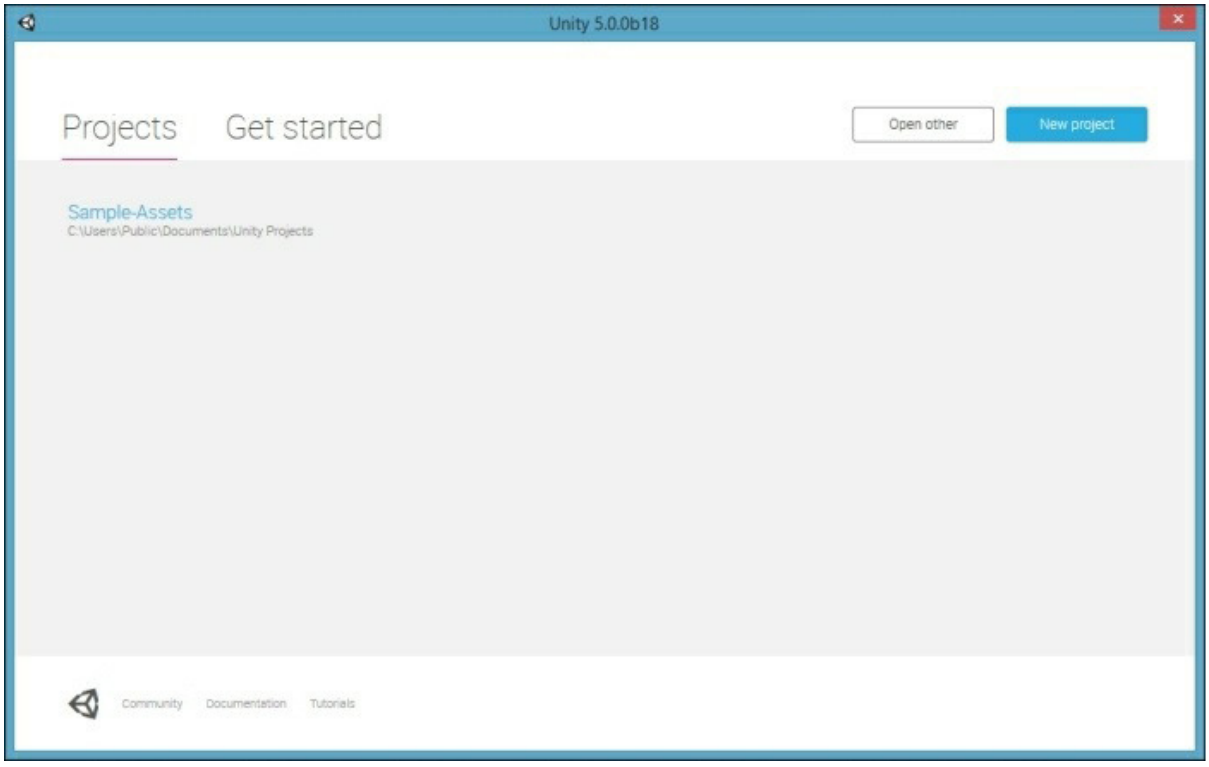
۳. در صفحه دانلود ، یک دکمه Download Unity می بینید که شامل ورژن فعلی موتور است مثلا به شکل Download Unity 5.1 است . صفحه دانلود ، سیستم عامل مورد استفاده کامپیوتر شما را می شناسد و ورژن مناسب را دانلود خواهد کرد . دکمه Download را کلیک کنید . فایل نصب خیلی بزرگ است و بیشتر از 1GB است و زمان زیادی را برای دانلود می گیرد مخصوصا اگر سرعت اینترنت شما ، کند باشد .

۴. بعد از پایان دانلود ، فایل installation را اجرا کرده و تنظیمات پیش فرض را قبول کنید .

۵. حالا آماده اید برای اولین بار یونیتی را اجرا کنید . آیکون Unity را دوبار کلیک کنید .

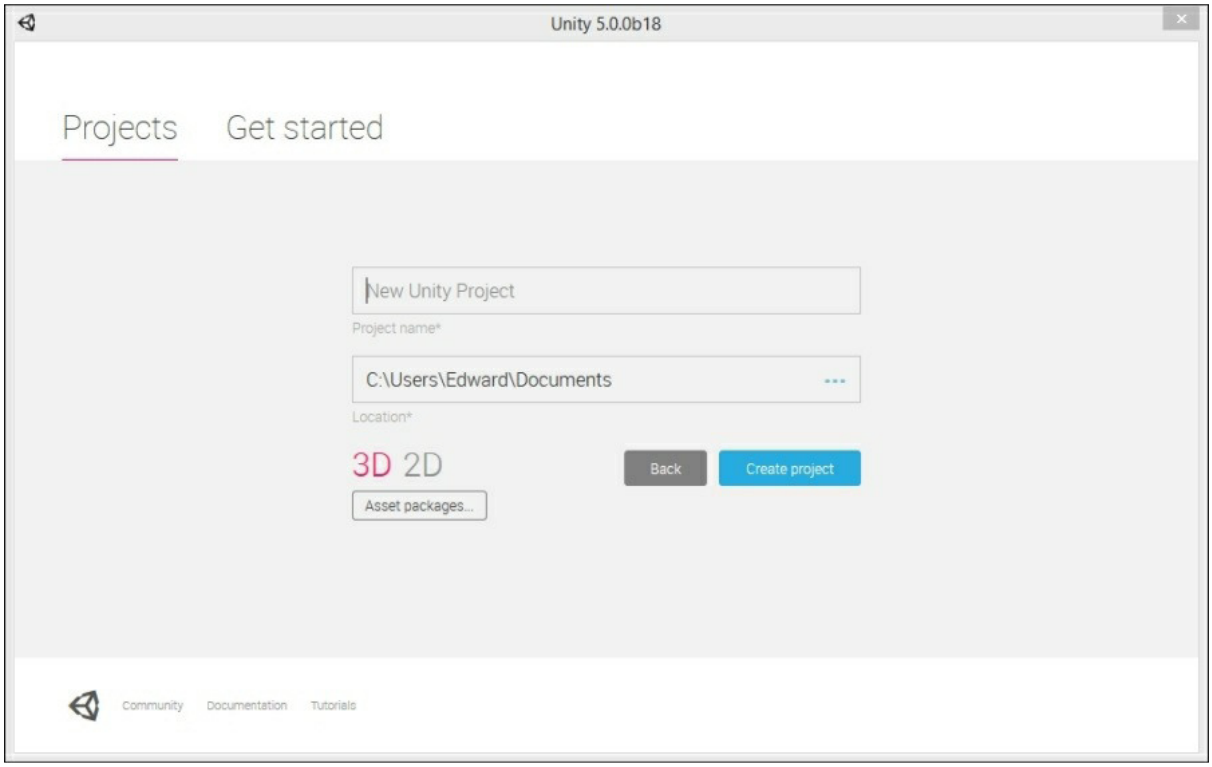
پروژه ها

اولین کاری که برای استفاده از یونیتی باید انجام دهیم ، ایجاد یک پروژه است . یونیتی از پروژه ها برای ساختن بازی ها استفاده می کند . پروژه ها ، شامل ساختارهائی برای همه اسکریپت ها ، اشیاء بازی ، گرافیک ها و دیگر فایل های مورد استفاده در بازی است . هنگام اجرای یونیتی ، اولین چیزی که می بینید ، صفحه خوش آمدگوئی است :



از این پنجره می توانید به راحتی پروژه ای را که اخیرا بر روی آن کار می کرده اید ، باز کنید . این ها در سمت چپ زیر لینک Projects لیست شده اند . در ضمن با کلیک دکمه New project که در قسمت راست پنجره وجود دارد ، می توانید یک پروژه جدید باز کنید .

اجازه دهید یک پروژه جدید ایجاد کنیم . مرحله اول کلیک دکمه New project است . این کار یک پنجره جدید به شکل زیر باز می کند :



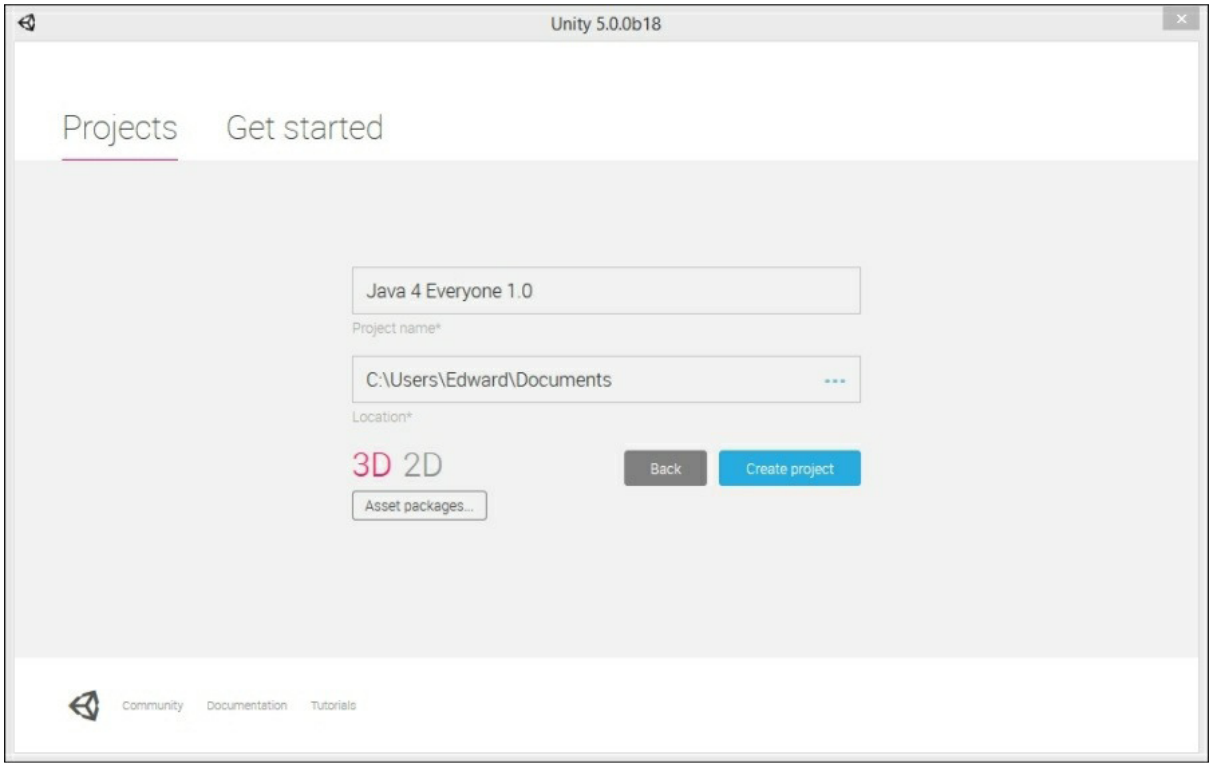
قبل از کلیک بر روی دکمه Create project ، ۴ کار باید انجام دهید . ابتدا ، باید به پروژه تان یک اسم بدهید . این کار مثل نام گذاری یک فایل است و من پیشنهاد می کنم اسمی که انتخاب می کنید ، شامل ورژن بازیتان هم باشد . مثلا اگر بر روی اولین ورژن یک بازی جدید به نام Java For Everyone کار می کنید ، می توانید پروژه را Java4Everyone 1.0 ، java_for_everyone_1.0, یا اسامی شبیه به این بنامید .

مرحله بعد این است که باید به یونیتی بگوئیم ، در کجا می خواهیم پروژه را ذخیره کنیم . می توانید مسیر پیش فرض را قبول کنید یا یک مسیر جدید انتخاب نمائید .

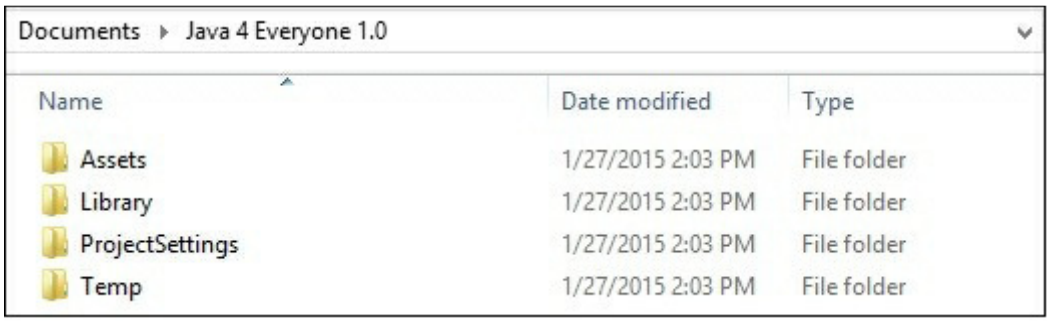
سپس باید مشخص کنیم که یک پروژه دوبعدی ایجاد می کنیم یا یک پروژه سه بعدی ؟ البته مطمئنا می توانید هم اجزای دوبعدی و هم اجزای سه بعدی را به بازیتان اضافه کنید . یونیتی در این خصوص ، انعطاف پذیری فوق العاده ای به ما می دهد .

آخرین کاری که باید انجام دهیم ، مشخص کردن بسته های منابع است که می خواهیم در بازی استفاده کنیم . البته بعد از شروع به کار بر روی پروژه نیز می توان این بسته ها را وارد بازی کرد . این شیوه ای است که من پیشنهاد می کنم چون فقط آنچه را که نیاز خواهید داشت ، بارگذاری خواهید کرد . در فصل بعد ، نحوه انجام این کار را نشان می دهیم .

بعد از انجام این کارها ، به شکل زیر دکمه Create project را کلیک کنید :



یونیتی پروژه شما را ایجاد خواهد کرد . اگر پنجره ناپدید شود و شما چیزی نبینید ، نگران نباشید چون یونیتی در پس زمینه درحال کارکردن است و محیط اصلی را به محض ایجاد پروژه ، باز خواهد کرد .
خواهید دید که یونیتی یک پوشه با نام پروژه شما ایجاد کرده است . در این پوشه ، چهار زیرپوشه به نام های Assets ، Library ، ProjectSettings و Temp وجود دارد . یونیتی پروژه ها را با استفاده از این پوشه ها ، سازماندهی می کند . اشیاء جدیدی که به بازی اضافه می شوند ، در این پوشه ها جای می گیرند :

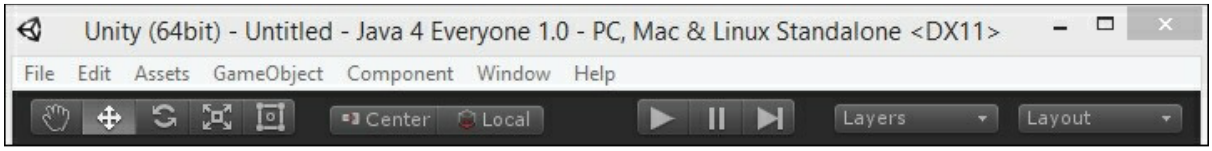


توجه

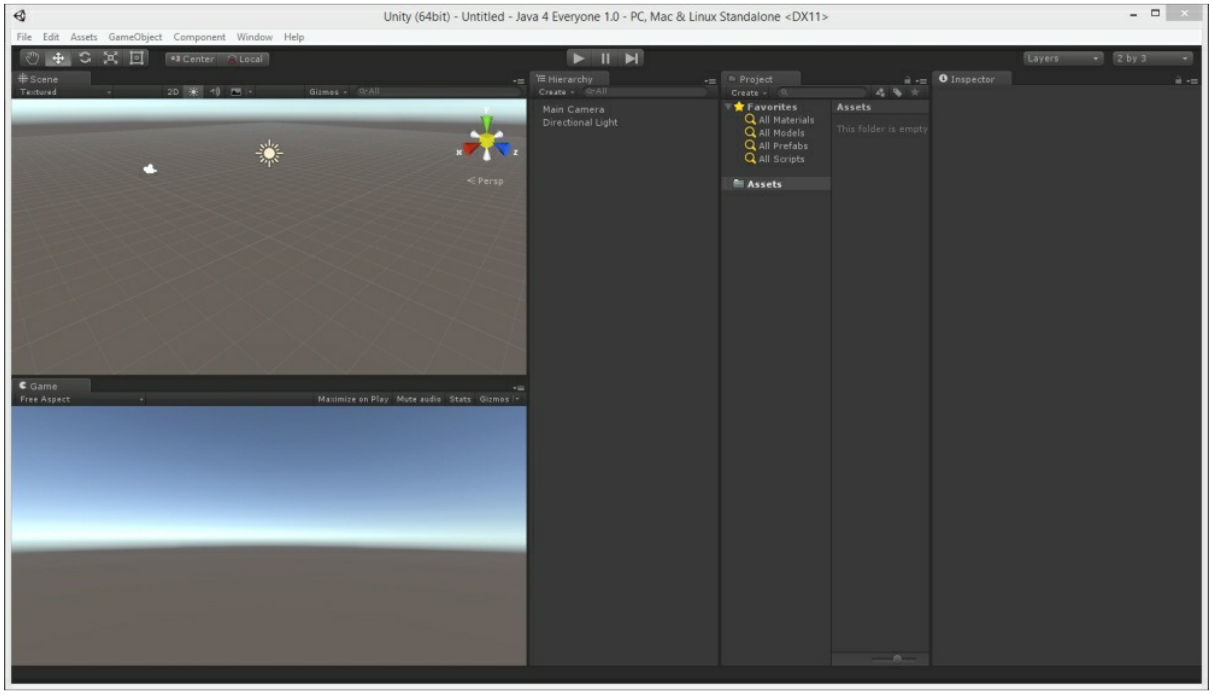
فایل های موجود در این پوشه ها را جابه جا نکنید و اسم پوشه ها را نیز تغییر ندهید . این کار باعث از بین رفتن ارتباط های موجود در پروژه می شود . سعی کنید تغییرات موردنیازتان را در داخل یونیتی انجام دهید .

طرح بندی ها

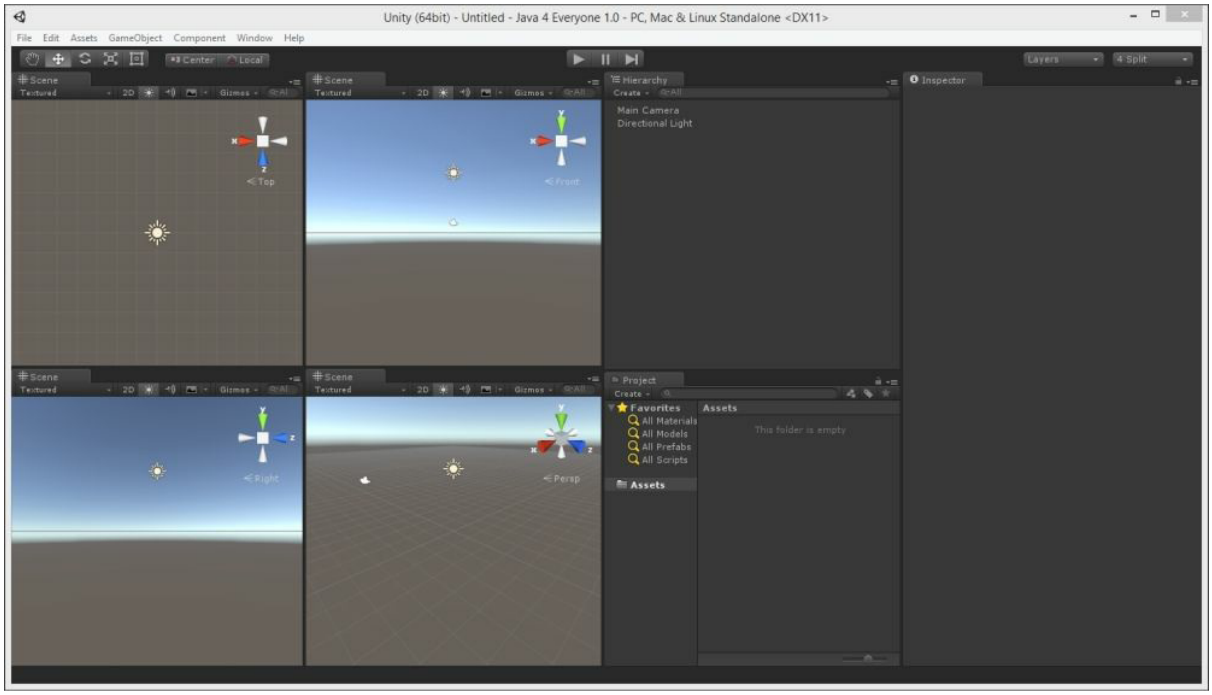
یکی از امکانات عالی درهنگام کار با یونیتی ، این است که شما می توانید شیوه چینش و طرح بندی محیط کاری موتور را تغییر دهید . می توان از یکی از چینش های ازپیش تعریف شده 2 by 3 ، 4 Split ، Tall یا Wide استفاده کرد یا اینکه خودتان یکی ایجاد کنید . طرح بندی و چینش ها (Layout) ؛ اشاره به نحوه قرارگرفتن نماهای مختلف یونیتی روی صفحه دارد . در بخش بعدی ، با نماها آشنا می شوید .
برای تغییر چینش ، روی دکمه Layout که در گوشه بالائی سمت راست محیط کاری یونیتی قرار دارد ، کلیک کنید :



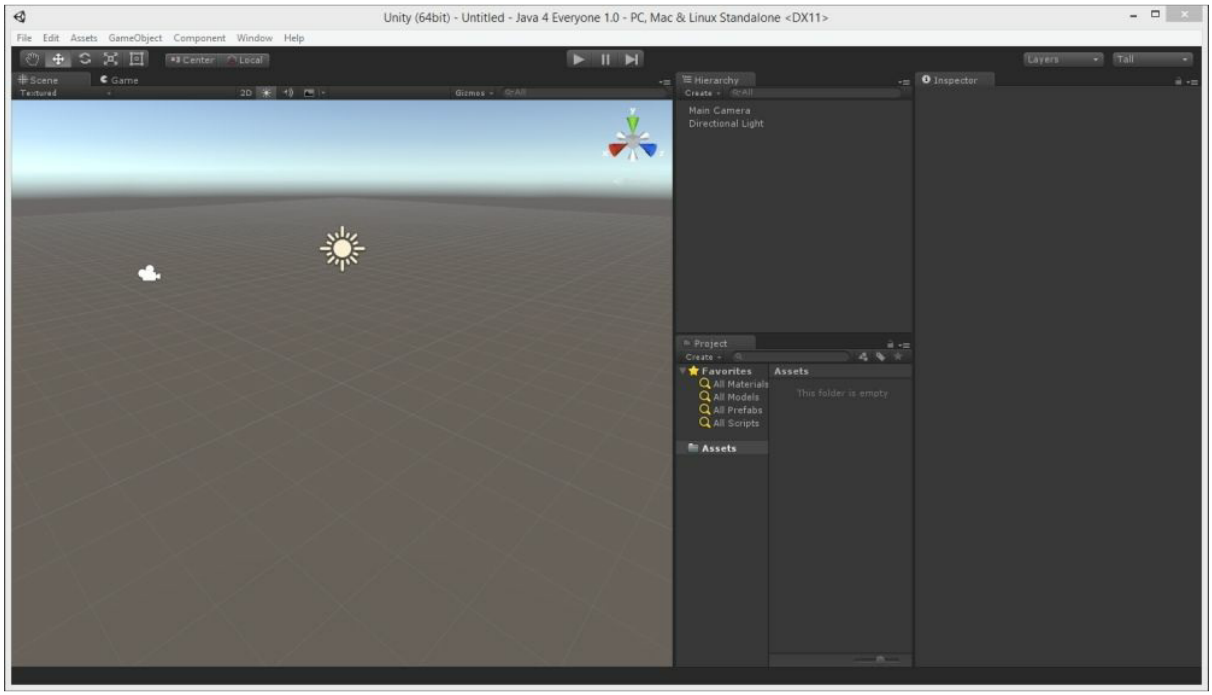
اجازه دهید هریک از چینش ها را ببینیم تا متوجه تفاوت های آنها شویم . اولین چینش ، 2 by 3 است . این چینش یک چیدمان زیبا از قرارگرفتن نماهای Scene و Game در سمت چپ ، نماهای Hierarchy و Project در وسط و نمای Inspector در سمت راست به شکل زیر فراهم کرده است :



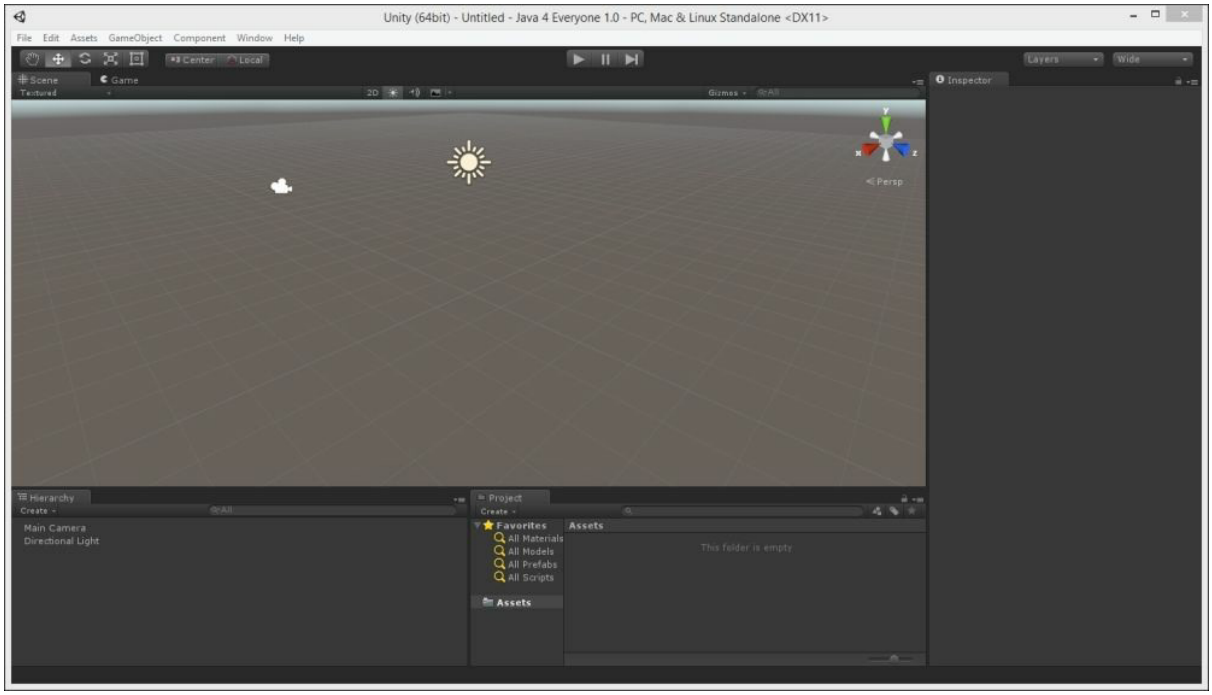
چینش 4 Split چهار نمای مختلف از یک صحنه را به شکل زیر فراهم کرده است . با استفاده از این چینش ، می توان یک مرور عالی بر روی نحوه پیاده سازی نورپردازی و سایه زنی بازی انجام داد . بعدا درباه نورپردازی و سایه زنی صحبت خواهیم کرد .



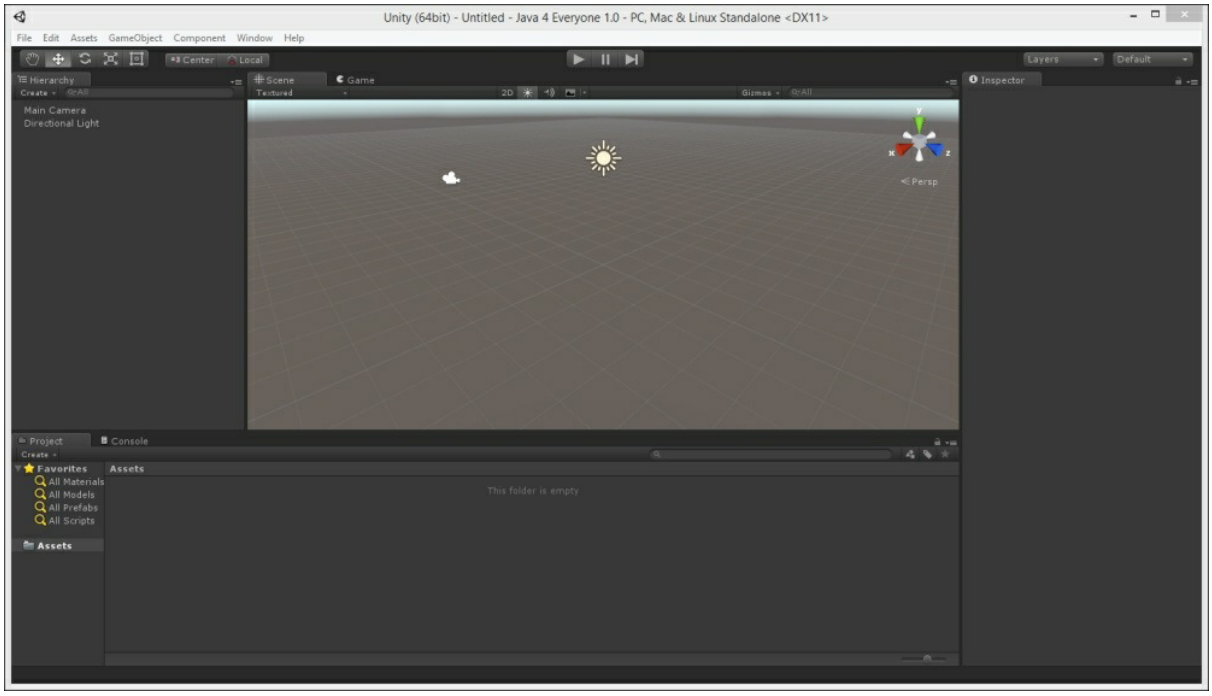
چینش Tall یک تصویر بلند نه چندان وسیع از نمای Scene به شکل زیر فراهم می کند :



چینش Wide ، یک دید وسیع و پهن از نمای Scene به شکل زیر فراهم می کند :



چینش Default نوعی از چینش Wide است . تفاوت این است که در چینش Default ، نمای Hierarchy در سمت چپ است :



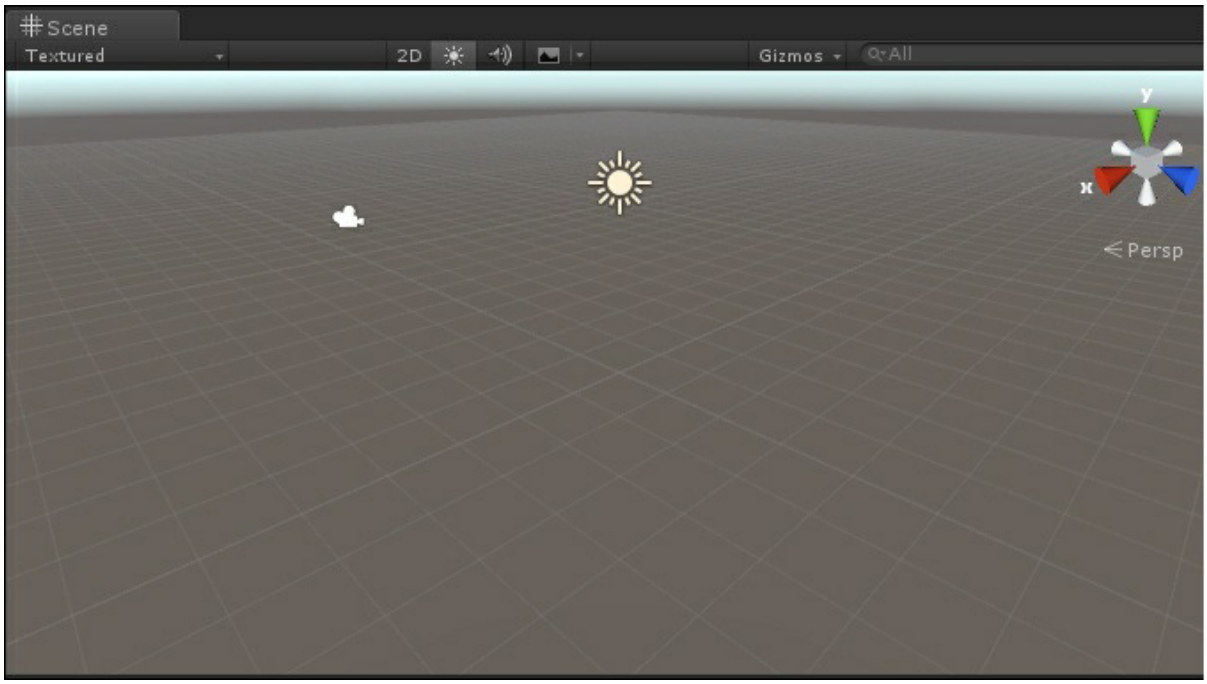
هروقت که خواستید می توانید این چینش ها را عوض کنید . اغلب کاربران یونیتی فقط در یک چینش کار نمی کنند . چینش های مختلف ، مزایای متفاوتی دارند و برای کارهای مختلفی ، مناسب هستند . حتی می توانید با درگ کردن لبه هر نما ، چینش را تغییر دهید . اگر خواستید این چینش جدید را ذخیره کنید ، تغییرات دلخواه را بر روی نماهای فعلی انجام داده و سپس ، دکمه Layout و Save Layout را انتخاب کنید . از شما یک نام نیز برای این چینش جدید پرسیده خواهد شد .

نماها

یونیتی امکان دیدن اجزای خاص یک پروژه را فراهم کرده است . این کار از طریق نماها (view) انجام می شود . ۵ نما در یونیتی وجود دارد .

نمای Scene

صحنه ها در یونیتی در حکم مراحل یک بازی هستند . برای هر مرحله از بازی ، صحنه های متفاوتی خواهید داشت . نمای Scene (صحنه) ، جائی است که اشیاء ویژوال مثل کاراکترها ، ساختمان ها ، زمین و ... را قرار می دهیم . همچنین با استفاده از ابزارهای تغییر شکل که در بخش بعدی معرفی می شوند ، می توان این اشیاء را در نمای Scene جابه جا کرد ، چرخاند و کوچک و بزرگ کرد . تصویر زیر را ببینید :

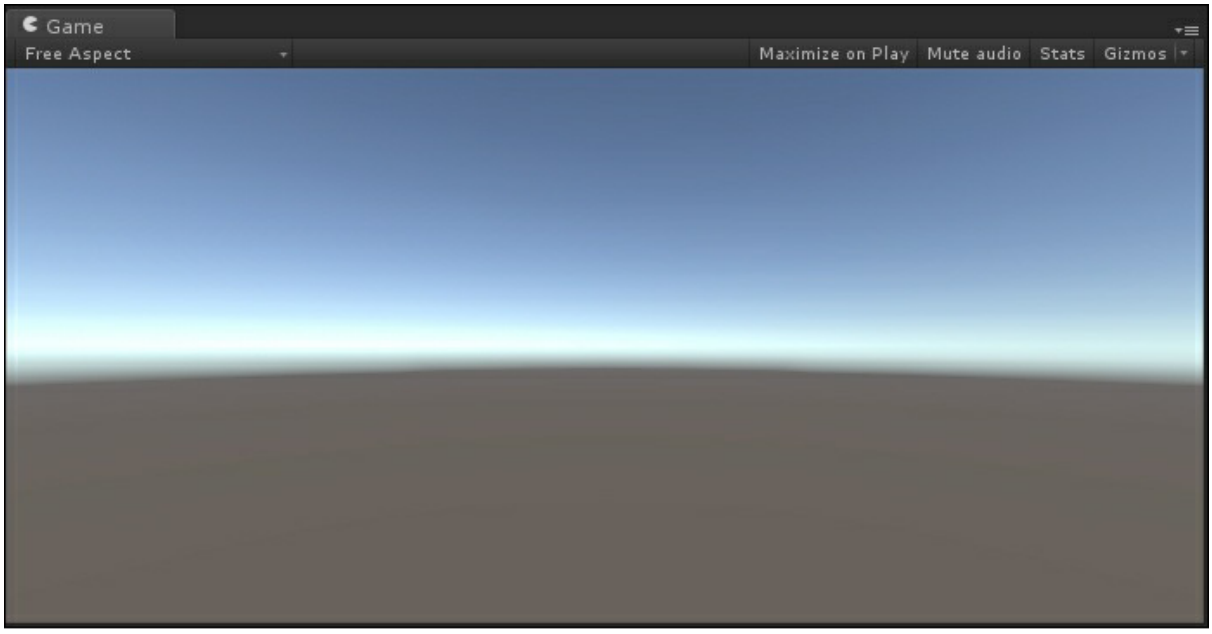


نمای Game

یکی از خصوصیات عالی یونیتی این است که می توان در داخل موتور بازی ، بازی را اجرا کرد . این یعنی لازم نیست بازی را ابتدا کامپایل کرده و در خارج از یونیتی آن را اجرا کرد . نمای Game جائی است که بازی در هنگام ساخته شدن ، اجرا می شود .

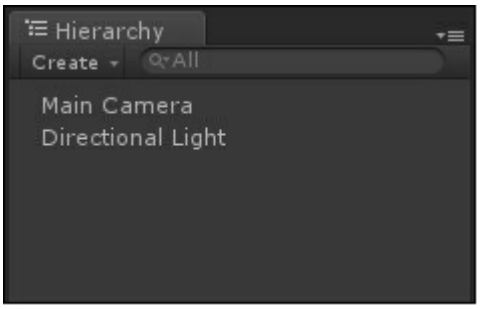
هر تغییر اعمال شده بر روی بازی درهنگامی که بازی در حال اجراست ، در بازی منعکس خواهد شد اما هنگامی که گیم پلی تمام می شود ، این تغییرات ذخیره نخواهد شد . این یک شیوه عالی برای تست بازی است و ریسک ازبین رفتن اسکریپت ها و دیگر اجزای بازی را منتفی می کند .

مثل اغلب بازی ها ، بازی های یونیتی نیز از دید بازیکن ، اجرا می شوند . معمولا ، بازیکن ، یک کاراکتر را در داخل بازی کنترل می کند . ما دنیای بازی را از طریق چشم های این کاراکتر نمی بینیم . ممکن است این طور به نظر آید اما در واقع ما دنیای بازی را به شکلی که توسط دوربین یا دوربین ها رندر شده اند ، می بینیم . بعدا در این باره بیشتر صحبت می کنیم . فعلا فکر کنید دوربین درست دربالا و پشت سر کاراکتر بازی ، پرواز می کند . این باعث می شود ؛ فکر کنیم از طریق چشم های این کاراکتر در حال دیدن اطراف هستیم :



نمای Hierarchy

این نما یک لیست از اشیاء موجود در صحنه نمایش می دهد . هنگامی که بر روی یک شئ موجود در این لیست ، دوبارکلیک می کنید ، دو اتفاق می افتد . ابتدا شئ انتخاب شده و در نمای Scene قابل دیدن می شود . یونیتی طوری نمای Scene را مرتب می کند که اشیاء انتخاب شده در جلو و وسط قرار می گیرند . دومین اتفاقی که می افتد این است که نمای Inspector تنظیمات و اجزای شئ انتخاب شده را نمایش خواهد داد . بعدا با این کار بیشتر آشنا می شوید . تصویر زیر را ببینید :

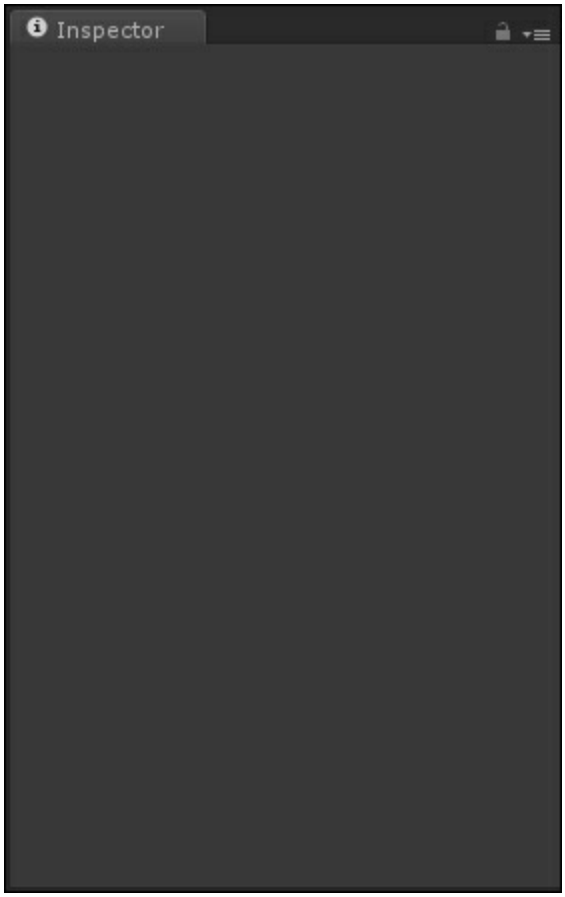


نمای Project

نمای Project یک ساختار فایلی از اجزای بازی شامل اشیاء ، گرافیک ها ، اسکریپت ها و ... برای ما فراهم می کند . ساختار فایلی اینجا همان چیزی است که روی هارد دیسک شما ذخیره شده است . پس اگر می خواهید تغییراتی را انجام دهید ، در نمای Project این کار را انجام دهید نه مستقیما از روی هارد دیسک . توجه کنید که این نما دو ستون دارد . ستون اول شامل ساختار فایلی ذکرشده و نیز Favorites است که به شما در یافتن سریع منابع کمک می کند . ستون دوم شامل محتویات پوشه هاست . مثلا اگر روی یک پوشه در ستون سمت چپ ، کلیک کنید ، محتویات این پوشه در ستون دوم نمایش داده می شود . می توانید اشیاء را از این قسمت ، مستقیما به داخل نمای Scene درگ کنید تا به بازی اضافه شوند . این کار را بعدا انجام می دهیم . تصویر این نما را ببینید :

نمای Inspector

هنگامی که در نمای دیگری ، اشیاء انتخاب شده باشند ؛ جزئیات آنها در نمای Inspector نمایش داده می شود . در بسیاری از موارد ، می توانید مستقیما در نمای Inspector ؛ تغییراتی را بر روی یک شئ اعمال کنید . در این کتاب این کار را زیاد انجام می دهیم .



یادآوری می کنم ، چینش ها از نماها ساخته شده اند و شما بر روی نحوه چیدمان محیط کاری ، کنترل کامل دارید . برای بیشتر این کتاب ، من از چینش 2 x 3 استفاده می کنم چون خیلی سودمند است .

ابزارهای تغییر شکل

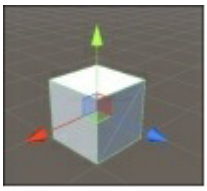
ابزارهای تغییر شکل ، در یونیتی به ما اجازه می دهند با نمای Scene تعامل داشته باشیم ، زمین را ویرایش کنیم ، اشیاء را جابه جا کنیم و تغییرات را انجام دهیم . ابزارهای تغییر شکل شامل ۵ دکمه هستند که در تصویر زیر می بینید . اینها در گوشه بالائی سمت چپ محیط کاری یونیتی ، واقع شده اند :

اولین دکمه ، ابزار پیمایش (View) است . هنگامی که این ابزار انتخاب شده باشد ، در نمای Scene مکان نمای ما به شکل یک دست می شود . با کمک این ابزار ، می توانیم با چرخاندن ماوس ، به داخل صحنه زوم کنیم . اگر دکمه سمت چپ ماوس را کلیک کنید ، قادر خواهید بود

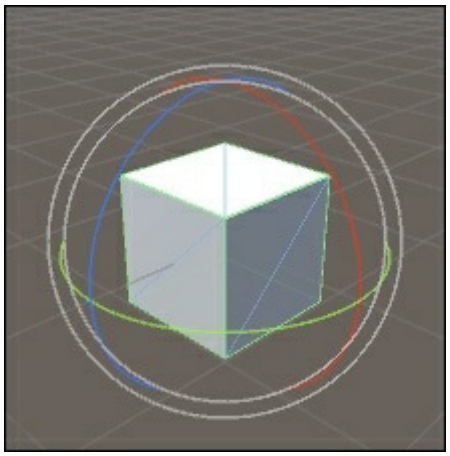
در اطراف صحنه حرکت کنید . با کلیک دکمه راست ماوس ، قادرخواهید بود بر طبق موقعیت فعلی مکان نما ، به اطراف نگاه کنید .

اگر کلید Alt را نگه دارید و دکمه سمت چپ ماوس را کلیک کنید ، می توانید اطراف منطقه فعلی ، بچرخید . نگه داشتن کلید Alt و دکمه سمت راست ماوس ، به سمت صحنه زوم می کند .

دکمه بعدی ، دکمه انتقال (Translate) است و به شکل یک پیکان چهارگوش است . هنگامی که یک شئ انتخاب شده باشد و سپس روی این ابزار کلیک کنیم ، شئ دارای یک گیره روی هر محور می شود . کلیک و درگ هر یک از این گیره ها ، به شکل زیر ، شئ را جابه جا خواهد کرد :

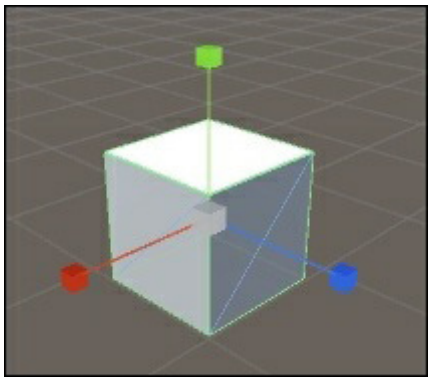


سومین ابزار تغییر شکل ، ابزار چرخش (Rotate) است که به شکل دو پیکان منحنی است . این ابزار به ما اجازه می دهد یک شئ را حول هر محور (x ، y یا Z) بچرخانیم . این ابزار با سه حلقه رنگی ؛ هرکدام برای هر محور ؛ نمایش داده می شود . کلیک بر روی حلقه و درگ آن ، شئ را حول آن محور می چرخاند . ببینید :

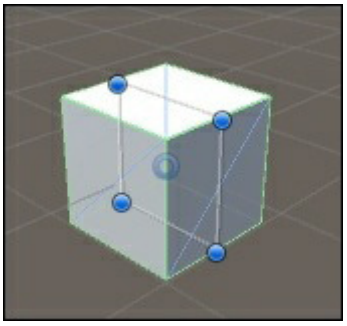


چهارمین ابزار ، ابزار تغییر اندازه (Scale) است . مثل ابزارهای دیگر ، یک گیره برای هر محور وجود دارد . کلیک و درگ هریک از این گیره ها ؛ اندازه شئ را در طول محور انتخاب شده ، افزایش یا کاهش می دهد . مثلا می توانید یک مکعب را پهن تر ، باریک تر ، بلند تر یا کوتاه تر کنید . اگر بخواهید نسبت دید

(aspect ratio) را حفظ کنید ، می توانید به جای کلیک بر روی مربع های قرمز ، آبی یا سبز ؛ روی مربع موجود در مرکز شئ کلیک کنید . حالا هنگام درگ ، شئ شما با نسبت دید مناسب ، کوچک و بزرگ می شود . تصویر زیر را ببینید :



آخرین ابزار ، ابزار Rect است و با یک مستطیل و نقطه ای در وسط نمایش داده می شود . از این ابزار می توان برای جابه جائی ، تغییراندازه و چرخاندن یک شئ در نمای Scene استفاده کرد . بنابراین ، این یک ابزار همه کاره است . تصویر زیر را ببینید :



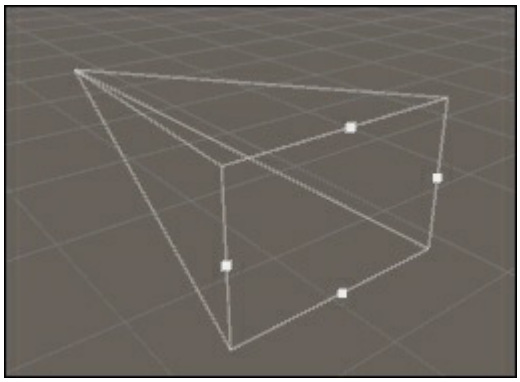
دوربین ها

دوربین ها ، صحنه ها را رندر می کنند طوری که کاربر بتواند آنها را ببیند . هر صحنه ای باید حداقل یک دوربین داشته باشد . درواقع ، هنگام ایجاد یک صحنه جدید ؛ یونیتی یک دوربین به نام Main Camera ایجاد می کند . یک صحنه می تواند چندین دوربین داشته باشد . در نمای Scene ؛ دوربین ها با یک نیمرخ سفید از یک دوربین به شکل زیر نمایش داده می شوند :

دوربین ها ، اشیاء بازی هستند و می توانند با استفاده از ابزار تغییر شکل و نیز در نمای Inspector ، ویرایش شوند . براساس تصویر دوربین ها ، می توانیم دوربین های یونیتی را دسته بندی کنیم . یک دوربین با دورنمای سه بعدی (Perspective)؛ صحنه را براساس زاویه دوربین به شکلی که در صحنه موجود است ، رندر می کند .

تصویر دیگر آرتوگرافیک است . یک دوربین با دورنمای آرتوگرافیک (Orthographic) ؛ صحنه را به طور یکنواخت بدون هیچ دورنمای سه بعدی رندر می کند . از این نوع دوربین معمولا برای بازی های ازبالا به پائین استفاده می شود و دوربین پیش فرض مورد استفاده در بازی دوبعدی و سیستم رابط کاربری یونیتی [User Interface (UI)] است .

هنگامی که در نمای Hierarchy یک دوربین انتخاب می شود ، مخروط آن در نمای Scene قابل دیدن می شود . مخروط ، یک شکل جغرافیائی است که شبیه به هرمی است که نوکش قطع شده باشد . صفحه بالائی ، موازی با پایه اش است . به عبارت دیگر ، صفحه های نزدیک و دور :



ما در آینده از دوربین ها در بازی هایمان استفاده خواهیم کرد .

توجه

اجزای دیگری از محیط کاری یونیتی وجود دارند که ما در این فصل آنها را معرفی نکردیم . با نیاز به آنها در هنگام ساختن بازی ، این ابزارها معرفی خواهند شد .

خلاصه

در این فصل ، ما به سرعت موتورهای بازی را معرفی و با هم مقایسه کردیم . همچنین نگاهی دقیق تر به موتور بازی یونیتی و بعضی از خصوصیات کلیدی آن داشتیم . ما یونیتی را دانلود و نصب کردیم و آماده ساختن بازی ها شدیم . همچنین پروژه های یونیتی را مورد بحث قرار داده و برای آشنا شدن با آنها ، یک پروژه ایجاد کردیم . در انتهای فصل ، با محیط کاری و چهره یونیتی آشنا شده و به معرفی چینش ها ، نماها ، ابزارهای تغییر شکل و دوربین ها پرداختیم .

در فصل بعد ، شروع به طراحی بازی می کنیم که در طول این کتاب ساخته می شود و شما یاد می گیرد که چطور محیط بازی را ایجاد کنید .

فصل ۲ : ایجاد محیط بازی

در فصل قبل ، موتورهای بازی را بررسی کردیم ، یونیتی را نصب کرده و با محیط کاری آن آشنا شدیم . حالا تقریبا آماده هستیم تا ساختن بازیمان را شروع کنیم . قبل از این کار ، به دو موضوع باید توجه کنیم : طراحی بازی و ایجاد محیط بازی . در این فصل ، این دو کار را انجام خواهیم داد .

ما فصل را با طراحی بازیمان آغاز می کنیم که به عنوان طرح اولیه ای خواهد بود که آن را در طول کتاب دنبال خواهیم کرد . سپس محیط بازیمان را ایجاد می کنیم که شامل زمین ، آب و آسمان خواهد بود . زمین بازی ما از درخت ها ، کوه ها و رودخانه تشکیل شده است .

در طول این فصل :

با هدف طراحی بازی آشنا می شوید .

قادر خواهید بود در یونیتی ، زمین ایجاد کنید .

استفاده از بافت ها را یاد می گیرید .

با ایجاد کوه ها آشنا می شوید .

با کاشتن درخت ها آشنا می شوید .

با ایجاد آب آشنا می شوید .

با ایجاد آسمان در یونیتی ، آشنا می شوید .

طراحی بازی

طراحی بازی شبیه به طرح اولیه یک ساختمان است . شما بدون داشتن یک نقشه اولیه ، به ساختن خانه اقدام نخواهید کرد ؛ در خصوص بازی ها هم ، ساختن بازی بدون اینکه آن را ابتدا طراحی کنید ، کار اشتباهی است . طراحی بازی ، مسئول مستندسازی چهره بازی ؛ ماموریت های بازیکن ؛ گیم پلی بازی ؛ رفتارهای بازیکن ؛ هوش مصنوعی مورد نیاز و جریان بازی است .

طراحی بازی ، خیلی پیچیده تر از بررسی این شش وظیفه ذکر شده است . ما زمان نداریم تا به فرایند کامل طراحی بازی بپردازیم بنابراین ، در این فصل فقط همین اجزای اولیه از بازی را طراحی خواهیم کرد .

بازی ما Little Farmer Colt نام دارد . این بازی یک کشاورز جوان به نام Colt را نشان می دهد که باید یک مزرعه را مدیریت کند . اجازه دهید به اجزای این بازی بپردازیم .

چهره بازی

هر بازی یک محیط و چهره مخصوص به خود را دارد . برای بازی هائی که در فضا انجام می شوند ، چهره بازی ممکن است خیلی پیشرفته و مکانیکی باشد . اگر بازی مخصوص بچه ها باشد ؛ ممکن است بازی به سبک کارتون ها باشد . نکته مهم در زمینه چهره بازی این است که باید سازگار با بازی باشد .

بازی ما در یک مزعه کوچک اتفاق می افتد بنابراین چهره بازی شبیه به مزارع روستائی خواهد بود و شامل حیوانات مزرعه ، ساختمان های مناسب ، محصول کشاورزی ، آب ، سبزه و خاک است . یک نکته در حفظ سازگاری چهره بازی این است که یک گرافیست ، همه گرافیک های بازی را آماده کند . در پروژه های بزرگ ؛ این کار همیشه عملی نیست ؛ اما برای بازی کوچک ما شدنی است . تمام گرافیک های مورد استفاده در بازی ما ، توسط یک گرافیست ایجاد شده است .

ماموریت های بازیکن

کاربرانی که بازی ما را انجام می دهند باید ماموریت هایشان را بدانند . این ماموریت ها ، به آنها برای اجرای بازی ؛ هدف می دهد . ماموریت ها ، به بازی سرگرمی نیز اضافه می کند . بازی ما شامل سه ماموریت است :

ماموریت ۱ : یادگرفتن نحوه کاشت زمین . بازیکن ، این ماموریت را توسط صحبت با یک کشاورز سالخورده ، انجام خواهد داد .

ماموریت ۲ : پرورش جوجه ها . این کار توسط جمع آوری دانه و آب و تغذیه جوجه ها انجام خواهد شد .

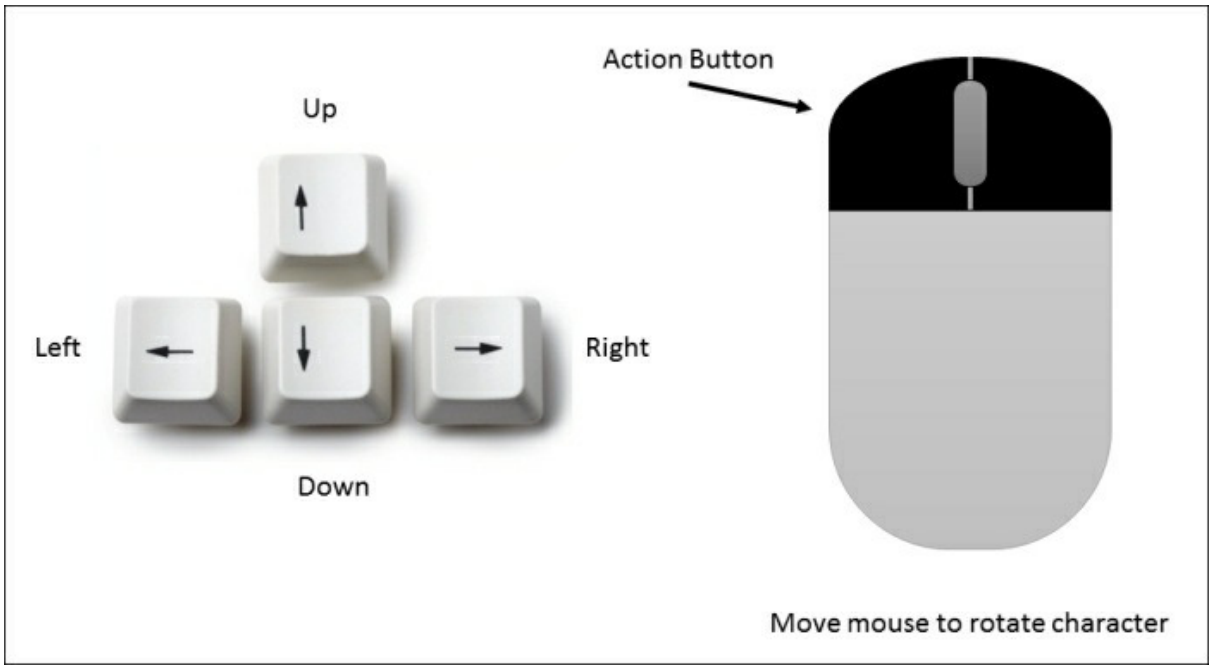
ماموریت ۳ : پرورش گرازها . این کار نیز مثل پرورش جوجه ها توسط فراهم کردن غذا و آب انجام می شود .

برای این ماموریت ها ، ما مزارع کاشته شده با غلات و یک نهر آماده می کنیم . بازیکن دانه ها را از این زمین ها برداشت کرده و آب را از نهر به دست می آورد . آب و دانه ها برای تغذیه حیوانات مزرعه استفاده خواهند شد .

گیم پلی

گیم پلی (Gameplay) به نحوه تعامل بازیکنان با بازی اشاره دارد . برای بازی ما ، بازیکنان از صفحه کلید و ماوس استفاده می کنند . از ماوس برای چرخاندن کاراکتر اصلی بازی و از کلیدهای جهتی صفحه کلید نیز برای حرکت دادن و هدایت او در محیط بازی استفاده می شود .

تصویر زیر ، یک نمای گرافیکی از این کنترل هاست . اینها استاندارد هستند و یونیتی ، استفاده از این کنترل ها را خیلی آسان کرده است . در آینده به این موضوع می پردازیم :



رفتارهای بازیکن

یکی دیگر از اجزای طراحی بازی ، تعریف کارهائی است که بازیکنان قادر به انجام آن خواهند بود . این کارها می تواند شامل راندن یک کشتی ، شلیک یک موشک ، شناکردن ، بالاپریدن و ... باشد . در بازی ما ، بازیکن قادر به انجام کارهای زیر خواهد بود :

راه رفتن در محیط بازی

برداشتن دانه

برداشتن آب

غذادادن به جوجه ها

غذادادن به گرازها

صحبت با کشاورز مسن

این رفتار ها ، به عنوان فهرستی از چیزهائی است که باید در بازی پیاده سازی کنیم . در ادامه کتاب این کارها را انجام می دهیم .

هوش مصنوعی

هوش مصنوعی [Artificial Intelligence (AI)] هنگامی اجرا می شود که ما اشیاء بازی را براساس اشیاء دنیای واقعی برنامه نویسی می کنیم . بازیکن ، Colt یعنی همان کاراکتر اصلی بازی را کنترل می کند اما برای کاراکترهای دیگر باید هوش مصنوعی را پیاده سازی کنیم . این کاراکترها عبارتند از : کشاورز مسن ، جوجه ها و گرازها

ما از کشاورز مسن برای نمایش یک سیستم گفتگو و محاوره استفاده می کنیم . بنابراین او هیچ هوش مصنوعی ندارد بلکه براساس انتخاب های Colt ، یک سری پاسخ های از پیش تعریف شده خواهد داشت .

مرغ ها و گرازها ، هوش مصنوعی شبیه به هم خواهند داشت . آنها به شکل جوجه و بچه گراز متولد می شوند . با آب و دانه کافی ، بزرگ می شوند . هنگامی که کامل رشد کردند ، به صورت خودکار جوجه و بچه گراز تولید مثل می کنند . اگر آب و غذای کافی به آنها نرسد ، می میرند .

بنابراین ، لیست هوش مصنوعی برای این بازی به شکل زیر خواهد بود :

Colt

هوش مصنوعی : ندارد

توضیح : توسط بازیکن کنترل می شود .

کشاورز مسن

هوش مصنوعی : ندارد

توضیح : توسط سیستم گفتگو کنترل می شود .

جوجه ها

هوش مصنوعی : خوردن / آشامیدن – رشدکردن و تبدیل به مرغ شدن ـ مردن

توضیح : چندین نمونه از این شئ در بازی موجود خواهد بود .

مرغ ها

هوش مصنوعی : خوردن / آشامیدن – تولید مثل جوجه ـ مردن

توضیح : چندین نمونه از این شئ در بازی موجود خواهد بود .

بچه گراز

هوش مصنوعی : خوردن / آشامیدن – رشدکردن و تبدیل به گراز بالغ شدن ـ مردن

توضیح : چندین نمونه از این شئ در بازی موجود خواهد بود .

گراز بالغ

هوش مصنوعی : خوردن / آشامیدن – تولید مثل بچه گرازها ـ مردن

توضیح : چندین نمونه از این شئ در بازی موجود خواهد بود .

وضعیت پایان بازی

بخش پایانی طراحی بازی ، تعیین نحوه پایان بازی است . دو نوع وضعیت پایانی وجود دارد : برد و باخت . وضعیت برد ، هدف نهائی است . ما باید مشخص کنیم که بازیکنان چطور بازی را می برند . در بازی ما ، شرط پیروزی داشتن ۵ مرغ و گراز بالغ است . در بازی های بزرگ تر ، این وضعیت پایانی ممکن است حاکی از پایان مرحله اول باشد . در انتهای کتاب ، این به شما فرصت توسعه بازی را خواهد داد .

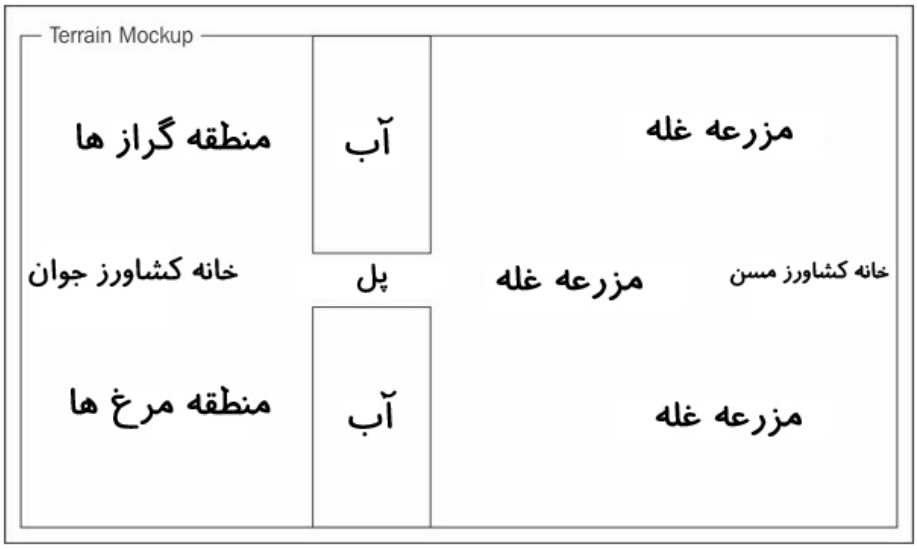
وضعیت پایانی دیگر ، باخت است . قبلا گفتیم که حیوانات می توانند بمیرند . بنابراین شرط باخت ، این است که حداقل یکی از هرنوع از حیوانات بمیرند . فرقی نمی کند که جوجه ها ؛ بچه گرازها ، مرغ ها یا گرازها باشند .

خوب ، بازی ما کاملا طراحی شده است . حالا می توانیم شروع به ایجاد محیط بازی کنیم . در بخش بعدی ؛ این کار را با ایجاد زمین ؛ آغاز می کنیم .

زمین

محیط بازی ما به شکل یک مزرعه روستائی همراه با آب و زمین است . ما زمین را همراه با آبی که در وسط زمین جاری است و یک پل که به بازیکن اجازه می دهد بین دو سمت زمین حرکت کند ؛ ایجاد می کنیم . در سمت چپ زمین ، خانه Colt و محیطی برای جوجه ها و گرازها قرا می دهیم . کشاورز مسن در سمت راست زمین یک خانه خواهد داشت که با مزارع غلات ، احاطه شده است .

برای جذاب تر شدن زمین ، چند تپه و درخت نیز اضافه می کنیم . شکل زیر ، مدلی از آنچه که در این فصل ایجاد خواهیم کرد ؛ نشان می دهد :



ما یک پروژه ایجاد می کنیم و آن را در باقیمانده کتاب ، تکمیل خواهیم کرد . برای ایجاد این پروژه ، مراحل زیر را انجام دهید :

۱. یونیتی را اجرا کنید .

۲. در پنجره آغازی یونیتی ، دکمه New Project را کلیک کنید .

۳. برای اسم پروژه ، LittleFarmerColt را وارد کنید .

۴. مسیری برای ذخیره پروژه انتخاب کنید .

۵. دقت کنید که 3D انتخاب شده باشد .

۶. دکمه Create Project را کلیک کنید .

بعد از انجام این مراحل ، یونیتی یک پروژه جدید باز می کند . در بخش بعد ، ایجاد زمین را شروع می کنیم .

ایجاد زمین

یونیتی با زمین به عنوان یک شئ بازی رفتار می کند . در یونیتی ، به این به شکل GameObject اشاره می شود . GameObject یکی از خصوصیات اصلی یونیتی است . هر چیزی یک شئ بازی است مثلا

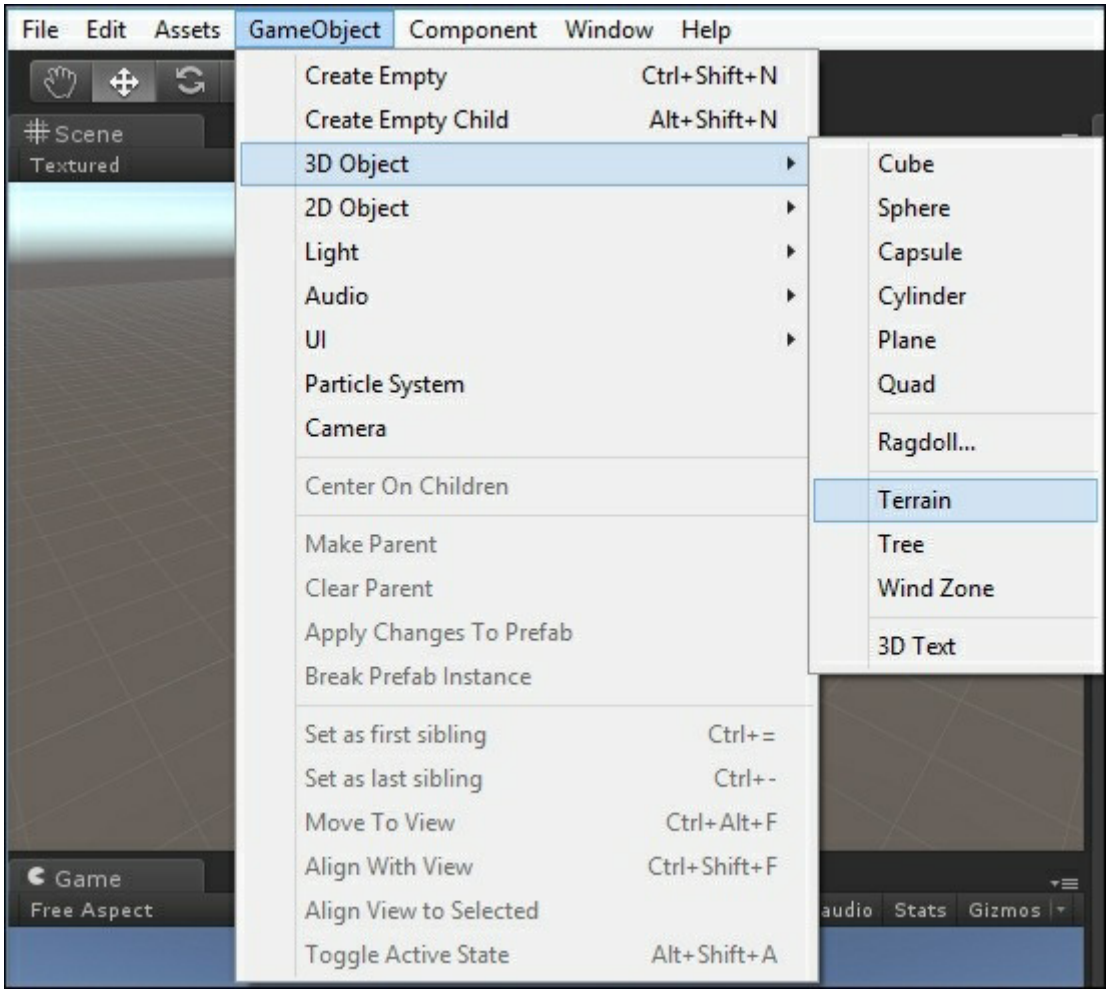
کاراکترها ، درخت ها ، نورها ، دوربین ها و ... این اشیاء به عنوان محفظه ای برای component ها ، عمل می کنند .

توجه

در یونیتی ، پروژه ها شامل انواع اشیاء بازی هستند . در نمای Project می توانید همه اشیاء بازی را ببینید . از بعضی از این اشیاء بازی ، در صحنه هایتان استفاده می کنید . بنابراین اشیاء بازی موجود در صحنه ، در نمای Hierarchy قابل دیدن خواهند بود . هنگامی که اشیاء بازی را از نمای Project به داخل نمای Hierarchy درگ می کنیم ؛ یک نمونه یا یک کپی از آن شئ بازی را ایجاد خواهیم کرد .

بعدا خواهید دید که چگونه می توان component ها را به اشیاء بازی اضافه کرد . فعلا می خواهیم زمین بازیمان را ایجاد کنیم . این کار را با اجرای مراحل زیر انجام می دهیم :

۱. از منو GameObject و بخش 3D Object ، گزینه Terrain را انتخاب کنید . این کار یک شئ زمین ایجاد می کند . خواهید دید که اشیاء در نمای های Hierarchy و Project لیست می شوند :



۲. سپس ، اندازه زمین را کاهش می دهیم تا مدیریتش آسان تر شود . برای این کار ، در نمای Hierarchy ، Terrain را انتخاب کرده و سپس در نمای Inspector ، دکمه Terrain Settings را انتخاب می کنیم . Terrain Width و Terrain Length را 250 قرار دهید . این باعث می شود اندازه زمین 250x250 شود .

توجه

هنگامی که اولین بار در یونیتی ، یک زمین ایجاد می کنید ؛ مبداء این زمین در یک گوشه است . می توانید مبداء صحنه را به (0,0,0) تغییر دهید تا در وسط زمین قرار گیرد . این کار ، مدیریت زمین را آسان تر می کند .

۳. در نمای Inspector ، روی دکمه Paint Height Tool کلیک کنید و سپس روی دکمه Flatten کلیک کنید . توجه کنید که با انجام این کار ممکن است اشیاء ؛ قاب موجود در نمای Scene را ترک کنند . می توانید با دوبارکلیک بر روی اشیاء در نمای Hierarchy ، آنها را به قاب برگردانید .

حالا زمین ایجاد شده است و آماده اضافه کردن اجزای دیگر هستیم .

اضافه کردن کوه ها

ما به زمین ، کوه و تپه اضافه می کنیم تا طبیعی تر شود . چون بازی ما در یک مزرعه انجام می شود ، کوه ها را فقط در اطراف لبه های زمین اضافه می کنیم . این به ما کمک می کند کاراکتر بازی از لبه زمین به پائین سقوط نکند .

یونیتی ، ایجاد کوه ها را تا حد درگ مکان نمای ماوس ، آسان کرده است . مراحل ایجاد کوهها را ببینید :

۱. در نمای Inspector ، روی ابزار Raise/Lower Terrain کلیک کنید .

۲. در نمای Inspector و زیر آیکون های این ابزار ، انواع قلم مو (brush) را خواهید دید . با یک کلیک ، یکی از آنها را انتخاب کنید . حتی می توانید اندازه آن را نیز تغییر دهید .

۳. ماوس را به نمای Scene ببرید . هنگامی که دکمه چپ ماوس را فشار داده و مکان نما را روی زمین ، درگ می کنید ؛ می بینید که کوه ها روی زمین ایجاد می شوند .

۴. در اطراف ۴ لبه زمین ، این کوه ها را ایجاد کنید .

هنوز کوه ها و زمین خیلی شبیه به هم نیستند .

اضافه کردن رودخانه

کار بعدی اضافه کردن رودخانه است . ما این کار را با حفر یک چاله یا گود کردن زمین در جائی که می خواهیم رودخانه وجود داشته باشد و سپس اضافه کردن آب انجام می دهیم . برای حفر چاله مراحل زیر را انجام دهید :

۱. در نمای Inspector ، روی ابزار Raise/Lower Terrain کلیک کنید .

۲. یک قلم موی بزرگ انتخاب کنید .

۳. مکان نمای ماوس را به نمای Scene ببرید .

۴. کلید Shift را بگیرید و دکمه چپ ماوس را فشار دهید . حالا با درگ ماوس روی زمین ، قادر به گود کردن زمین خواهید بود .

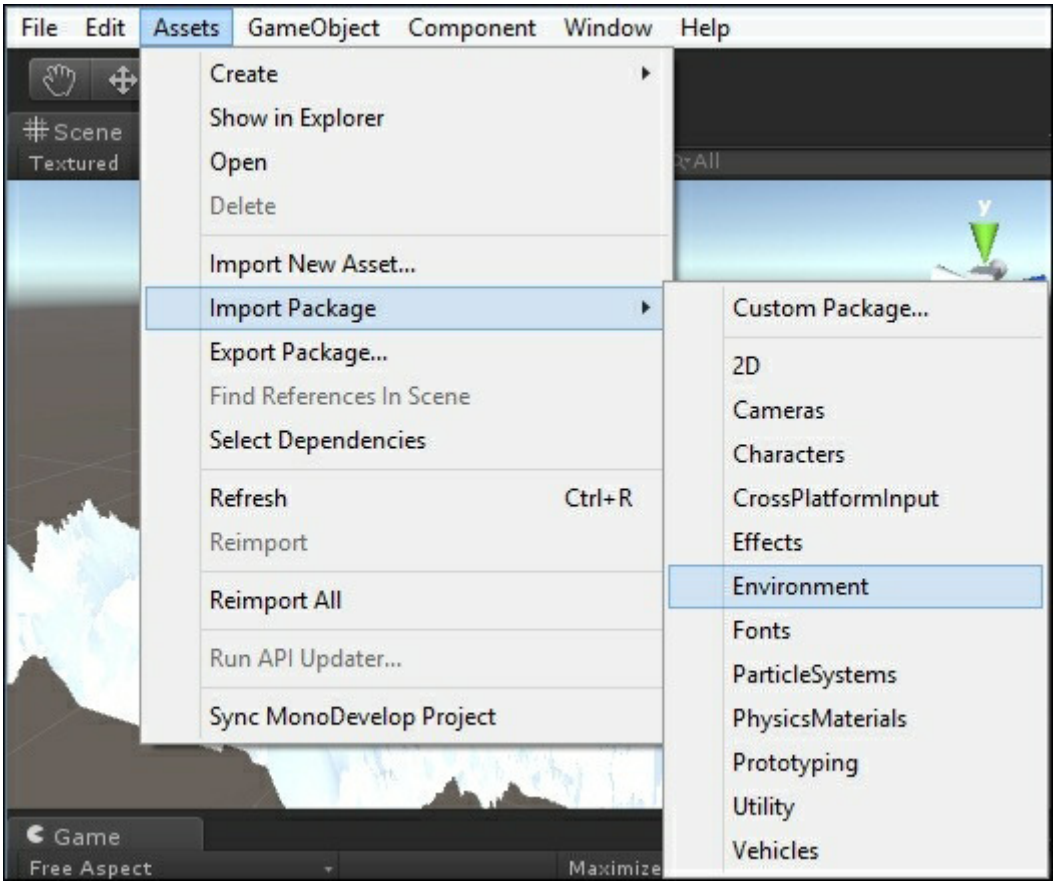
۵. زمین را در قسمت وسط از یک سمت به سمت دیگر گود کنید . این کار ، منطقه ای را برای قرارگرفتن آب آماده می کند طوری که زمین را دو نیمه خواهد کرد .

در ادامه آب را نیز اضافه خواهیم کرد .

بافت ها

بافت (Texture) در یونیتی ؛ فایل های تصویری هستند که برای اضافه کردن یک سطح بافت بندی شده به یک شئ ، استفاده می شوند . با انجام مراحل زیر ، چند بافت زمین اضافه می کنیم :

۱. از منو Assets و قسمت Import Package ، گزینه Environment را انتخاب کنید :

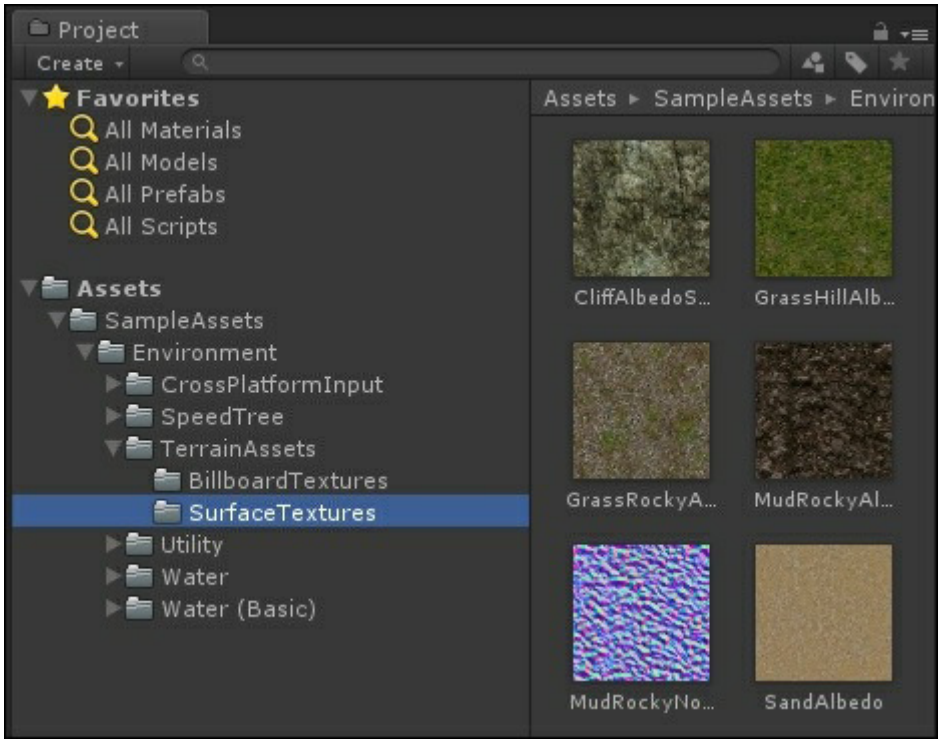


این کار باعث باز شدن پنجره Import Package می شود .

۲. دکمه Import را کلیک کنید تا بسته به پروژه اضافه شود .

بسته انتخاب شده ، بسته ای بزرگ است بنابراین ممکن است چند دقیقه طول بکشد تا به پروژه اضافه شود . بعدا درباره بسته ها (package) صحبت خواهیم کرد . فعلا بدانید که بسته ها ، راهی سریع برای اضافه کردن بسیاری از اشیاء به پروژه ها هستند .

بسته Environment چندین بافت دارد . اگر در نمای Project آن را باز کنید ، می بینید یک پوشه به نام Standard Assets یا SampleAssets وجود دارد . داخل پوشه Environment ، پوشه Terrain را می بینید :

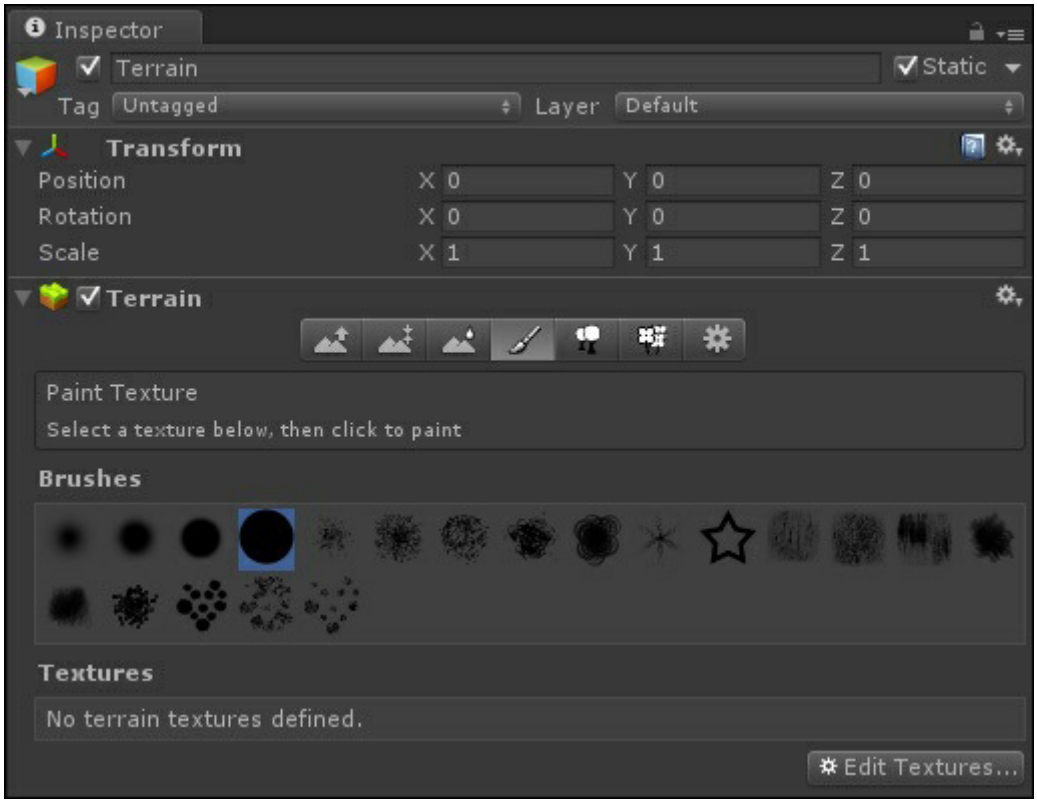


این پوشه محتوی بافت هائی است که ما از آنها برای زیباترکردن زمین استفاده می کنیم . با مراحل زیر این کار را انجام دهید :

۱. در نمای Hierarchy ، Terrain را انتخاب کنید .

۲. در نمای Inspector ، دکمه Paint Texture را انتخاب کنید .

۳. در نمای Inspector ، زیر قسمت قلم موها ، دکمه Edit Textures را کلیک کنید :



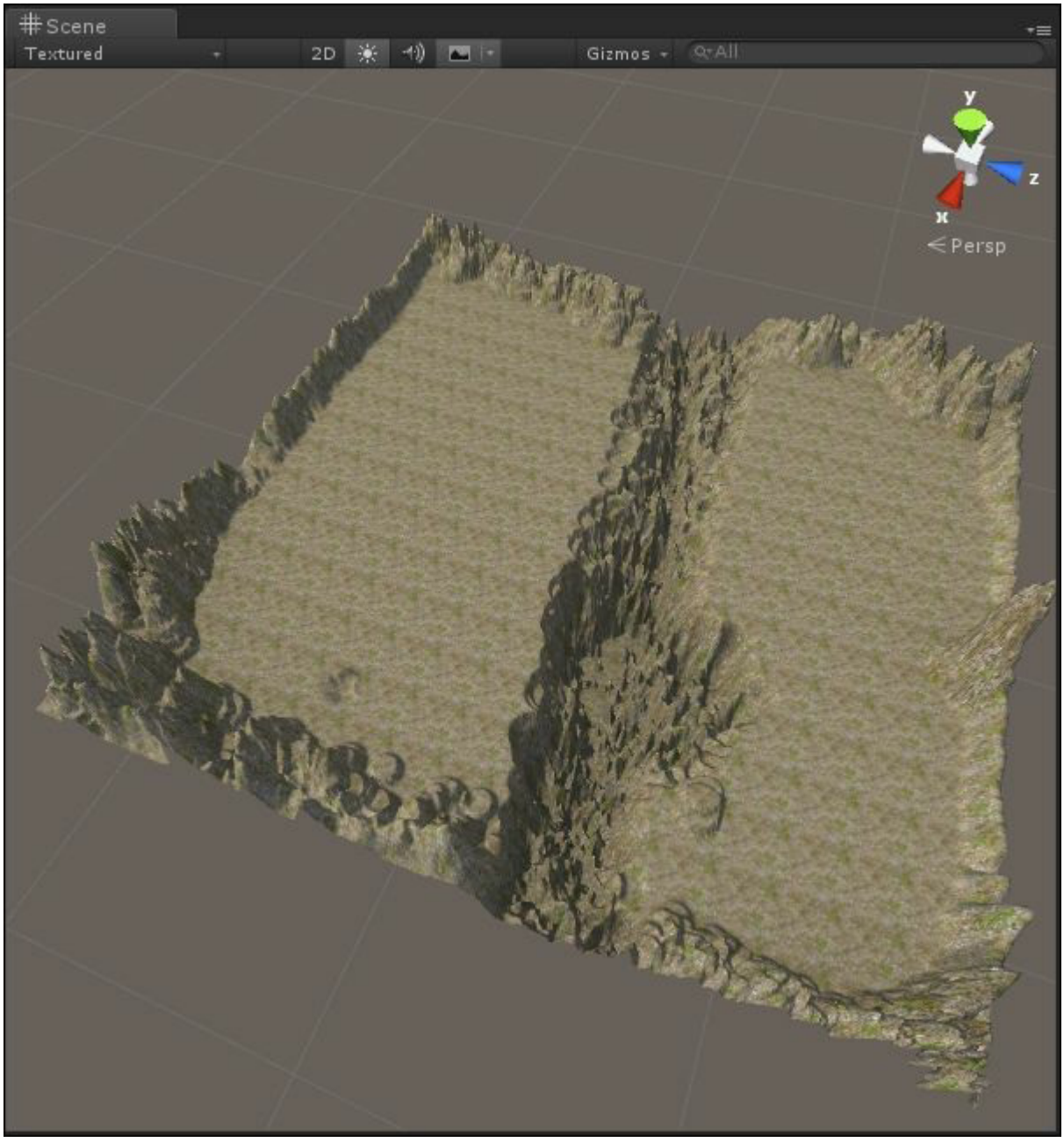
۴. از منو باز شده Add Texture را انتخاب کنید .

۵. در پنجره Add Terrain Texture ، در زیر Texture ، دکمه Select را انتخاب کنید .

۶. در پنجره Select Texture ، آیکون MudRockyAlbedoSpecular را دوبار کلیک کنید .

۷. دکمه Add را کلیک کنید تا این بافت به زمین شما اضافه شود .

حالا بافت به زمین اعمال شده و شبیه به تصویر زیر می شود :



زمین ما کمی واقعی تر شده چون بافت را به آن اعمال کرده ایم . سپس از یک بافت دیگر برای نقاشی کوه ها استفاده می کنیم . مراحل زیر را برای این کار دنبال کنید :

۱. در نمای Hierarchy ، Terrain را انتخاب کنید .

۲. در نمای Inspector ، دکمه Paint Texture را انتخاب کنید .

۳. در نمای Inspector ، دکمه Edit Textures را کلیک کنید .

۴. در منو باز شده Add Texture را انتخاب کنید .

۵. در پنجره Add Terrain Texture در زیر Texture ، دکمه Select را انتخاب کنید .

۶. در پنجره Select Texture پائین بیائید تا آیکون CliffAlbedoSpecular را ببینید . آن را دوبار کلیک کنید .

۷. دکمه Add را کلیک کنید تا این بافت با بافت های دردسترس برای نقاشی ، اضافه شود .

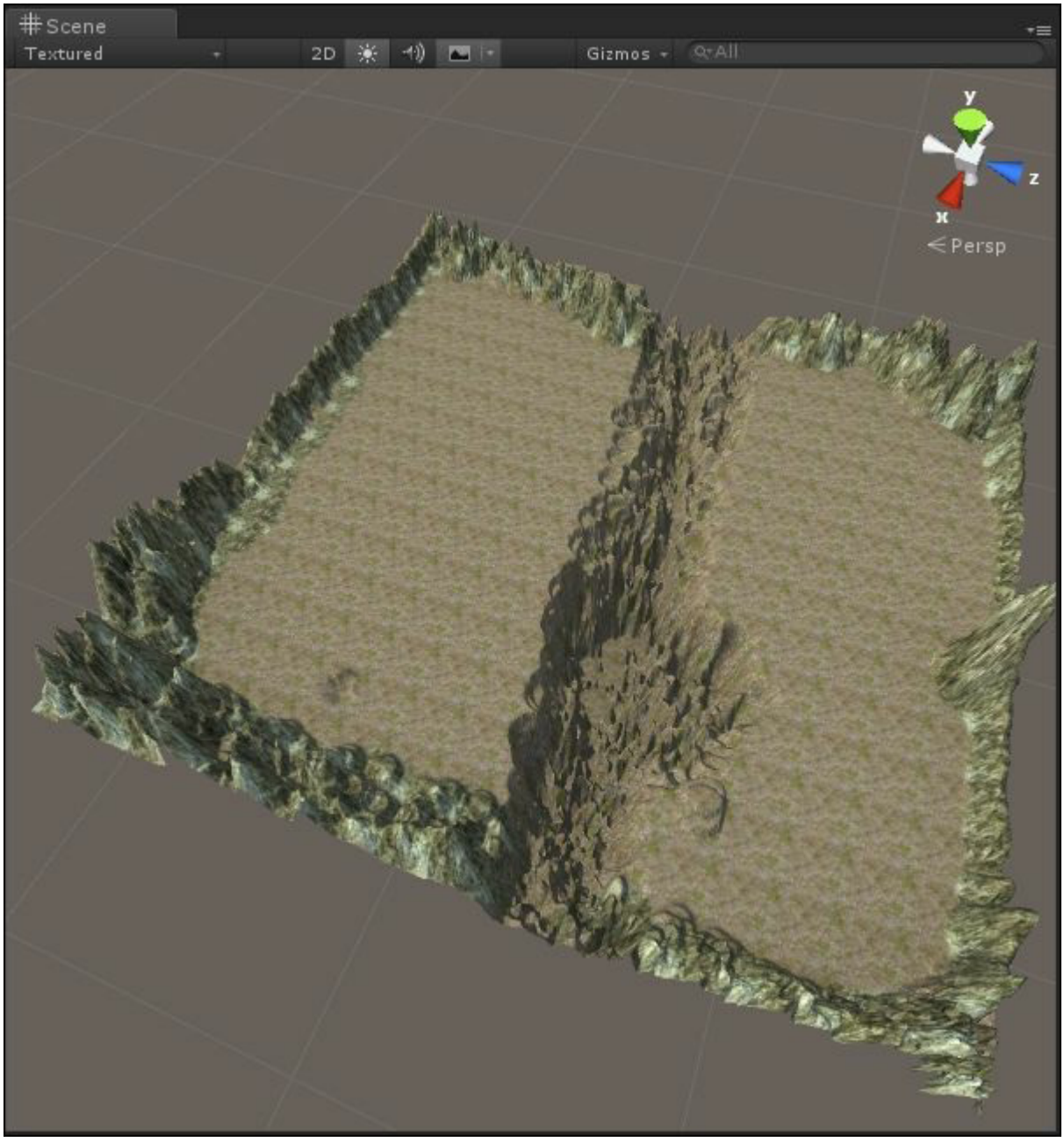
۸. در نمای Inspector ، این بافت جدید را کلیک کنید .

۹. یک قلم مو با اندازه مناسب انتخاب کنید .

۱۰. مکان نمای ماوس را به نمای Scene ببرید .

۱۱. دکمه چپ ماوس را فشار داده و مکان نما را روی کوه حرکت دهید . این کار باعث نقاشی بافت انتخابی روی کوه می شود .

حالا بافت های زمین و کوه ها با هم متفاوت است :



در قسمت بعد ، یک پل ایجاد می کنیم ، درخت ها را می کاریم ؛ آب را اضافه کرده و آسمان را نیز ایجاد می کنیم .

تکمیل محیط بازی

در این قسمت ، چند آیتم دیگر به زمین اضافه می کنیم .

اضافه کردن پل

چندین راه برای ساختن یک پل برای بازی ما وجود دارد . ما می خواهیم بازیکن بتواند از یک طرف نقشه بازی به سمت دیگر برود . شیوه مورد استفاده ما ، به کارگیری یک شکل هندسی بافت بندی شده است . مراحل زیر را دنبال کنید :

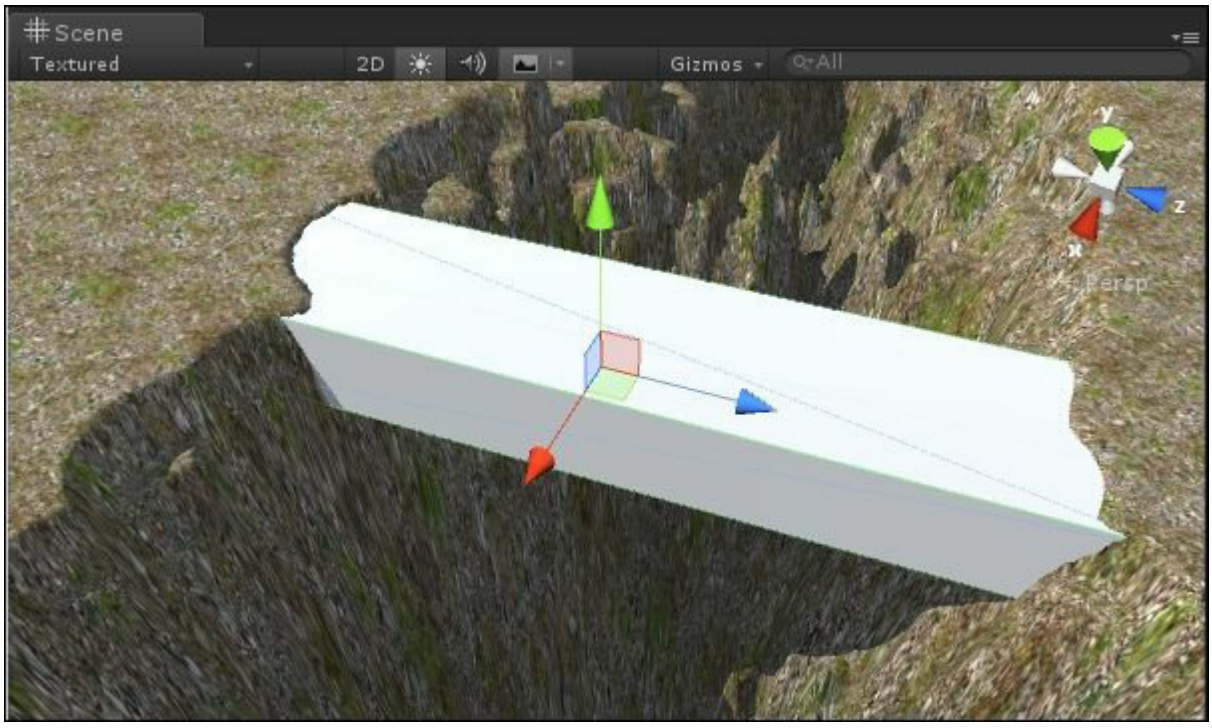
۱. در نمای Scene از ابزار تغییر شکل استفاده کرده و به سمت منطقه ای که می خواهیم پل در آنجا قرار گیرد ، زوم کنید . من پل را در وسط زمین قرار می دهم .

۲. از منو GameObject و بخش 3D Object ، گزینه Cube را انتخاب کنید .

۳. در نمای Hierarchy روی مکعب راست کلیک کرده و Rename را انتخاب کنید . اسم مکعب را Bridge انتخاب کنید .

۴. با استفاده از ابزارهای تغییر شکل ، طول و عرض مکعب را افزایش دهید . اندازه 12, 5, 58 مناسب است .

۵. با استفاده از ابزارهای تغییر شکل ، پل را در محل مناسب قرار دهید . دقت کنید که هر دو لبه پل در زیر سطح زمین قرار گیرند تا دیده نشوند :



۶. حالا بافتی را به پل اعمال می کنیم تا طبیعی تر شود . در نمای Project ، به

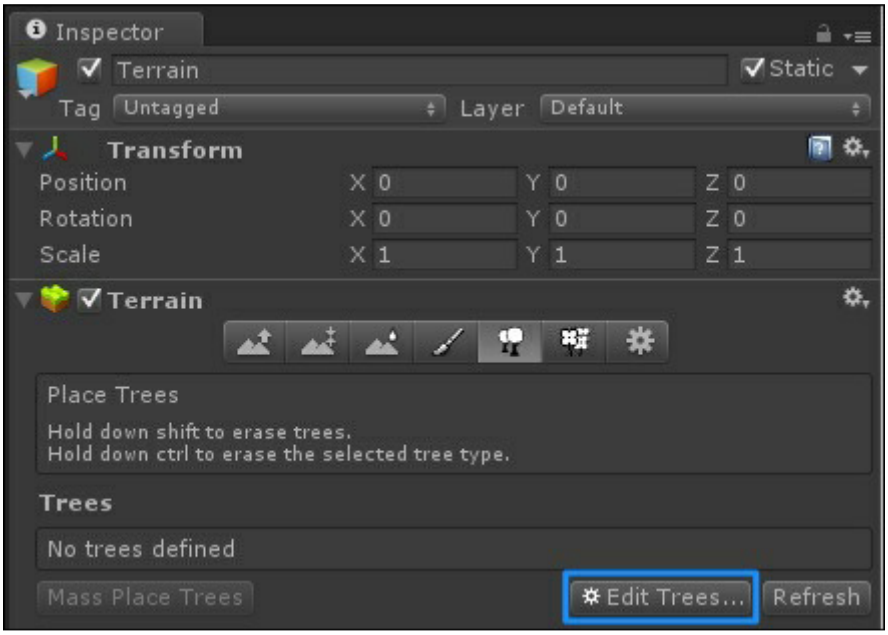
Assets | SampleAssets | Environment | TerrainAssets | SurfaceTextures
بروید و در ستون دوم نمای Project ، GrassRockyAlbedo را انتخاب کنید . در نمای Hierarchy ، این بافت را به سمت پل درگ کنید :

حالا صحنه و پروژه را ذخیره کنید .

کاشتن درخت ها

کاشتن درخت ها در یونیتی آسان تر از دنیای واقعی است . در واقع ، یونیتی به ما اختیاراتی را می دهد . آنها ایجاد درختان از پایه است . این کار زمان زیادی می خواهد اما نتیجه درختان عالی خواهد بود .
ما همچنین می توانیم از درخت ها از پیش آماده نیز استفاده کنیم . تعدادی از آنها در SampleAssets وجود دارند .
با مراحل زیر ، درخت ها را اضافه کنید :

۱. در نمای Scene و با استفاده از ابزار تغییر شکل طوری به سمت بیرون زوم کنید که بتوانید تمام زمین را ببینید .
۲. در نمای Hierarchy ، Terrain را انتخاب کنید .
۳. در نمای Inspector روی دکمه Place Trees کلیک کنید .
۴. زیر این دکمه قسمت Trees را می بینید . مثل تصویر زیر دکمه Edit Trees را کلیک کنید :



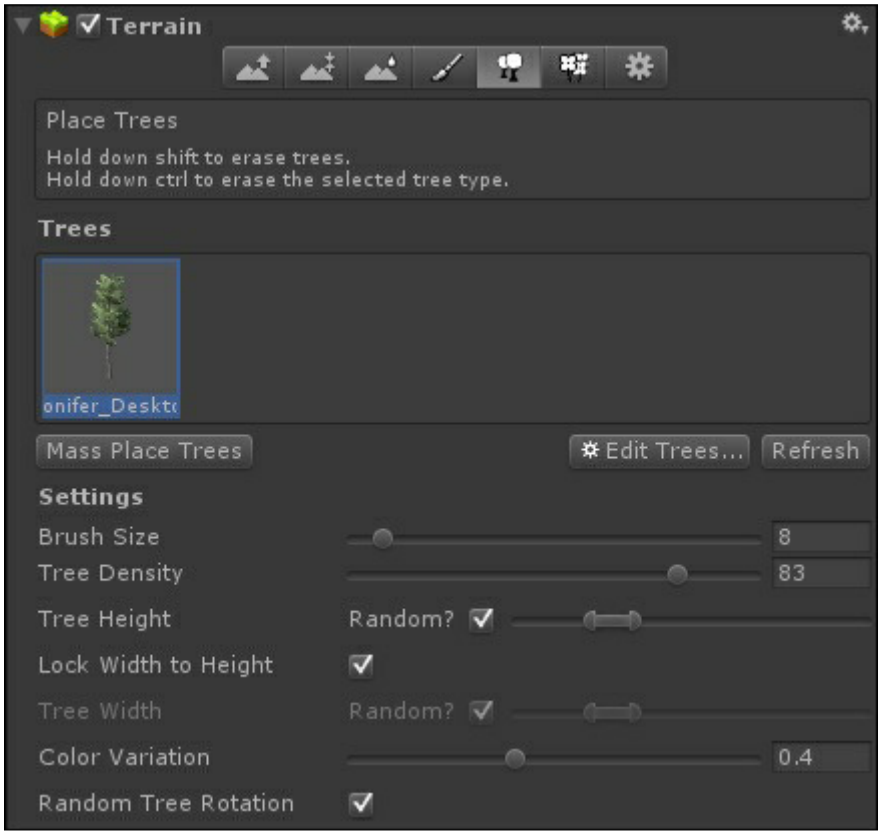
۵. در منو باز شده Add Tree را انتخاب کنید .

۶. در پنجره Add Tree ، روی دایره کوچکی که در سمت راست خط Tree Prefab است ، کلیک کنید . پنجره Select GameObject باز می شود .

۷. در پنجره Select GameObject روی آیکون Conifer_Desktop دوبار کلیک کنید .

۸. در پنجره Add Tree ، دکمه Add را کلیک کنید .

حالا که در نمای Inspector ، یک درخت را به زمین اضافه کرده ایم ؛ می توانیم آنها را به صحنه اضافه کنیم . ابتدا باید تنظیماتی انجام دهیم . به شکل زیر در نمای Inspector و در Settings بخش Terrain این کار را انجام می دهیم :

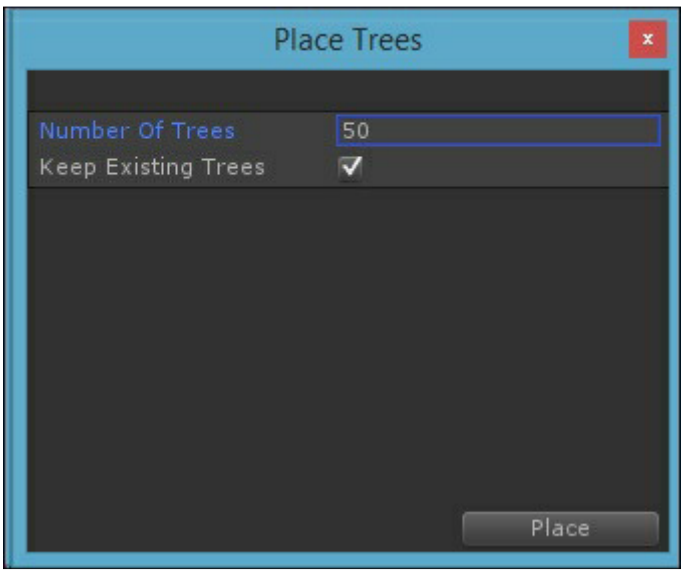


برای اضافه کردن درخت ها به صحنه :

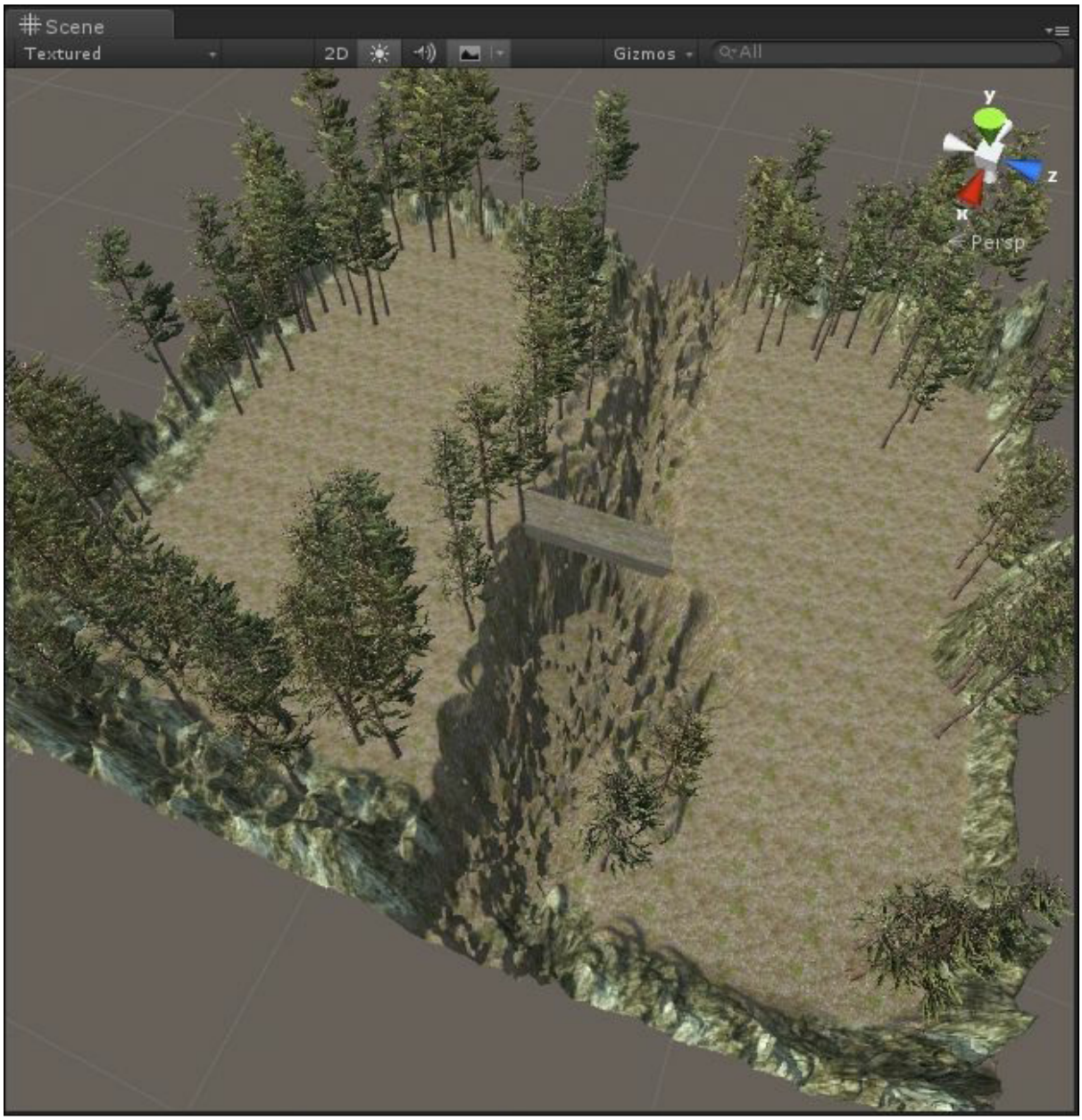
۱. Brush Size را به 8 تغییر داده و اندازه Tree Height را کاهش دهید .

۲. ماوس را به نمای Scene برده و بر روی هر نقطه ای که می خواهید درخت اضافه شود ، کلیک کنید .

۳. یک شیوه سریع برای اضافه کردن درختان زیاد به زمین ؛ استفاده از دکمه Mass Place Trees است که درست در زیر تصویر درخت در نمای Inspector وجود دارد . با کلیک این دکمه ، پنجره Place Trees باز می شود . فیلد Number Of Trees را به 50 تغییر داده و به شکل زیر دکمه Place را کلیک کنید :



۴. اضافه کردن تصادفی درخت ها ، کمی زمان را ذخیره می کند اما ممکن است درخت ها را در جائی قرار دهد که شما نمی خواهید . برای حذف درخت ها ، در نمای Inspector ، دکمه Place Trees را انتخاب کرده و در نمای Scene ، روی درختان با نگه داشتن کلید shift ، کلیک کنید :



اضافه کردن آب

ما از یک شیوه سریع برای اضافه کردن آب استفاده می کنیم . شیوه ای که ما استفاده می کنیم ، اضافه کردن شئ استاندارد water است که همراه با یونیتی می آید . مراحل زیر را دنبال کنید :

۱. در نمای Project ، به

Assets | Sample Assets | Environment | Water (Basic) | Prefabs

بروید . در ستون دوم نمای Project ، دو شئ آب می بینید .

۲. روی شئ WaterBasicDaytime کلیک کرده و آن را به داخل صحنه درگ کنید . می بینید که آب ، یک شئ صاف دایره ای است :

۳. با استفاده از ابزار تغییر شکل ، اندازه و محل قرارگیری آب را طوری تنظیم کنید که رودخانه را بپوشاند و روی پل یا زمین نریزد . زمین شما باید به شکل زیر شود :

ایجاد آسمان

آخرین کاری که می کنیم ، ایجاد آسمان است . ما از Skybox استفاده می کنیم که یک مکعب ۶ پهلو است و برای بازیکن قابل دیدن است و دربالای همه اشیاء دیگر است . برای ایجاد Skybox مراحل زیر را انجام دهید :

۱. در نمای Project روی پوشه Assets کلیک کنید .

۲. Material | Create | Assets را انتخاب کنید . این کار یک شئ متریال در پوشه Assets ایجاد می کند .

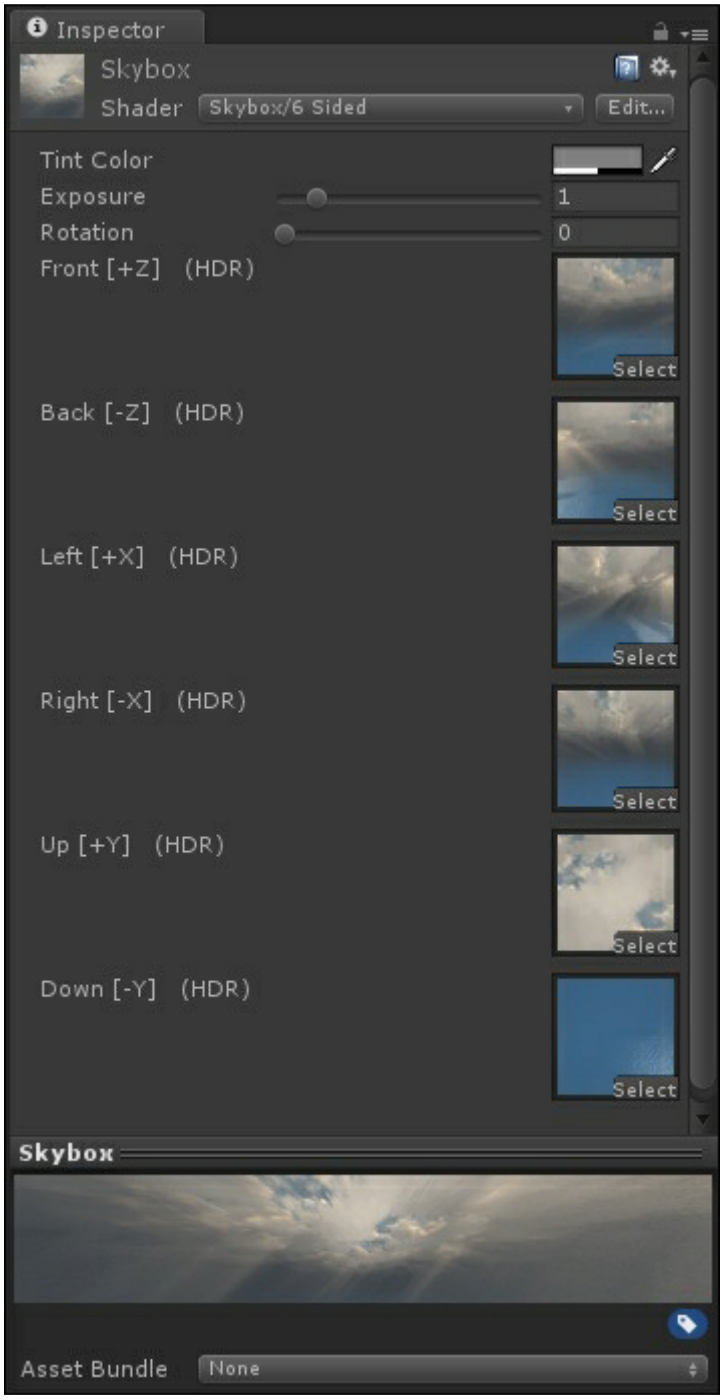
۳. این متریال جدید را Skybox بنامید .

۴. در نمای Project ، Skybox را انتخاب کنید .

۵. در نمـــای Inspector ، گزینـــه Shader بـــه صـــورت پـــیش فـــرض Standard اســـت . آن را بـــه Skybox/ 6 Sided تغییر دهید :

۶. حالا از فایل منابع کتاب که از سایت bazi-dev.ir باید دانلود کنید ، و از داخل پوشه Chapter 2 ، شش فایل تصویری را به داخل پوشه Assets درگ کنید . اسامی این تصاویر با CloudyLightRays شروع می شود .

۷. در نمای Inspector ، هر فایل جدید را به قسمت متناظر درگ کنید . توجه کنید که فایل ها شامل back ، down ، front ، left ، right یا up هستند . هر یک از اینها یک قسمت متناظر در Skybox دارند . بعد از انجام این کار ، نمای Inspector باید به شکل زیر Skybox را نمایش دهد :



۸. در نمای Hierarchy ، Main Camera را انتخاب کنید .

۹. در پائین نمای Inspector ، دکمه Add Component را کلیک کنید . سپس Rendering و Skybox را انتخاب کنید . این یک Component جدید در نمای Inspector برای دوربین اصلی ، ایجاد می کند :

۱۰. در نمای Inspector ، Skybox ای را که ایجاد کرده ایم ، به داخل قسمت Custom Skybox درگ کنید :

حالا در هر دو نمای Scene و Game می بینید که بازیکن یک دید ۳۶۰ درجه ای از Skybox ما دارد .
صحنه و پروژه را ذخیره کنید .

خلاصه

در این فصل ، مفهوم طراحی بازی را به صورت خیلی کوتاه معرفی کرده و بازی Little Farmer Colt را مستندسازی کردیم . طراحی بازی به عنوان نقطه مبنا برای کارهای باقی مانده بازی در طول این کتاب ، عمل می کند . ما همچنین محیط بازی را با ایجاد زمین ، اضافه کردن بافت به زمین ، اضافه کردن کوه ها ، درختان و رودخانه شکل دادیم . ما حتی یک پل ساختیم و فصل را با اضافه کردن آسمان به پایان رساندیم . حالا شالوده اصلی بازی ما ساخته شده است .
در فصل بعد ؛ نگاهی دقیق تر به نحوه داخل کردن و استفاده از منابع در یونیتی ، خواهیم داشت . ما Colt ، کاراکتر اصلی بازی و کشاورز مسن و حیوانات را به پروژه داخل می کنیم .

فصل ۳ : مدیریت منابع بازی

در فصل قبل ، بازیمان را طراحی کرده و محیط بازی را ساختیم . حالا آماده هستیم اشیاء دیگری غیر از آب و درخت را به محیط بازی اضافه کنیم .

در این فصل به بررسی دقیق نحوه وارد کردن منابع به بازی از طریق فروشگاه منابع یونیتی ، می پردازیم . از این منابع به منظور جذاب تر کردن محیط بازی برای کاربران ، استفاده می کنیم . ما گراز ، بچه گراز ، جوجه ، مرغ ، کشاورز مسن و Colt (کشاورز جوان) را اضافه می کنیم . اینها کاراکترهای بازی ما هستند . در ضمن انبار غلات را نیز برای دو کشاورز ، اضافه خواهیم کرد و یک منبع سه بعدی را نیز با استفاده از یک نرم افزار رایگان مدلسازی ایجاد و آن را به بازی ؛ داخل می کنیم .

بعد از مطالعه این فصل :

با منابع یونیتی آشنا خواهید شد .

می توانید منابع یونیتی را به بازی هایتان داخل کنید .

با بسته های یونیتی آشنا می شوید .

می توانید بسته های یونیتی را به بازی هایتان داخل کنید .

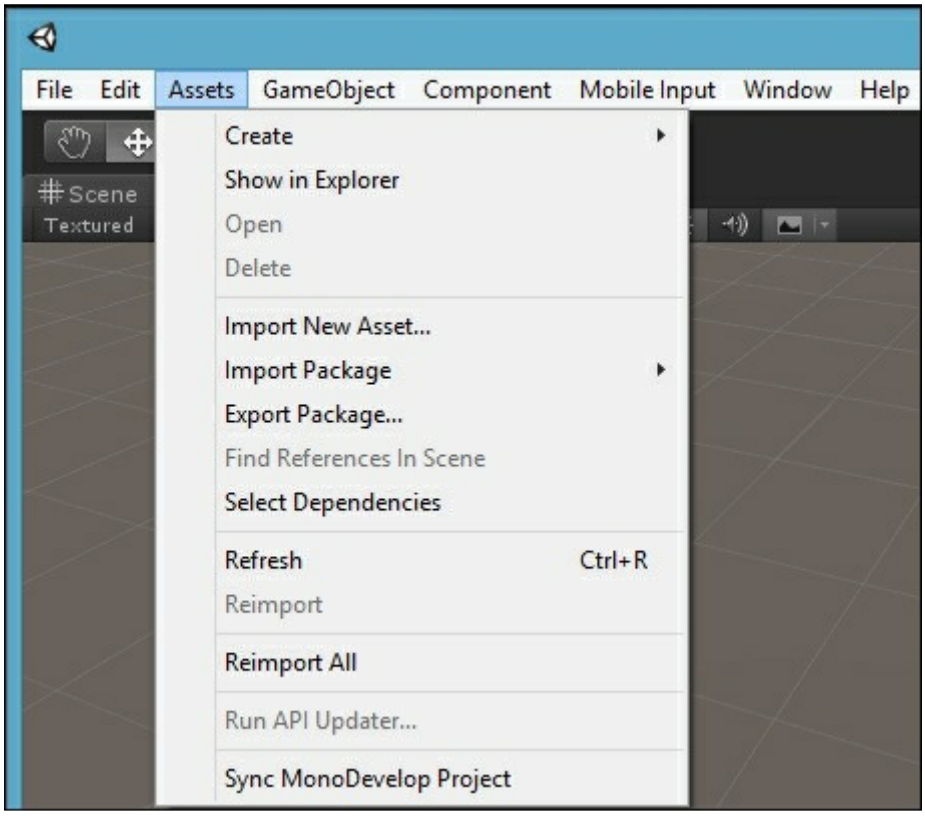
توانائی استفاده از نرم افزار مدل سازی Blender برای ذخیره یک شئ سه بعدی را خواهید داشت .

قادر خواهید بود ، شئ سه بعدی ساخته شده در Blender را به بازی هایتان داخل کنید .

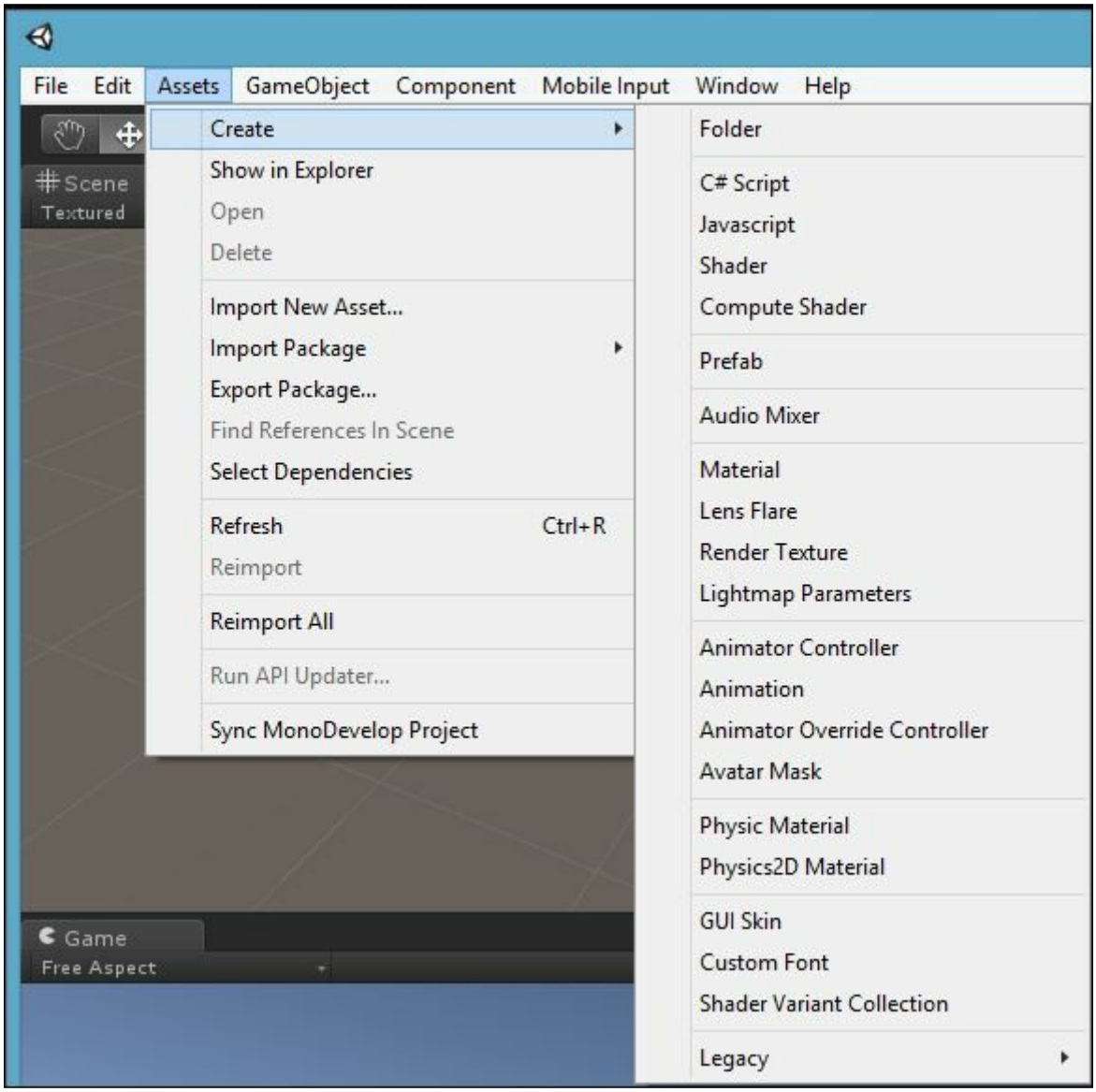
منابع

منابع (Assets) به معنی چیزهای مفید و باارزش است . در یونیتی ، منابع ؛ اشیاء قابل استفاده بازی است . سه نوع اصلی از منابع وجود دارد : منابع یونیتی ، منابع ساخته شده توسط کاربر و منابع ثالث . همانطور که این اسامی نیز گویا هستند ، منابع براساس نحوه ایجادشان دسته بندی شده اند . منابع یونیتی ؛ آنهائی هستند که به صورت رایگان همراه یونیتی وجود دارند و همچنین منابع غیررایگانی که به صورت یک کتابخانه در یک فروشگاه عرضه می شوند . منابع ساخته شده توسط کاربر ، آنهائی هستند که خودتان ایجاد می کنید . ما در ادامه این فصل ، یک منبع سه بعدی ایجاد خواهیم کرد . منابع ثالث ، آنهائی هستند که یک نفر غیر از شما یا یونیتی ؛ آنها را ایجاد کرده است .

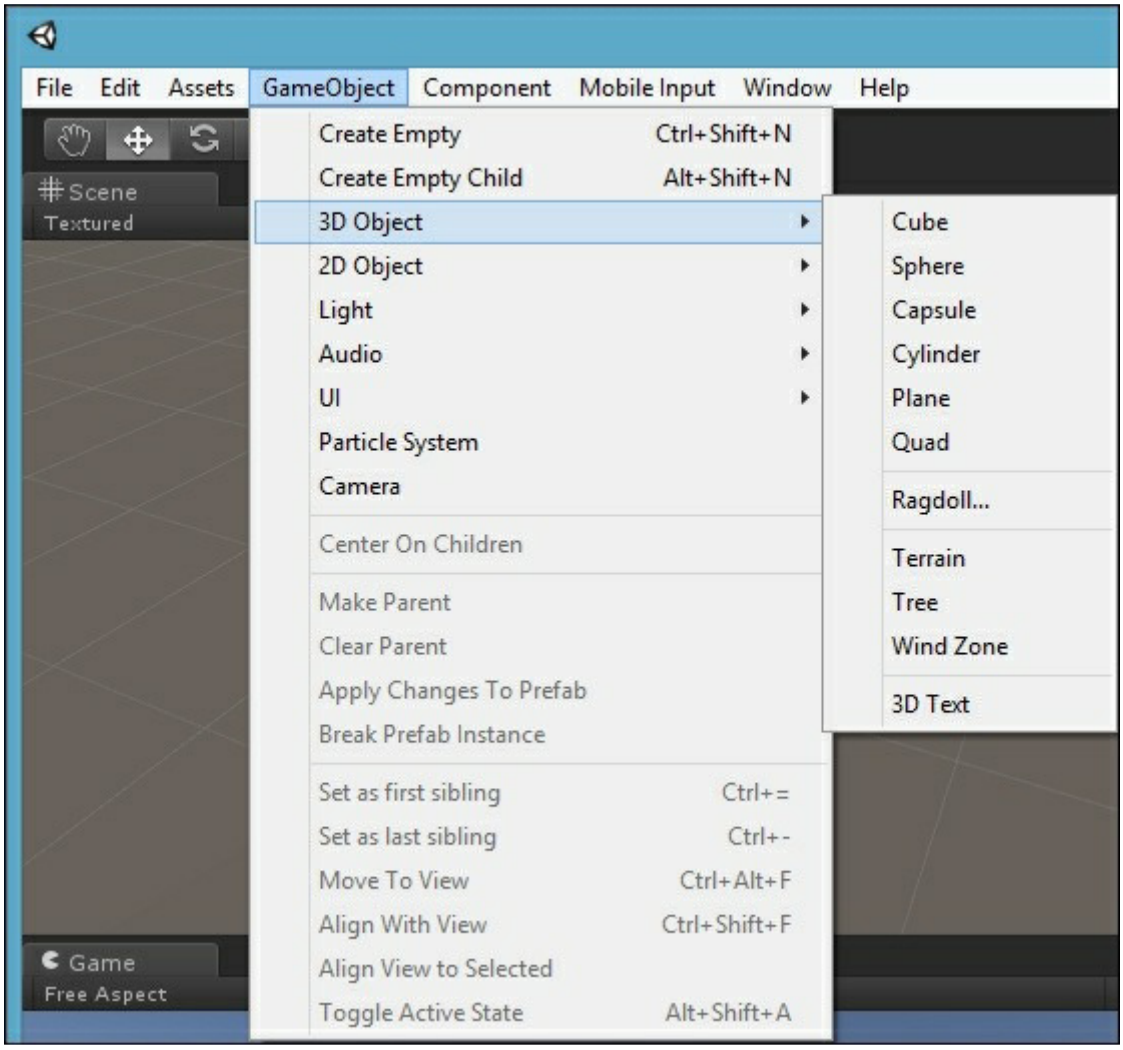
با انتخاب منو Assets ، چندین کارکرد در دسترس ما خواهد بود . کارکردهائی که ما در این فصل بررسی خواهیم کرد عبارتند از : Create ، Import New Asset... و Import Package :



با رفتن به منو Assets | Create می بینید که می توانیم مستقیما در داخل یونیتی ، چندین نوع از منابع را ایجاد کنیم . این منابع می توانند اسکریپت ها ، شیدرها ، متریال ها ، انیمیشن ها و ... باشند . منابع در واقع هر آیتمی است که ما از آن در بازی هایمان استفاده می کنیم . در فصل های بعد ؛ از بعضی از این منابع استفاده خواهیم کرد :



همچنین می توان منابع سه بعدی اولیه مثل یک مکعب یا کره را ایجاد کرد . برای ایجاد یک شئ سه بعدی فیزیکی ، مثل تصویر زیر به منو GameObject | 3D Object رفته و شئ ای را که می خواهید ، انتخاب کنید . در بازی Little Farmer Colt ما به این اشیاء نیازی نداریم . اینها برای تست اشیاء و اسکریپت ها در یونیتی مناسب هستند . بعد از ایجاد آنها ، می توان به راحتی حذفشان کرد :



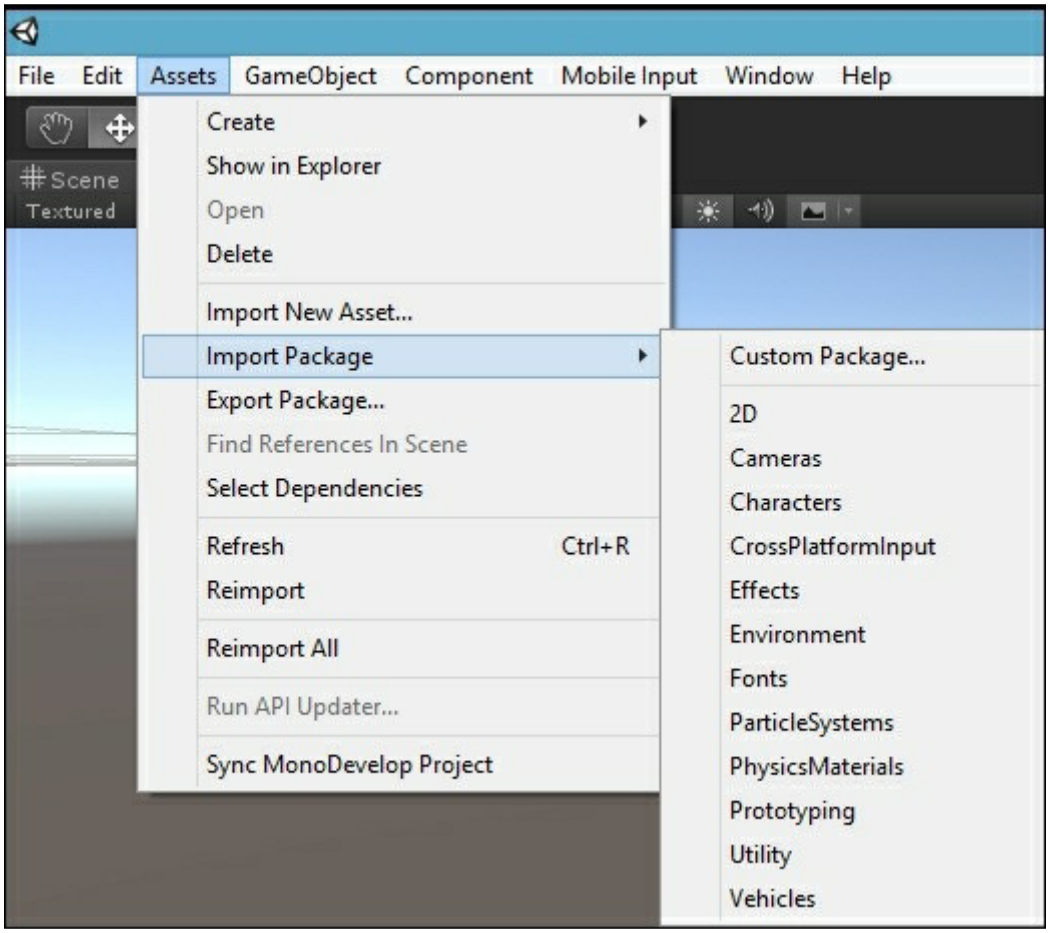
بسته های منبع

در فصل قبل ، ما یک بسته منبع را به پروژه داخل کردیم تا بتوانیم زمین بازیمان را ایجاد و اصلاح کنیم . در این قسمت ، به بررسی این بسته ها پرداخته و نحوه استفاده از آنها در یونیتی و بازی خودمان را می بینیم .

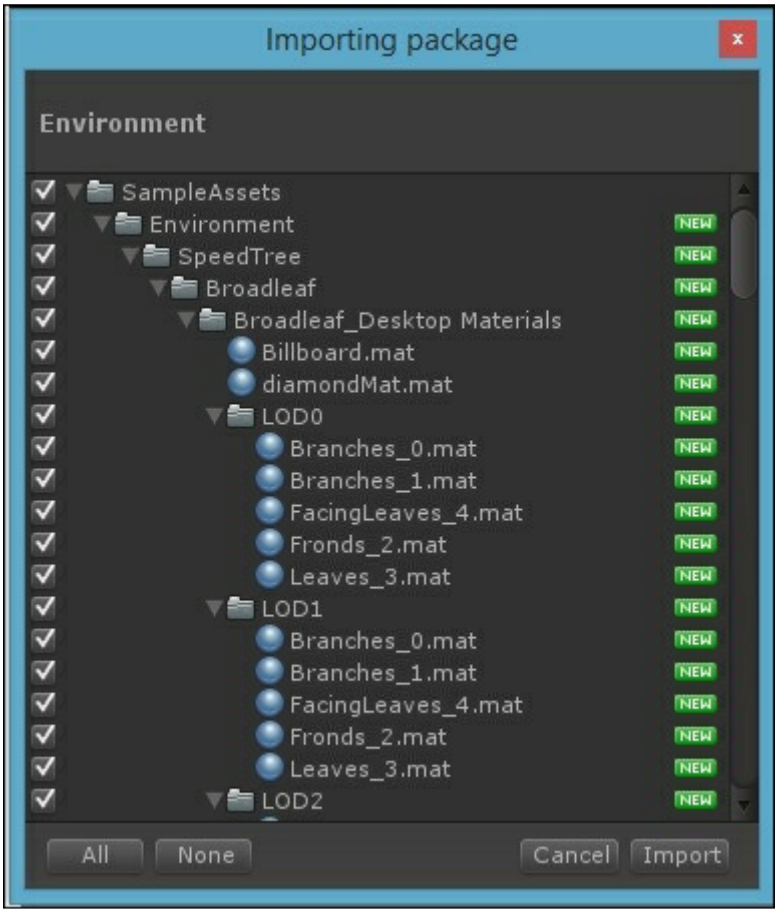
بسته های منبع (Asset packages) ، مجموعه ای از منابع هستند که گرد هم جمع آوری شده اند . برای به اشتراک گذاشتن منابع با دیگران یا برای ذخیره آنها به منظور استفاده در بازی های دیگر ؛ این بسته ها ایجاد می شوند . حتی می توانیم آنها را برای فروش در فروشگاه منابع یونیتی ، ایجاد کنیم .

برای ایجاد یک بسته منبع ، با استفاده از نمای Project ، همه منابعی را که می خواهید در بسته وجود داشته باشد ، انتخاب کنید . سپس دو راه دارید . راه اول راست کلیک و انتخاب Export Package است . راه دوم رفتن به منو Assets | Export Package است . هر دو راه ، یک خروجی تولید می کند .

همچنین می توانیم بسته های منبع را به یونیتی داخل کنیم . برای این کار به منو Assets | Import Package بروید :



هنگام داخل کردن بسته های منبع ، می توانید یکی از ۱۲ بسته پیشنهادی یونیتی را انتخاب کنید . این بسته ها به شکل بالا در هنگام انتخاب منو Assets | Import Package لیست می شوند . انتخاب هر یک از این ها باعث بازشدن بسته و سپس نمایش محتویات بسته می شود :



بعد از بازشدن بسته و قبل از اینکه منابع موجود در بسته به پروژه داخل شود ؛ فرصت بازدید ، انتخاب و عدم انتخاب منابع موجود را دارید . سپس دکمه Import را کلیک می کنید . یونیتی ، بسته را در پروژه شما قرار خواهد داد . در نمای Project می توانید منابع را ببینید . فایل های اصلی منابع روی هارد دیسک ذخیره می شوند .

همچنین می توانید بسته دلخواهتان را با رفتن به منو Assets | Import Package | Custom Package داخل کنید . با انتخاب این گزینه پنجره ای باز می شود که در آن می توانید محل ذخیره بسته مورد نظرتان روی هارد دیسک را مشخص کنید . پسوند فایل بسته های منابع یونیتی unitypackage. است .

فروشگاه منابع یونیتی

یونیتی فروشگاهی به نام فروشگاه منابع یونیتی (Unity Asset Store) را اداره می کند . در این فروشگاه ، تعداد زیادی از منابع در دسترس کاربران یونیتی است . آدرس فروشگاه http://assetstore.unity3d.com است . در داخل یونیتی نیز می توان پنجره ای برای نمایش این فروشگاه باز کرد . برای این کار به منو Window | Asset Store بروید .

شما یک دسته بندی گروهی در گوشه بالائی سمت راست فروشگاه می بینید :

کلیک آیکون مثلثی شکل سمت چپ هر گروه ، زیرگروه ها را نمایش می دهد . با کلیک هر دسته ، محتویات آن قابل نمایش می شود . کلیک هر منبع ، مشخصات منبع را نمایش می دهد . این مشخصات شامل اطلاعات زیر است :

عنوان

ناشر

رتبه بندی

قیمت

دکمه خرید یا در مورد منابع رایگان ، دکمه "Open in Unity"

ورژن یونیتی مورد نیاز

توضیحات

محتویات بسته

اندازه فایل

ورژن منبع

تصاویر

ویدئوها

لینک وب سایت ناشر

چند چیز در خصوص به دست آوردن منابع از فروشگاه وجود دارد که باید دقت کنید . ابتدا باید مطمئن شوید که حقوق لازم برای استفاده از منبع مورد نظرتان را داشته باشید . مثلا اگر حق استفاده تجاری را ندارید ، نباید از آن منبع در بازی های تجاری استفاده کنید .

نکته بعدی ، اندازه فایل مورد نظر است . خوشبختانه ؛ این اطلاعات بخشی از آنچه است که در هنگام پیش نمایش یک منبع ؛ مشاهده می کنید .
در بخش بعدی ، فروشگاه منابع یونیتی را بازدید کرده ، یک منبع انتخاب می کنیم و آن را به پروژه بازی مان اضافه خواهیم کرد .

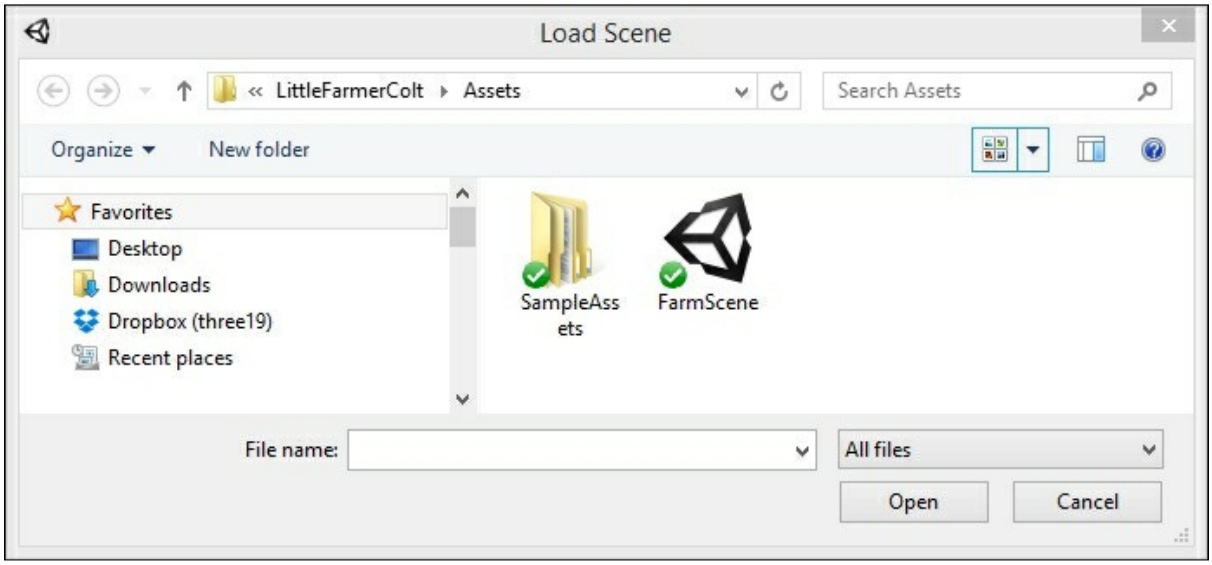
اضافه کردن منابع به بازی

اجازه دهید به بازی Little Farmer Colt چند منبع اضافه کنیم . ابتدا از فروشگاه منابع یونیتی بازدید کرده ، یک منبع را انتخاب می کنیم ، آن را بررسی و دانلود کرده و به بازی مان اضافه می کنیم . سپس یک بسته منبع دلخواه را داخل بازی خواهیم کرد .

استفاده از فروشگاه منابع یونیتی

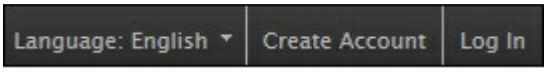
مراحل زیر نحوه پیداکردن یک منبع در فروشگاه منابع یونیتی و اضافه کردن آن به بازی مان را نشان می دهد :

۱. یونیتی را اجرا و پروژه تان را باز کنید . یادآوری می کنم ما پروژه را LittleFarmerColt نامیده ایم .
۲. از منو File | Open Scene صحنه را باز کنید . پنجره Load Scene باز می شود که پوشه Assets پروژه شما را نیز نشان می دهد . اسم صحنه من FarmScene است . این فایل را دوبارکلیک کنید تا باز شود :



حالا آماده باز کردن فروشگاه منابع یونیتی هستیم . ما به دنبال یک شئ مناسب برای بازی خودمان هستیم .
۳. به منو Window | Asset Store بروید تا فروشگاه در یک پنجره جدید باز شود .

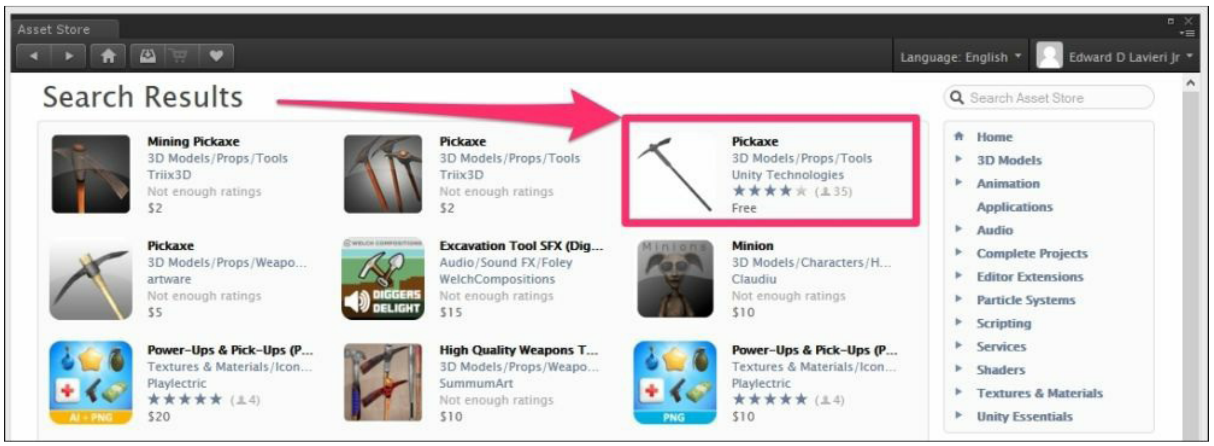
۴. وارد حساب کاربری خود در یونیتی شوید . لینک ورود Log In در گوشه بالائی سمت راست پنجره است :



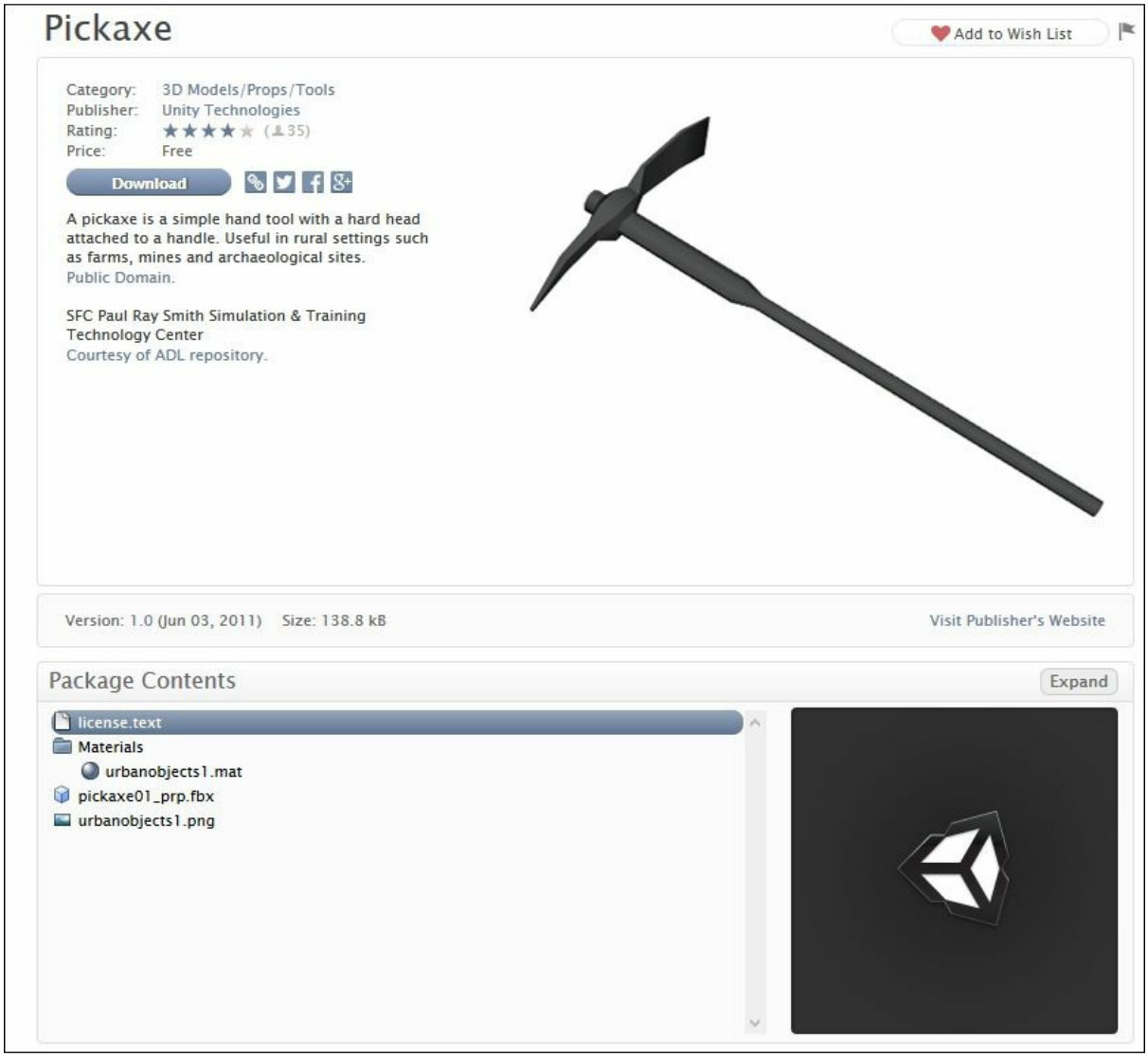
۵. بعد از ورود ، اسم شما در بالای پنجره نمایش داده می شود :

۶. با استفاده از قسمت جستجو ، کلمه pickaxe را جستجو کنید :

۷. چندین منبع پیدا می شود . نتایج جستجو را به دنبال pickaxe از Unity Technologies بگردید :

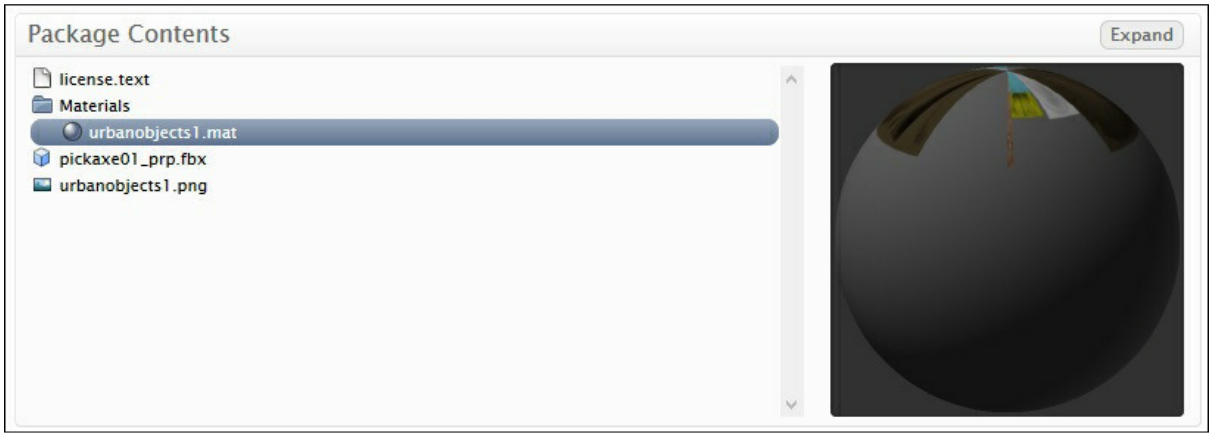


۸. روی تصویر کلنگ کلیک کنید تا جزئیات این منبع را ببینید :

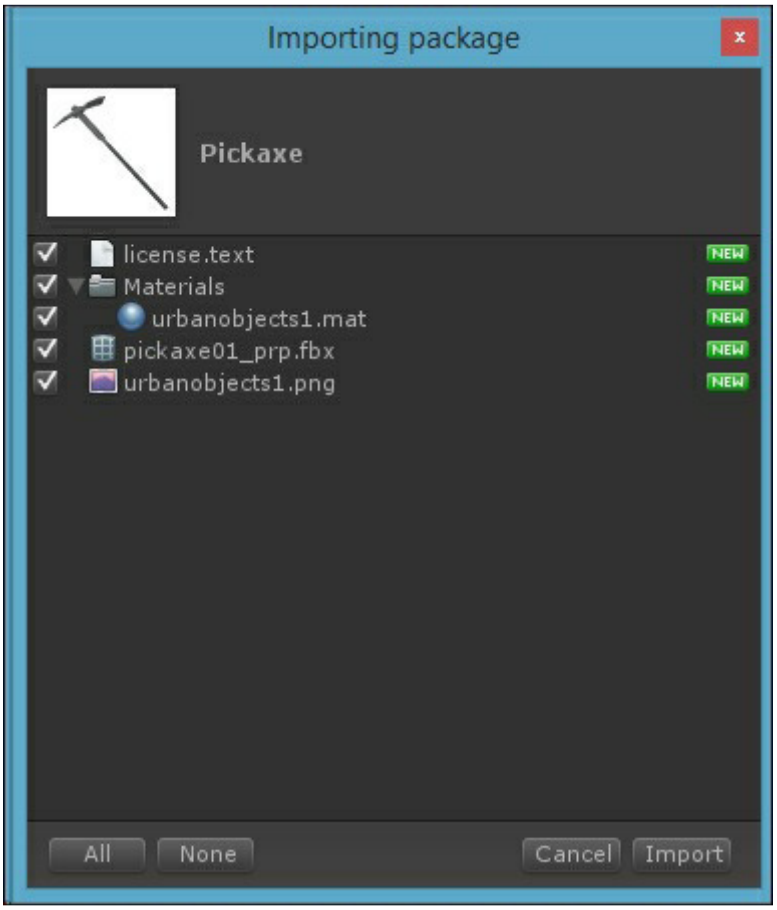


انواع مختلفی از اطلاعات درباره این منبع به شما ارائه خواهد شد . دکمه دانلود این منبع (Download) نیز وجود دارد . در پائین نیز ورژن منبع ، اندازه فایل و لینک سایت ناشر نشان داده شده است :

در قسمت محتویات بسته (Package Contents) نیز لیستی از محتویات بسته نشان داده می شود . بسته pickaxe شامل ۴ عضو است . یک فایل متنی که اطلاعات گواهینامه را نمایش می دهد ، یک فایل متریال ، شئ کلنگ و یک فایل png.
روی فایل mat. کلیک کنید تا پیش نمایش آن را ببینید :



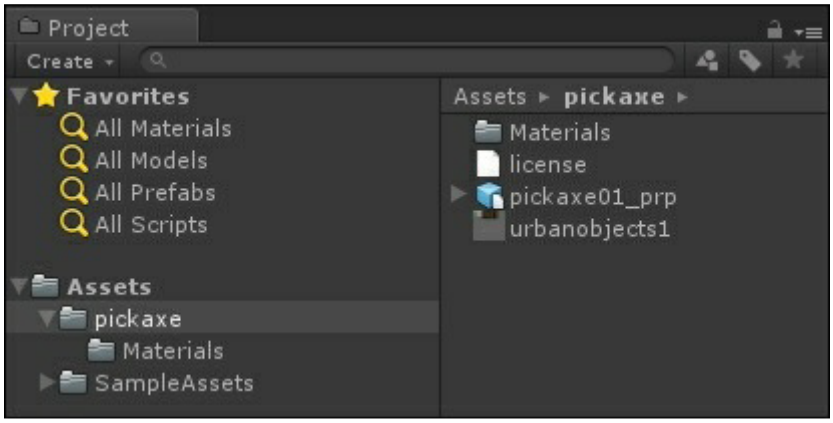
۹. دکمه Download را کلیک کنید . پنجره Importing Package باز می شود . به صورت پیش فرض همه منابع موجود در بسته انتخاب شده است :



۱۰. دکمه Import را کلیک کنید .

۱۱. در نمای Project ، روی پوشه Assets راست کلیک کرده و به منو Assets | Create | Folder بروید . پوشه را pickaxe بنامید .

۱۲. در نمای Project ، محتویات بسته pickaxe را به داخل پوشه ایجاد شده ، درگ کنید . بعد از این کار ، نمای Project به شکل زیر می شود :



۱۳. پنجره فروشگاه منابع یونیتی را ببندید .

حالا شئ کلنگ در دسترس بازی شماست . فعلا آن را در نمای Project به حال خود بگذارید !

داخل کردن یک بسته منبع دلخواه

دو بسته منبع در فایل منابع کتاب که آن را قبلا از سایت دانلود کرده ایم در پوشه Chapter 3 وجود دارد . اولین بسته character_assets.unitypackage است و محتوی مدل های هر دو کشاورز ، بچه گراز ، گراز ، جوجه و مرغ است . دومین بسته building_assets.unitypackage است و شامل دو ساختمان انبار غله برای هر یک از کشاورزهاست .

داخل کردن کاراکترهای بازی

برای بارگذاری کاراکترهای بازی به داخل بازیتان ، مراحل زیر را دنبال کنید :

۱. فایل منابع کتاب را از سایت www.bazi-dev.ir دانلود کنید و آن را از حالت فشرده خارج کنید .

۲. یونیتی را اجرا و پروژه بازی را باز کنید .

۳. به منو Assets | Import Package | Custom Package بروید . این کار پنجره Import Package را باز می کند .

۴. در پنجره Import Package به مسیری که فایل های بسته منابع را ذخیره کرده اید رفته و فایل character_assets.unitypackage را انتخاب کنید .

۵. دکمه Open را کلیک کنید .

۶. یونیتی بسته را باز کرده و پنجره Importing Package را نمایش می دهد . دکمه Import را کلیک کنید . بعد از اتمام این کار ، در نمای Project ، چند شئ جدید می بینید :

در اینجا مدل های chick (مرغ) ، chicken (جوجه) ، farmer (کشاورز) ، farmer boy (پسر کشاورز) ، pig adult (گراز) و piglet (بچه گراز) را می بینید . در فصل های آینده با اینها کار خواهیم کرد .
پروژه را ذخیره کنید .

داخل کردن ساختمان ها

برای داخل کردن ساختمان ها نیز فایل building_assets.unitypackage را انتخاب کرده و مراحل قسمت قبلی را دنبال کنید . در انتها و در نمای Project ، منابع barn1 و barn2 در دسترس خواهند بود . پروژه را نیز ذخیره نمائید .

استفاده از نرم افزار Blender برای ساختن منابع بازی

چندین ابزار وجود دارد که می تواند به منظور ایجاد اشیاء سه بعدی برای استفاده در یونیتی ، به کار گرفته شود . یکی از اینها Blender است . Blender یک نرم افزار رایگان و متن باز مخصوص مدل سازی سه بعدی است . کتاب ها و آموزش ها زیادی در خصوص نحوه استفاده از این نرم افزار و تولید منابع مورد نیاز وجود دارد . آموزش نحوه ایجاد این منابع از پایه ، خارج از محدوده این کتاب است .

کاری که ما در اینجا انجام می دهیم ، استفاده از شئ Blender و ذخیره آن به فرمت قابل استفاده یونیتی است . برای این کار ، مراحل زیر را دنبال کنید :

۱. یک کپی از نرم افزار Blender را از سایت http://www.blender.org دانلود کنید .
۲. فایل corn_stalk.blend را که در پوشه Chapter 3 فایل منابع کتاب دانلود شده ، انتخاب کنید .

۳. Blender را اجرا کنید .

۴. منو File | Open را انتخاب کنید .

۵. فایل corn_stalk.blend را که قبلا دانلود کردیم ، انتخاب کنید .

۶. دکمه Open Blender File را انتخاب کنید . حالا می توانید شئ را در Blender ببینید :

حالا باید شئ را به فرمت قابل استفاده یونیتی تبدیل کنیم . برای این کار :

۱. به منو File | Export | FBX (.fbx) رفته تا شئ را به شکل یک فایل fbx. ذخیره کنیم .

۲. مسیری را برای ذخیره فایل انتخاب کنید .

۳. دکمه Export FBX را کلیک کنید .

۴. یونیتی را اجرا و پروژه را باز کنید .

۵. منو Assets | Import New Asset را انتخاب کنید . پنجره Import New Asset باز می شود .

۶. فایل fbx. را که ذخیره بودید ، انتخاب کرده و دکمه Import را کلیک کنید .

۷. Blender را ببندید .

۸. پروژه یونیتی را ذخیره کنید .

حالا در نمای Project ، مدل بوته ذرت را می بینید .

خلاصه

در این فصل ، ما منابع یونیتی و بسته های منبع را معرفی کردیم . شما از فروشگاه منابع یونیتی بازدید کردید و منابع مورد نیازتان را انتخاب و دانلود کردید . همچنین درخصوص نحوه ایجاد اشیاء سه بعدی با استفاده از نرم افزار رایگان مدل سازی Blender ، نکاتی را یاد گرفتید .

در فصل بعد ؛ کاراکترهای بازی را انیمیت و متحرک سازی خواهید کرد . انیمیشن های مورد استفاده ، شامل انیمیشن idle (حالت بیکاری) برای همه کاراکترها و انیمیشن talk (صحبت کردن) برای کاراکترهای کشاورز خواهد بود . Colt یعنی کارکتری که توسط بازیکن کنترل می شود ، انیمیشن های walk (راه رفتن) و take (برداشتن) را نیز خواهد داشت . انیمیشن take برای جمع آوری محصول غله و آب است . در ضمن ، برای سرگرمی ، توانائی پرش زدن را نیز به Colt اضافه خواهیم کرد !

فصل ۴ : متحرک سازی کاراکترهای بازی

تا به حال ، طراحی بازی ما تکمیل شده است ، محیط بازیمان ایجاد شده و کاراکترهای بازی و منابع دیگر را به بازیمان داخل کرده ایم . کاراکترهای بازی که اضافه کرده ایم شامل گراز ، بچه گراز ، جوجه ، مرغ ، کشاورز مسن و Colt (کشاورز جوان) است . اگر کاراکترهای بازی مثل یک سنگ ، ثابت و بی تحرک بایستند ، بازی خیلی خسته کننده می شود . ما با اضافه کردن انیمیشن به هریک از این کاراکترها ، زندگی را به آنها هدیه می کنیم !

ما به هر حیوان ، ۲ انیمیشن می دهیم تا در هنگامی که در حالت بیکاری و در حالت غذاخوردن هستند ، تحرک داشته باشند . کشاورز مسن ، انیمیشن های صحبت کردن و بیکار بودن را خواهد داشت . کشاورز جوان ، کاراکتری است که توسط بازیکن کنترل می شود ، بنابراین قادر به راه رفتن ، صحبت کردن ، دویدن ، برداشتن و حالت بیکاربودن خواهد بود .

بعد از مطالعه این فصل :

با انیمیشن های یونیتی آشنا می شوید .

با کنترل کننده بازیکن آشنا می شوید .

قادر به پیش نمایش انیمیشن ها خواهید بود .

می توانید کاراکترهای بازی را متحرک سازی کنید .

می توانید کلیپ انیمیشن ، ایجاد کنید .

اصول انیمیشن

انیمیشن به صورت حرکت شبیه سازی شده تعریف می شود که به وسیله نمایش یک سری از تصاویر یا فریم ها ، ایجاد می شود . در یونیتی ، انیمیشن ها به شیوه ای که در فیلم ها به کارگیری می شوند ، نمایش داده می شوند به این صورت که در هر ثانیه ، یک تعداد مشخص از فریم ها به تدریج و پشت سرهم ، نمایش داده می شوند . ما از انیمیشن ها برای جان دادن به کاراکترها استفاده می کنیم .

هنگامی که بازیکن از کلیدهای جهتی یا دستگاه های ورودی دیگری برای حرکت دادن کاراکتر اصلی در اطراف صفحه ، استفاده می کند ؛ بازی به ورودی کاربر واکنش نشان داده و انیمیشن مناسب را پخش می کند . مثلا هنگامی که بازیکن ، کلید جهتی سمت چپ را فشار می دهد ، کاراکتر اصلی بازی به سمت چپ حرکت می کند . در یونیتی ، این را انیمیشن راه رفتن می گویند .

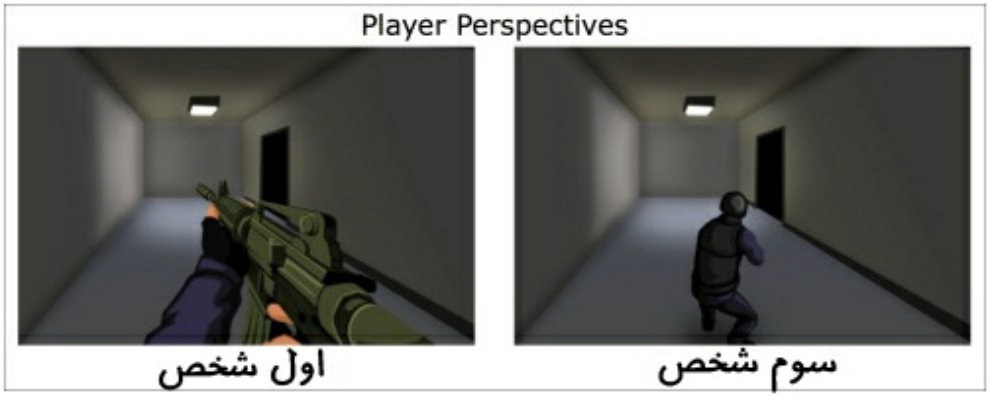
انیمیشن ها معمولا با استفاده از ابزارهای مدل سازی سه بعدی مثل Blender ، 3D Studio Max یا Maya ایجاد می شوند . همانطور که بعدا خواهید دید ، از داخل یونیتی هم می توانیم انیمیشن ها را ایجاد کنیم .

در بازی ما ، درخصوص انیمیشن ، دو نکته حائز اهمیت است . اول اینکه باید کاراکتر کشاورز جوان را طوری متحرک سازی و انیمیت کنیم که کاربر بتواند جائی که می رود و کاری که انجام می دهد را کنترل کند . اولین چیزی که نیاز است ، ایجاد یک کنترل کننده کاراکتر است .

کنترل کننده های کاراکتر

در بازی ها ، دو نوع کنترل کننده بازیکن وجود دارد : آنهائی که توسط انسان کنترل می شوند و آنهائی که توسط کامپیوتر کنترل می شوند . اولی را کاراکتر بازیکن می گویند و دومی به کاراکتر غیربازیکن (non-player) معروف است . کاراکترهای بازیکن می توانند هرآنچه که شما می توانید تصور کنید ، باشند . در بازی های مدرن ، کاراکترهای بازیکن معمولا انسان یا شبه انسان هستند . البته نمی توان گفت امکان ایجاد بازی که در آن ، کاراکتر بازیکن یک درخت از ریشه کنده شده یا یک سنگ با هفت پا باشد ، وجود ندارد ! در بازی ما ، کاراکتر بازیکن ، یک پسربچه به نام Colt است .

یونیتی از یک شئ بازی به نام کنترل کننده کاراکتر (character controller) استفاده می کند تا کاربر را قادر به کنترل کردن کند . در یونیتی دو نوع از این کنترل کننده کاراکتر وجود دارد : کنترل کننده اول شخص و کنترل کننده سوم شخص . اینها اشاره به نوع دید کاربر در طول بازی دارند . تصویر زیر دید بازیکن را نشان می دهد :



با کنترل کننده اول شخص ، فقط بخشی از کاراکتر بازیکن روی صفحه ، قابل دیدن خواهد بود . اغلب بازی ها ، دست ها ، بازوها ، اسلحه یا ترکیبی از اینها را نشان می دهند . منظره بازی درواقع از طریق چشمان کاراکتر ، دیده می شود . کنترل کننده سوم شخص اجازه می دهد همه کاراکتر در طول گیم پلی دیده شود .

کنترل کننده اول شخص

ما در بازی Little Farmer Colt ، نیازی به کنترل کننده اول شخص نداریم اما برای بازی های دیگری که در یونیتی می سازید ، باید با نحوه ایجاد آن آشنا باشید .

برای ایجاد یک کنترل کننده اول شخص :

۱. یونیتی را اجرا کنید .

۲. یک پروژه 3D ایجاد کنید .

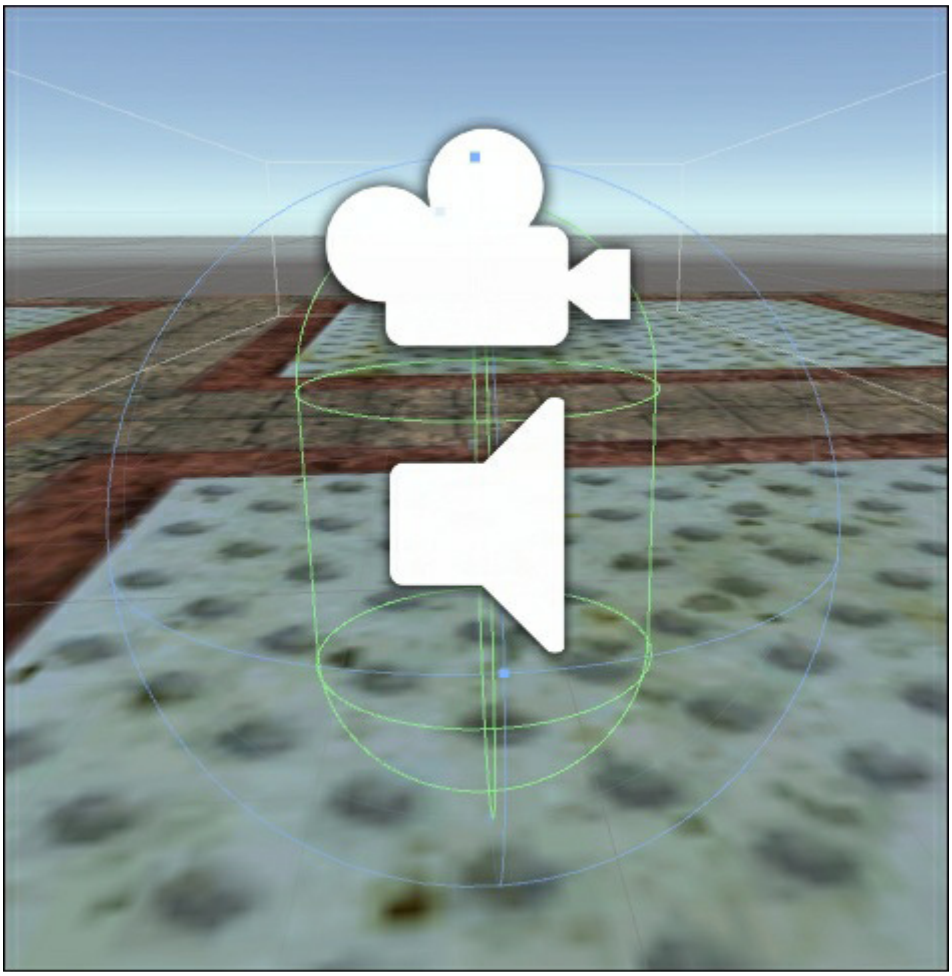
۳. بسته های منبع Characters و Environment را داخل پروژه کنید .

۴. یک زمین جدید ایجاد کرده و بافتی را به آن اعمال کنید . این کار به شما کمک می کند ، شئ کنترل کننده اول شخص را در نمای Scene ببینید .

۵. در نمای Project ، گزینه زیر را انتخاب کنید :

Assets | Sample Assets | Characters | FirstPersonCharacter | Prefabs

۶. در ستون دوم نمای Project ، شئ FPSController را انتخاب کرده و به داخل نمای Scene درگ کنید . با رها کردن ماوس ، یک کنترل کننده اول شخص در صحنه خواهید داشت :



شکل کپسول ، کنترل کننده اول شخص را نشان می دهد . این کنترل کننده می تواند به یک آواتار متصل شود تا شبیه به یک کاراکتر واقعی دیده شود . چون این یک کنترل کننده اول شخص است ، هنگام اجرای بازی ، هیچ بخشی از کاراکتر را نخواهید دید .
برای تست این کنترل کننده در هنگام اجرای بازی ، دکمه play را کلیک کنید . با استفاده از کلیدهای جهتی ، می توانید کنترل کننده را در اطراف ، حرکت دهید .

ایجاد یک کنترل کننده سوم شخص

یونیتی همراه با یک کنترل کننده سوم شخص به عنوان بخشی از بسته character می آید . آواتار متصل به این کنترل کننده ، Ethan نام دارد :

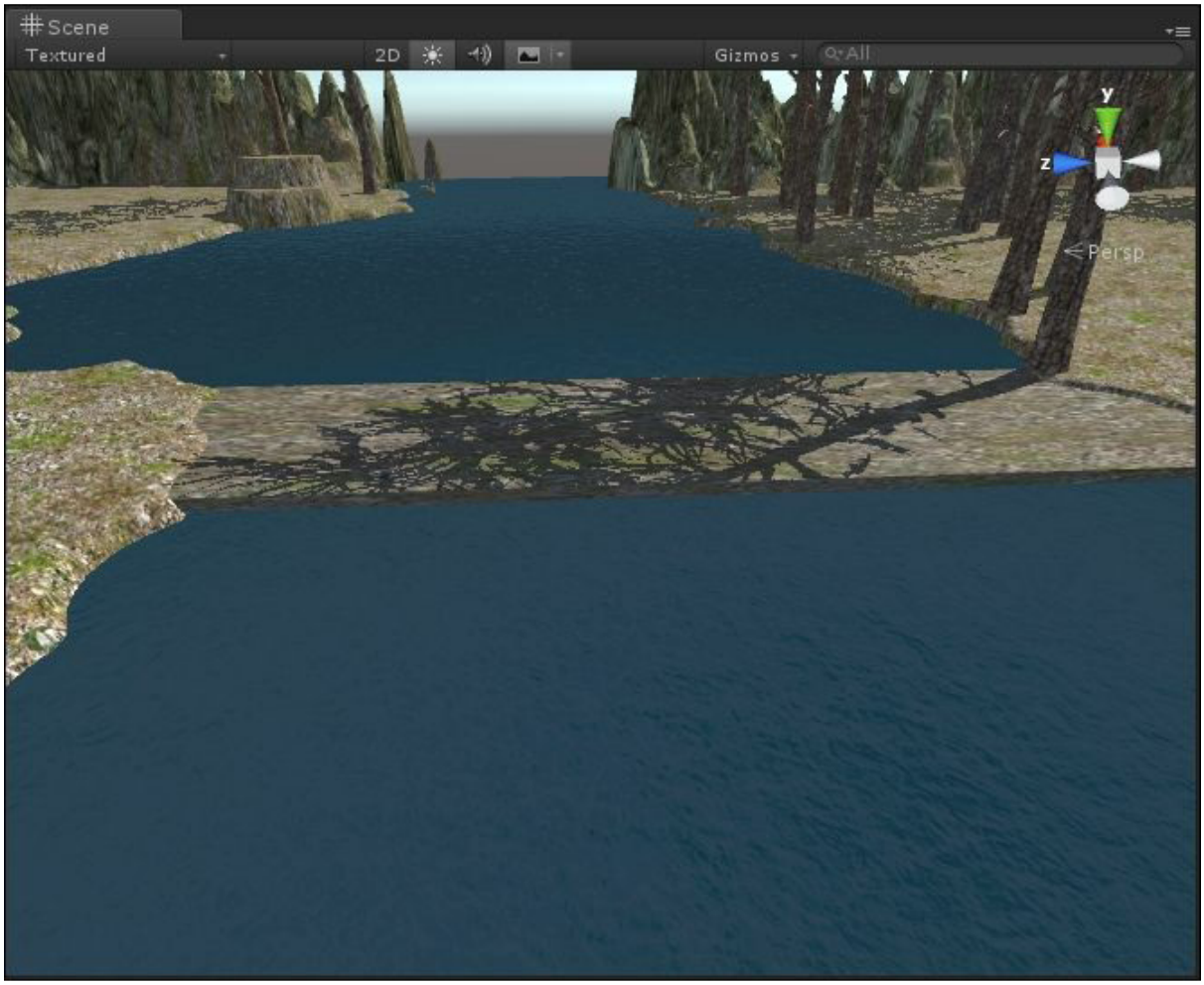
اما جائی برای Ethan در بازی ما نیست . برای اینکه به بازیکن اجازه کنترل کاراکتر کشاورز کوچک را بدهیم ، برای او به جای Ethan باید یک کنترل کننده سوم شخص ، پیاده سازی کنیم . برای انجام این کار :

۱. یونیتی را اجرا کنید .

۲. پروژه بازی را باز کنید .

۳ صحنه را بارگذاری کنید .

۴. در نمای Scene و با استفاده از ابزارهای تغییر شکل ، نمای زمین را طوری تنظیم کنید که از سطح زمین ، به پل نگاه کنیم . این به ما کمک می کند تا ببینیم که کاراکتر در کجا قرار می گیرد :



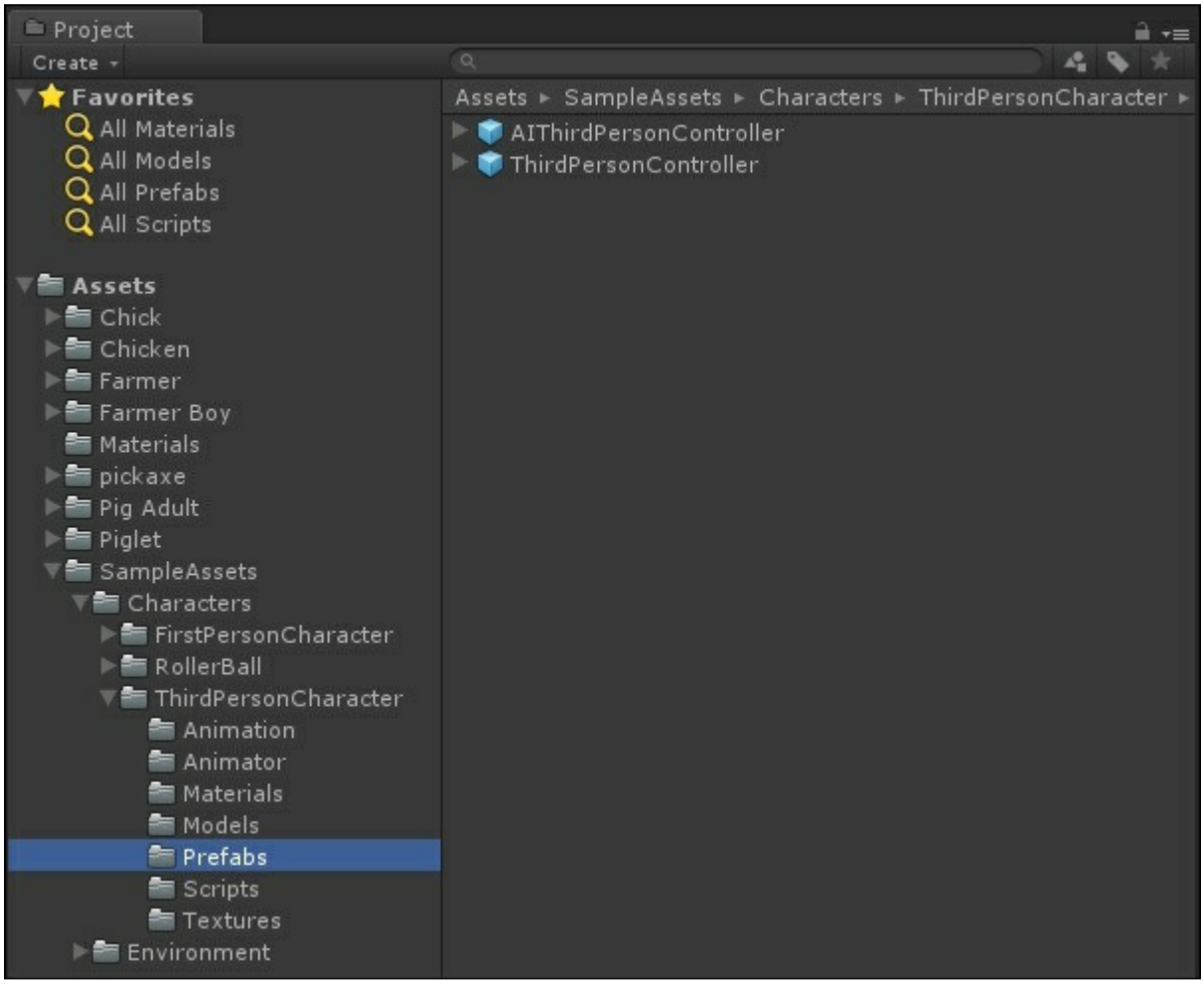
۵. در نمای Project ، پوشه SampleAssets را انتخاب کنید .

۶. منو Assets | Import Package | Characters را انتخاب کنید . بسته منبع باز شده و پنجره Importing package نمایش داده می شود . دکمه Import را کلیک کنید .

۷. در نمای Project ، پوشه ها را باز کنید تا گزینه زیر را ببینید :

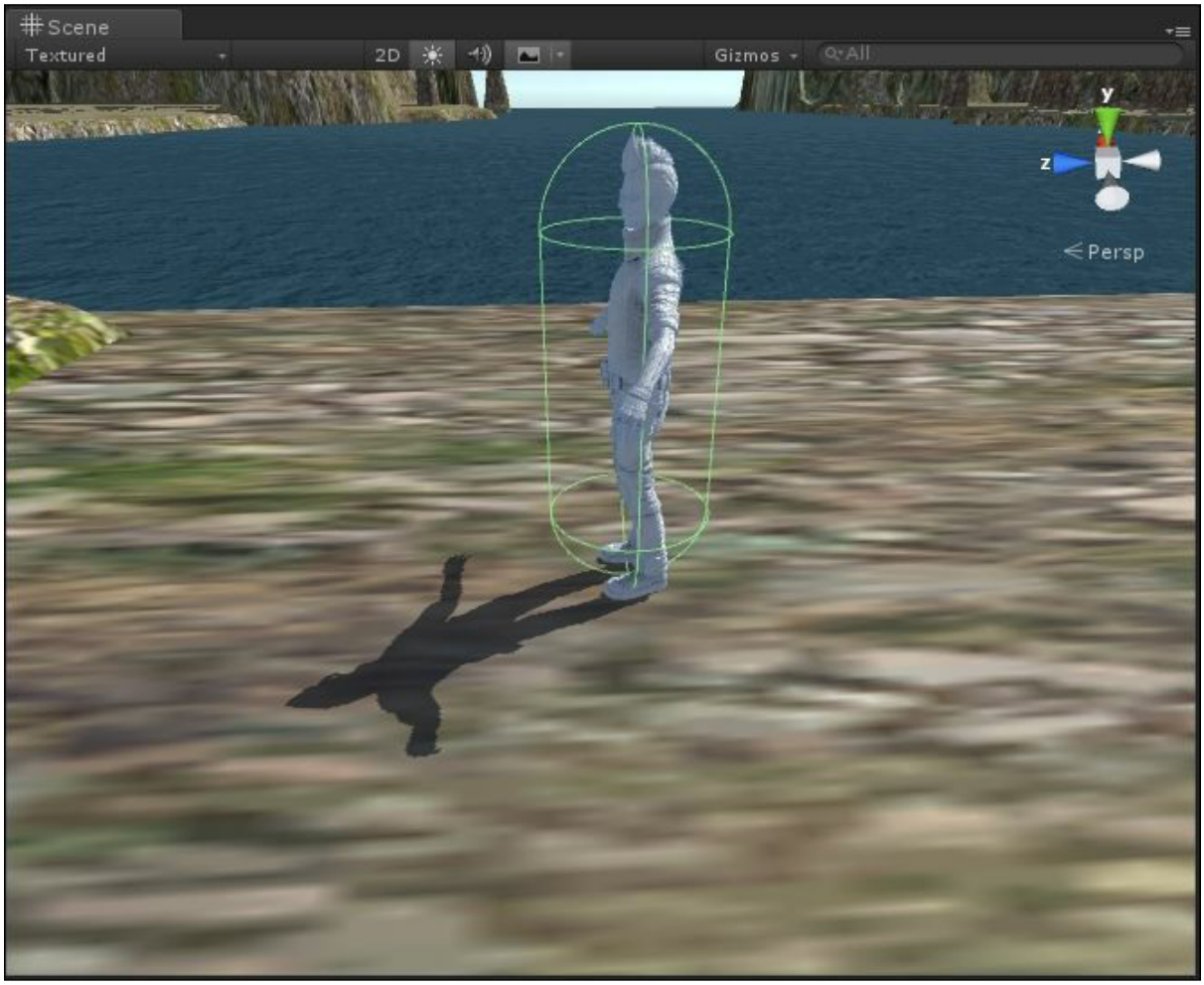
SampleAssets | Characters | ThirdPersonCharacter | Prefabs

پوشه Prefabs را کلیک کنید . در ستون دوم نمای Project ، شئ ای با نام ThirdPersonController خواهید دید :



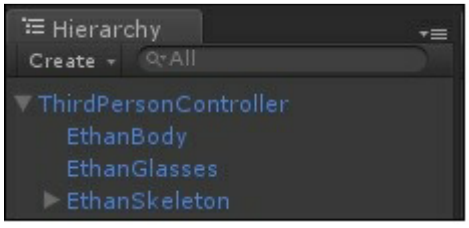
۸. شئ ThirdPersonCharacter را کلیک کرده و به داخل نمای Scene درگ کنید . شئ را روی پل رها کنید .

۹. به این شئ جدید زوم کرده و می بینید که Ethan در کنترل کننده کاراکتر ، جای گرفته است :



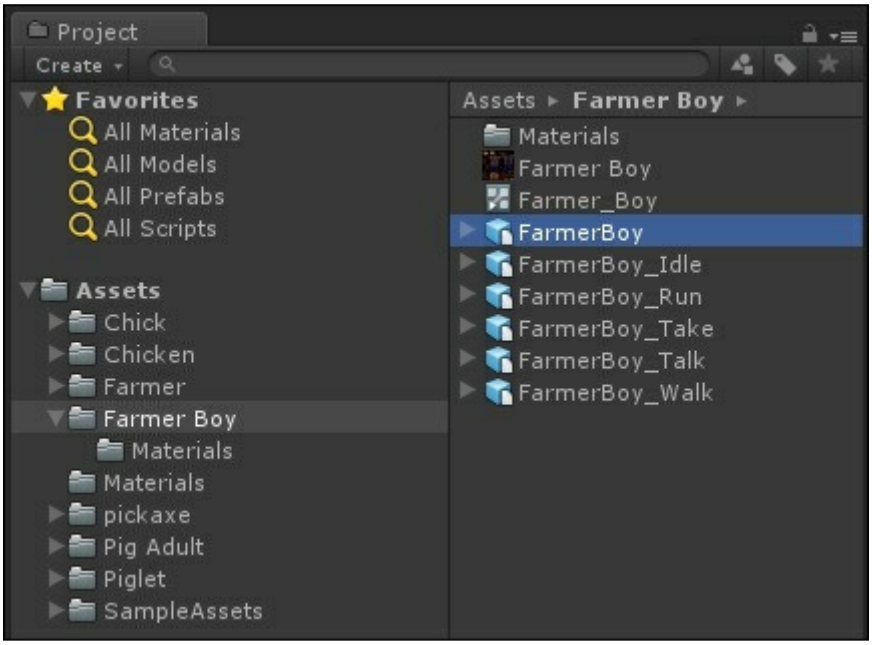
در این مرحله ، می توانید بازی را اجرا کرده و با استفاده از Ethan ، در محیط بازی گشت بزنید . سپس ، ما Colt را با Ethan ، جابه جا می کنیم . برای این کار :

۱. در نمای Hierarchy ، شئ ThirdPersonController را باز کنید . سه آیتم می بینید که با اسم Ethan شروع شده اند :



۳. سه آیتمی که با اسم Ethan شروع شده اند را حذف کنید . با این کار ، کنترل کننده سوم شخص هنوز وجود دارد اما فاقد آواتار است .

۴. در نمای Project ، گزینه Assets | Farmer Boy را انتخاب کرده و سپس در ستون دوم ، شئ FarmerBoy را انتخاب کنید :



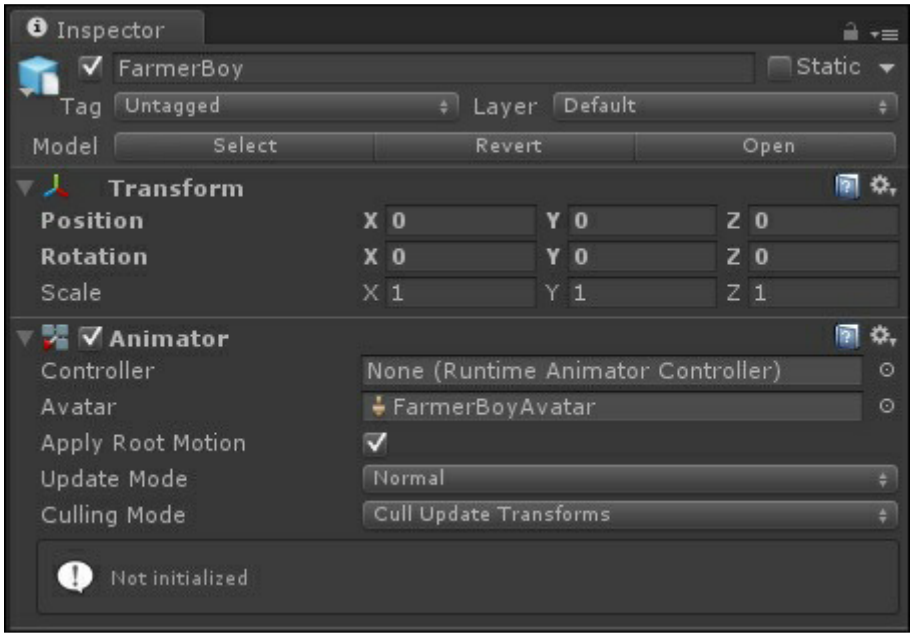
۴. شئ FarmerBoy را از نمای Project به نمای Hierarchy درگ کنید و روی شئ ThirdPersonController رها کنید :

۵. حالا می بینید که به جای Ethan ، کاراکتر Colt روی پل ایستاده است :

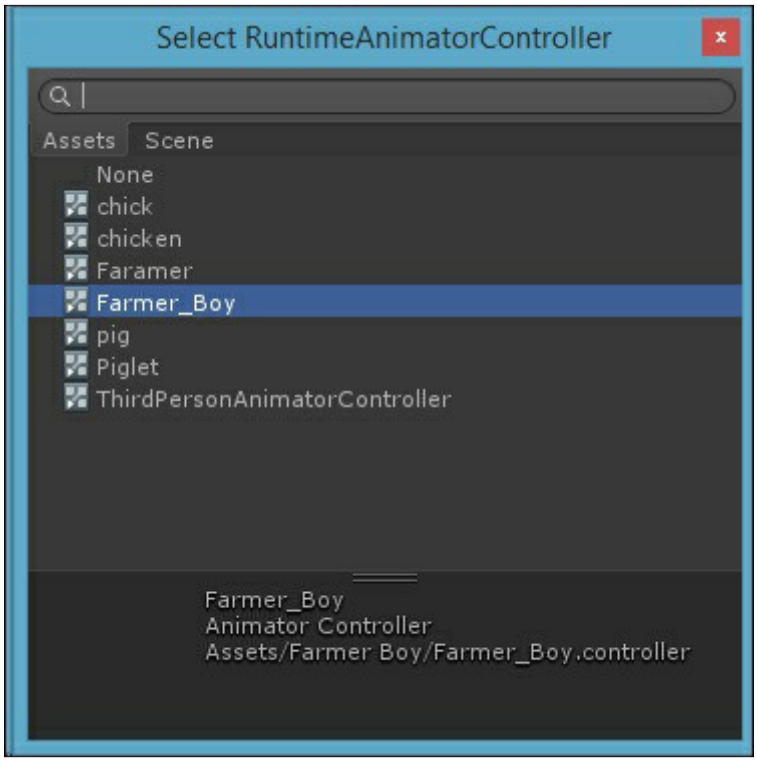
۶. صحنه و پروژه را ذخیره کنید .

۷. کار بعدی که باید انجام دهیم این است که به یونیتی بگوئیم از کدام کنترل کننده باید برای متحرک کننده (Animator) استفاده کند . در نمای Hierarchy ، گزینه ThirdPersonController | FarmerBoy را انتخاب کنید .

۸. در نمای Inspector ، آیکون دایره ای شکل کوچکی که در سمت راست قسمت Controller وجود دارد را انتخاب کنید . این آیکون در بخش Animator نمای Inspector وجود دارد :

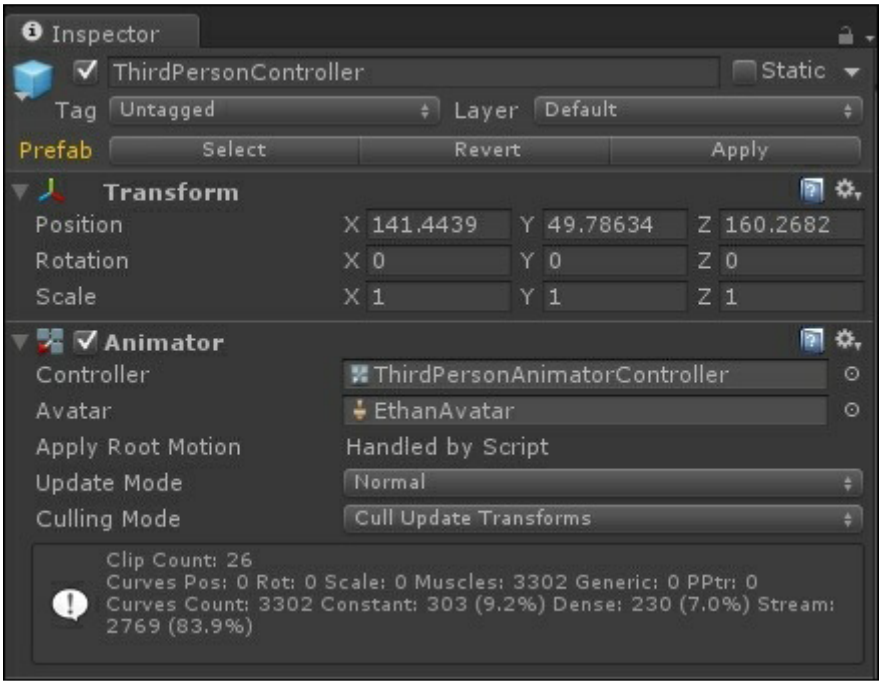


۹. پنجره Select RuntimeAnimatorController باز می شود . روی قسمت Assets کلیک کرده و سپس آیتم Farmer_Boy را دوبارکلیک کنید :

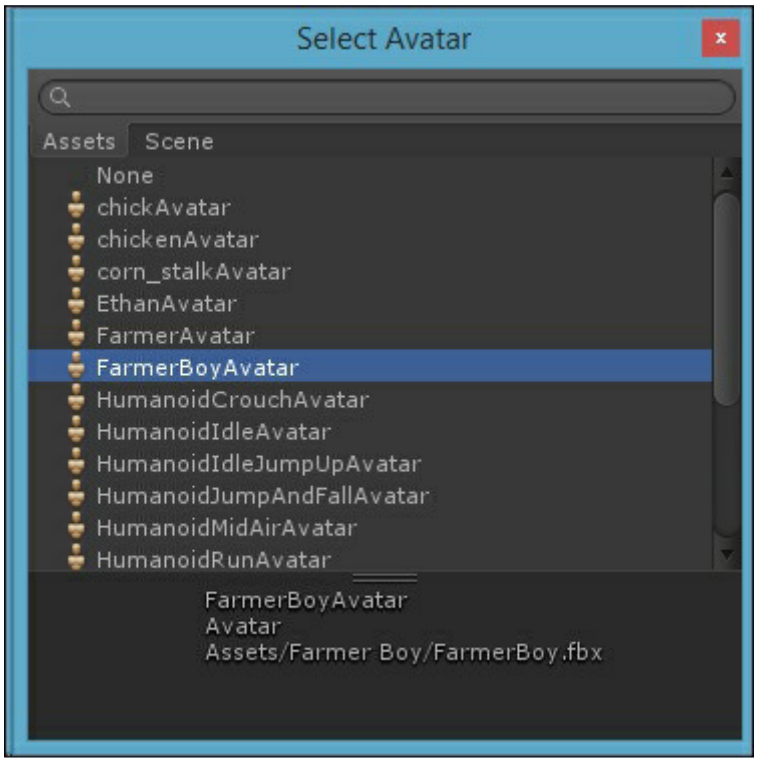


۱۰. کنترل کننده سوم شخصی که در نمای Hierarchy انتخاب شده بود را ، انتخاب کنید .

۱۱. در نمای Inspector ، می بینید که Ethan به عنوان آواتار انتخاب شده است :



۱۲. پس اگر در بخش Animator نمای Inspector ، Ethan به عنوان آواتار انتخاب شده است ، روی دایره کوچک سمت راست این فیلد کلیک کنید و از پنجره Select Avatar ، گزینه FarmerBoyAvatar را انتخاب کنید :

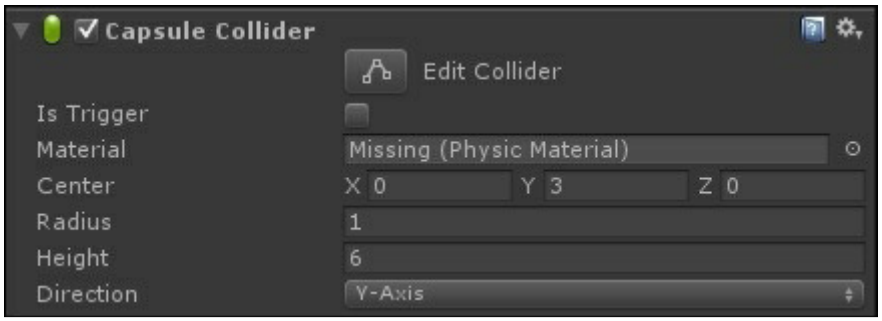


۱۳. در نمای Scene ، روی کاراکتر زوم کنید تا بتوانید کنترل کننده بازیکن را ببینید . یادآوری می کنم ، یک شکل استوانه ای سبز رنگ ، کنترل کننده بازیکن را نشان می دهد . می بینید که

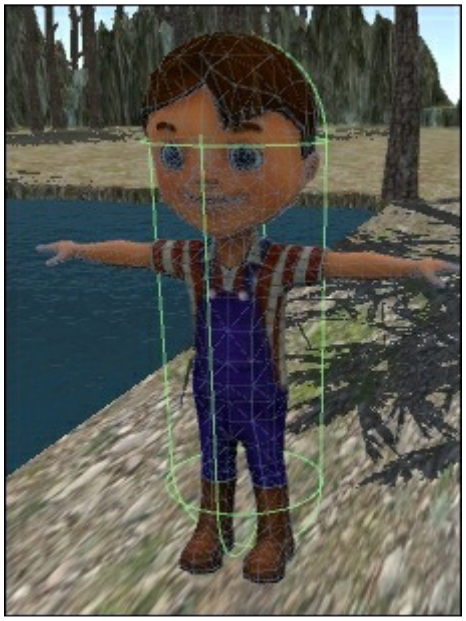
این استوانه ، کوچکتر از کاراکتر Colt است . باید کاری کنیم که کاراکتر ما به صورت کامل در داخل کنترل کننده کاراکتر جا بگیرد .

۱۴. در نمای Hierarchy ، گزینه ThirdPersonController را انتخاب کنید . با انتخاب این ، ما تغییرات را در بخش Capsule Collider نمای Inspector انجام می دهیم .

۱۵. در نمای Inspector و در بخش Capsule Collider ، مقدار Center Y را 3 قرار دهید ، مقدار Radius را 1 و مقدار Height نیز 6 شود :



۱۶. Capsule Collider حالا باید به صورت کامل کاراکتر Colt را دربرگرفته باشد . در نمای Scene ، روی Colt زوم کنید تا مطمئن شوید کپسول به اندازه کافی بزرگ و مناسب است :



۱۷. باید آخرین تغییر را نیز انجام دهیم . مختصات y کاراکتر را طوری تنظیم می کنیم که کمی بالاتر از زمین قرار گیرد . در نمای Inspector ، مقدار Position Y Transform را به 0.2 تغییر دهید . این کار از رفتن Colt به داخل زمین ، جلوگیری می کند :

حالا می توانید بازی را اجرا کرده و ببینید که کاراکتر Colt جان گرفته است . می توانید با استفاده از کلیدهای جهتی او را بچرخانید و در دنیای بازی گشت و گذار داشته باشید .

توجه
Colt نمی تواند شنا کند و روی آب راه برود . بنابراین فعلا پیشنهاد می کنم او را به داخل آب نبرید .

در بخش های بعدی ، انیمیشن ها و عملکردهای بیشتری را به Colt اضافه می کنیم .

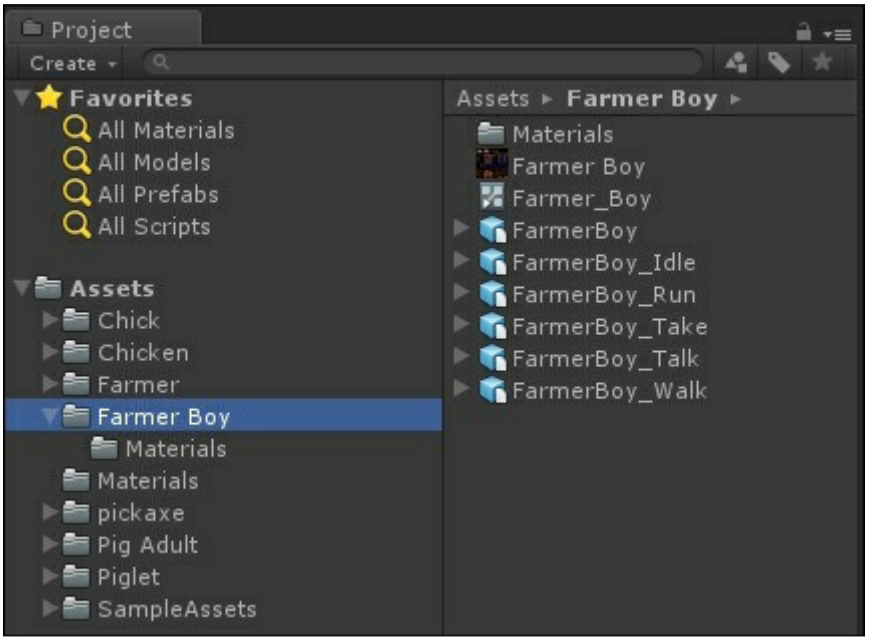
متحرک سازی کاراکترهای بازیکن

بازی ما ، مثل اغلب بازی ها ، شامل کاراکترهای انیمیت شده است . حتی اگر یک کاراکتر غیربازیکن ، فقط یک انیمیشن حالت بیکاری (idle) داشته باشد ، مهم این است که آنها متحرک سازی شده باشند . در غیر این صورت ، کاراکترها بیشتر یک مجسمه خواهند بود تا یک کاراکتر !

کاراکتر کشاورز کوچک Colt

اجازه دهید کار بر روی کاراکتر Colt را ادامه داده و مطمئن شویم که همه انیمیشن هایش به درستی کار می کند . با باز کردن پوشه آن در نمای Project ، می بینید که ۵ انیمیشن دارد . انیمیشن ها عبارتند از

idle (حالت بیکاری) ، run (دویدن) ، take (برداشتن) ، talk (صحبت کردن) و walk (راه رفتن) . این انیمیشن ها از قبل ایجاد شده اند به همین دلیل می توانیم از آنها در بازیمان استفاده کنیم :



به نظر می رسد بعضی از این انیمیشن ها به درستی کار می کنند ، علت ، کنترل کننده کاراکتری است که استفاده کرده ایم . برای مثال ، اگر بازی را اجرا کنید ، می بینید که مثلا انیمیشن walk به کلیدهای جهتی صفحه کلید یا کلیدهای WASD ، واکنش نشان می دهد . همچنین اگرچه ما هنوز انیمیشن پرش زدن را ایجاد نکرده ایم اما ، Colt می داند چطور بپرد . برای تست این کار ، کلید space را در هنگام اجرای بازی فشار دهید :

این جادو نیست بلکه مزایای استفاده از کنترل کننده کاراکتر سوم شخص موتور بازی یونیتی است . بعضی از انیمیشن ها به صورت پیش فرض ، بخشی از کاراکتر است ؛ بقیه با استفاده از ابزارهای بیرونی ایجاد می شود و بعضی نیز در داخل یونیتی ایجاد خواهند شد .

کاراکتر کشاورز مسن

با درگ شئ از نمای Project به داخل نمای Scene ، می توانیم کاراکتر کشاورز مسن را به بازی اضافه کنیم . ما هنوز آماده این کار نیستیم اما می خواهیم انیمیشن هایش را ببینیم . در نمای Project می بینید که کشاورز مسن انیمیشن های idle ، talk و walk دارد . هرچند ما از انیمیشن walk در بازیمان استفاده نمی کنیم ، اما پس از اتمام مطالعه این کتاب می توانید از آن برای توسعه و بهتر کردن بازی استفاده کنید .

حیوانات مزرعه

بازدید بقیه منابع در نمای Project نشان می دهد که کاراکترهای جوجه ، مرغ ، گراز و بچه گراز دارای یک انیمیشن idle و یک انیمیشن غذاخوردن (eat) هستند . کار دیگری برای این حیوانات جز ایستادن در یک طرف و غذاخوردن ، وجود ندارد . همانطور که بعدا خواهید دید ، ما اسکریپت هائی را به این حیوانات اضافه می کنیم تا بفهمیم ، چقدر غذا خورده اند و چقدر رشد کرده اند .

پیش نمایش انیمیشن ها

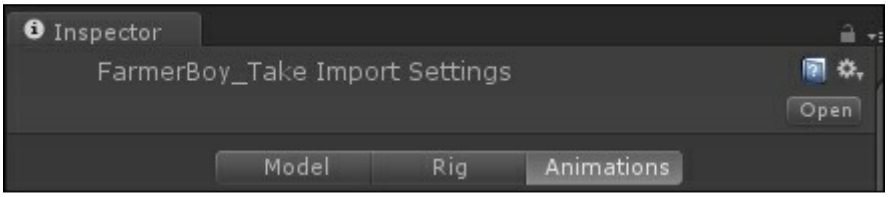
هنگام اجرای بازی ، همه انیمیشن ها دارای انیمیشن idle هستند که هنگامی که کاراکتر در حال بیکاری است ، پخش می شود . انیمیشن eat هنگامی پخش می شود که غذا برای خوردن یا آب برای نوشیدن وجود داشته باشد . انیمیشن walk کاراکتر Colt ، در پاسخ به ورودی صفحه کلید یا ماوس ، پخش خواهد شد . ما اسکریپت هائی را برای مدیریت انیمیشن take کاراکتر Colt و انیمیشن talk هردو کشاورز ، اضافه خواهیم کرد .

بدون اینکه مجبور باشیم ابتدا بازی را تکمیل کرده و اسکریپت ها را اضافه کنیم ، یونیتی قابلیت پیش نمایش انیمیشن ها را به ما می دهد . بازی ما ۱۶ انیمیشن دارد ، بنابراین قابلیت پیش نمایش آنها بدون برنامه نویسی ، زمان را خیلی ذخیره خواهد کرد .

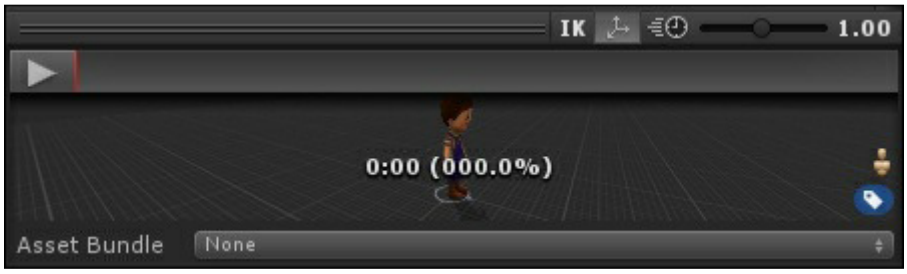
اجازه دهید انیمیشن های Colt را ببینیم . در مراحل زیر ، روی انیمیشن take متمرکز خواهیم بود :

۱. در نمای Project ، گزینه Assets | Farmer Boy | FarmerBoy_Take را انتخاب کنید .

۲. در نمای Inspector ، به شکل زیر روی قسمت Animations کلیک کنید :



۳. در نمای Inspector ، به شکل زیر ؛ پنجره ای که Colt در آن قرار دارد را درگ کنید تا به اندازه کافی بزرگ شود :



۴. حالا دکمه play را کلیک کنید تا انیمیشن پخش شود :

می توانید ۱۵ انیمیشن دیگر را نیز ببینید . در بخش بعدی ، با استفاده از عملکرد درونی ایجاد کلیپ انیمیشن یونیتی ، خودمان انیمیشن ایجاد می کنیم .

ایجاد کلیپ های انیمیشن

در یونیتی ؛ کلیپ های انیمیشن ، دنباله های کوچکی از انیمیشن هستند . آنها می توانند قطعات خیلی کوچک و مجزائی از حرکات باشند . برای مثال ؛ راه رفتن یک انیمیشن معتبر است . کلیپ های انیمیشن مربوطه ، به راست رفتن ، به چپ رفتن ، بالا نگاه کردن ، پائین نگاه کردن ، پریدن و ... است . چندین کلیپ انیمیشن ، می توانند یک انیمیشن را بسازند .

یونیتی ، دارای امکانات ایجاد کلیپ های انیمیشن سفارشی است . به منظور ایجاد یک کلیپ انیمیشن کوتاه برای کاراکتر Colt ، مراحل زیر را دنبال کنید :

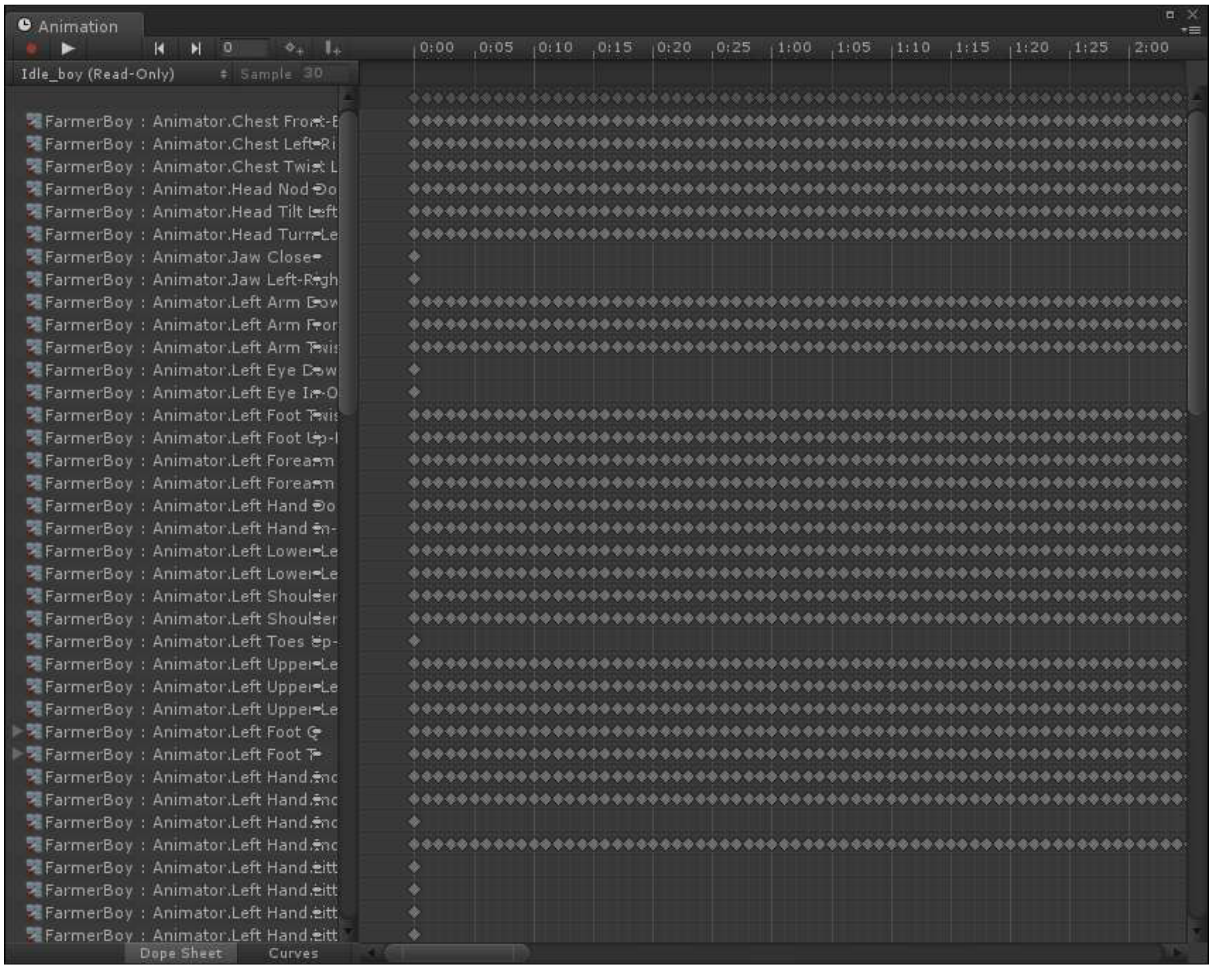
۱. یونیتی را اجرا کنید .

۲. پروژه Little Farmer Colt را باز کنید و صحنه را بارگذاری کنید .

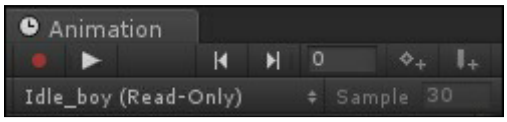
۳. در نمای Hierarchy ، شئ ThirdPersonController را انتخاب کنید .

۴. در نمای Scene ، به مقدار لازم زوم کنید تا بتوانید کاراکتر Colt را درست ببینید .

۵. منو Window | Animation را انتخاب کنید . این کار پنجره Animation را باز می کند . می بینید که کاراکتر Colt دارای تعداد زیادی از کلیپ های انیمیشن است که در این پنجره لیست شده اند :

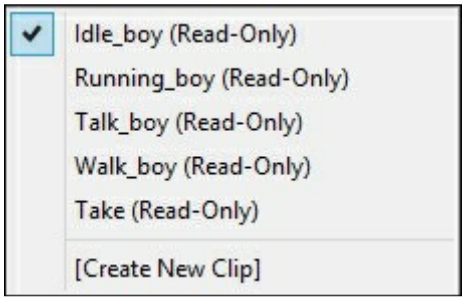


در قسمت سمت چپ پنجره Animation ، دکمه های کنترل پخش وجود دارند که درست زیر آنها ، انیمیشن Idle_boy انتخاب شده است . اگر روی دکمه play کلیک کنید ، خواهید دید که در نمای Scene ، کلیپ انیمیشن Idle_boy پخش می شود . این یک راه دیگر برای پیش نمایش کلیپ های انیمیشن است :



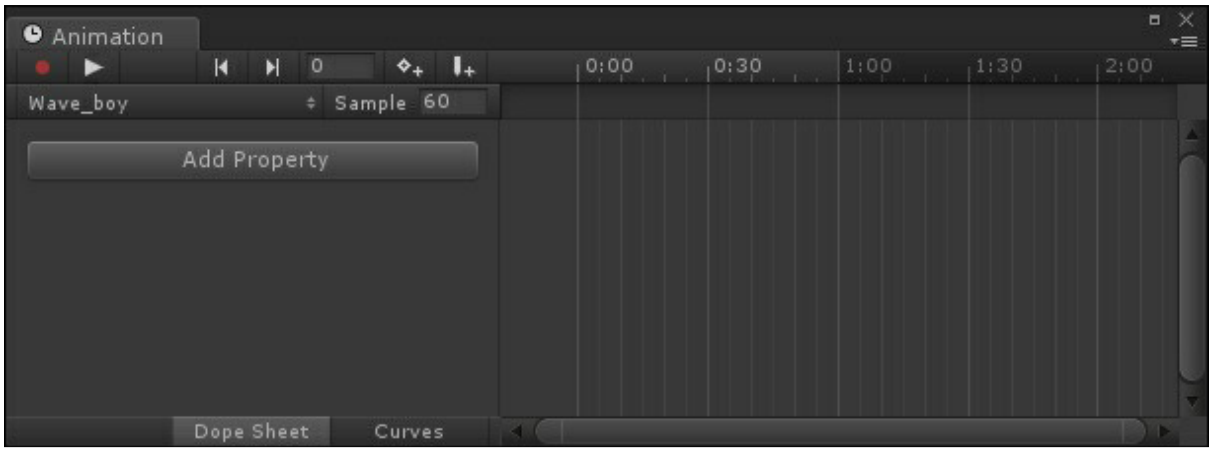
حالا مراحل زیر را دنبال کنید :

۱. در پنجره Animation ، روی دکمه up/down که در سمت راست متن Idle_boy (Read-Only) واقع شده است ، کلیک کنید . این کار ، به شکل زیر ؛ یک لیست از انیمیشن های کاراکتر فعلی را نشان می دهد :



۲. در پائین این لیست ، [Create New Clip] را انتخاب کنید . این کار پنجره Create New Animation را باز می کند . این پنجره یک نام برای انیمیشن جدید و یک مکان برای ذخیره آن ، از شما می پرسد . انیمیشن را Wave_boy بنامید ، تنظیمات پیش فرض دیگر را پذیرفته و روی دکمه Save کلیک کنید .

۳. در پنجره Animation ، حالا انیمیشن Wave_boy انتخاب شده و هنوز هیچ کلیپ انیمیشنی به آن متصل نشده است :

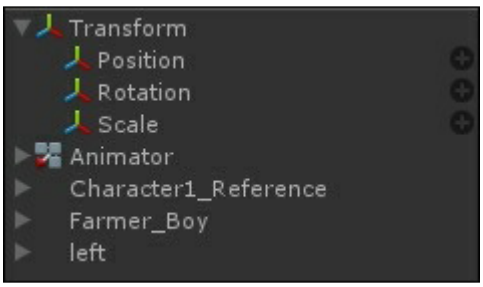


۴. اگر لازم باشد ، پنجره Animation را حرکت دهید تا مانع دیدن نمای Scene نشود .

۵. دکمه record را کلیک کنید . هنگامی که این دکمه کلیک شد ، آماده ضبط انیمیشن جدید خواهید بود . می بینید که حالا دکمه record با رنگ قرمز پررنگ شده و یک خط قرمز عمودی در قسمت timeline پنجره Animation وجود دارد .

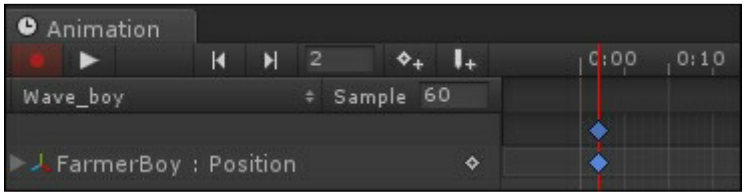
۶. دکمه Add Property را در پنجره Animation کلیک کنید .

۷. به شکل زیر ، قسمت Transform را باز کنید :

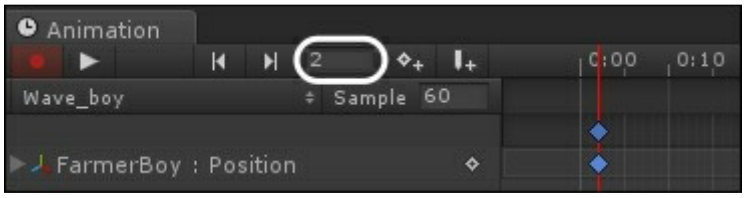


۸. علامت + سمت راست Transform | Position را کلیک کنید . این کار یک نقطه شروع به انیمیشن ما می دهد . اگر کاراکتر روی زمین نایستاده باشد ، از ابزارهای تغییر شکل استفاده کنید .

۹. حالا یک نقطه شروع داریم ، timeline را به علامت ۲ ثانیه ، جلو ببرید . برای این کار ، خط قرمز عمودی موجود در timeline را به سمت راست ، درگ کنید :



راه دیگر برای تنظیم timeline ، تایپ موقعیت دلخواه به شکل زیر است :



۱۰. در پنجره Animation ، گزینه Transform | Rotation را انتخاب کرده و آیکون + را کلیک کنید . این کار یک حرکت دیگر به انیمیشن اضافه می کند .

۱۱. در نمای Scene ، خیلی کم کاراکتر را به سمت راست بچرخانید .

۱۲. timeline را به علامت ۴ ثانیه ببرید .

۱۳. دکمه Add Property را در پنجره Animation کلیک کنید .

۱۴. Character1_Reference | Transform | Rotation را انتخاب کرده و سپس آیکون + را کلیک کنید .

۱۵. در نمای Scene ، خیلی کم کاراکتر را به سمت چپ بچرخانید .

۱۶. در پنجره Animation ، دکمه play را کلیک کنید تا بتوانید انیمیشن را تست کنید .

۱۷. دکمه record را کلیک کنید تا ضبط انیمیشن متوقف شود .

۱۸. پنجره Animation را ببندید .

۱۹. حالا می بینید که کلیپ انیمیشن در نمای Project ، لیست شده است :

خلاصه

در این فصل ، یاد گرفتید چطور با استفاده از انیمیشن ها ، زندگی را به بازی هایتان اضافه کنید . با کنترل کننده های اول و سوم شخص آشنا شدید و درخصوص دید بازیکن بازی ، آموزش دیدید . این فصل شما را در خصوص نحوه اصلاح کنترل کننده سوم شخص آموزش داد طوری که بازی بتواند کاراکتر کشاورز کوچک را نمایش دهد . شما پیش نمایشی از انیمیشن ها را مشاهده کردید و در انتها ، خودتان یک انیمیشن ساختید .

در فصل بعد ، با اسکریپت نویسی در یونیتی ، آشنا می شوید . نحوه استفاده از ابزاری به نام MonoDevelop را یاد می گیرید و با چگونگی دسترسی به اسکریپت های بازیتان ، آشنا می شوید . به ویژه ، ما بازی مان را طوری اسکریپت نویسی می کنیم که در انتهای فصل ، می توانید با ورژن اولیه آن ، بازی کنید !

فصل ۵ : اسکریپت نویسی بازی

بازی ما تقریبا شکل گرفته است . تا به حال ما ، طراحی بازی ، محیط بازی ، کاراکتر بازیکن ، کاراکترهای غیربازیکن و انیمیشن ها را تکمیل کرده ایم . چند چیز دیگر باقی مانده تا بازی کامل شود . اسکریپت نویسی یکی از آنهاست .

در این فصل برای انجام کارهای زیر اسکریپت می نویسیم :
مدیریت مقدار آب و غله ای که Colt جمع آوری می کند .
نظارت بر مقدار آب و غله ای که به هر حیوان مزرعه داده شده است .
مدیریت وقتی که یک بچه گراز و یک جوجه به گراز بالغ و مرغ تبدیل می شوند .

قبل از رفتن به سراغ اسکریپت نویسی ، مرور کلی بر روی برنامه نویسی C# خواهیم داشت . ما همچنین نحوه اسکریپت نویسی در یونیتی و نحوه استفاده از MonoDevelop را نیز بررسی می کنیم .

بعد از مطالعه این فصل :
با اصول C# آشنا می شوید .
می توانید از MonoDevelop برای ایجاد و ویرایش اسکریپت ها ، استفاده کنید .
می توانید اسکریپت ها را در پروژه یونیتی ، پیدا کنید .
با نحوه خواندن اسکریپت های یونیتی آشنا می شوید .
می توانید اسکریپت های دلخواهتان را ایجاد کنید .

توجه : در طول این فصل ، خواهید دید که اصطلاحات اسکریپت نویسی و برنامه نویسی به جای یکدیگر استفاده می شوند . در این فصل ، هردو اینها ، یک چیز هستند .

برنامه نویسی C#

C# یک زبان برنامه نویسی جامع است . آموزش نحوه استفاده از این زبان ، خارج از حد این کتاب است . اما در این قسمت ، ما اطلاعات مناسبی را فراهم کرده ایم تا بتوانید اسکریپت هائی را که برای بازی می نویسیم ، درک کنید .
C# یک زبان برنامه نویسی مبتنی بر تابع و شئ گراست . تابع ها ، قطعاتی از کدهاست که ما اسکریپت نویسی می کنیم و دارای قابلیت استفاده مجدد هستند . مثلا اگر اسکریپتی به نام `feedPig` ایجاد کنیم ،

احتمال دارد این اسکریپت مسئول مدیریت انتقال آب و غله از Colt به یک گراز باشد . چون ما چندین گراز در بازی داریم ؛ احتمالا از این تابع خیلی زیاد استفاده خواهد شد .
همانطور که گفتم ، C# یک زبان برنامه نویسی شئ گراست . زبان های شئ گرا از کلاس ها استفاده می کنند . کلاس ها مجموعه ای از داده ها و تابع ها هستند . برای مثال اگر یک کلاس به نام babyChick داشته باشیم ؛ این کلاس می تواند داده "مقدار غله مصرف شده توسط جوجه" را نگه دارد . کلاس دیگری می تواند "عمل رشد و تبدیل جوجه به مرغ" را انجام دهد .

گرامر

گرامر زبان برنامه نویسی ، شبیه به ساختار جمله و دستور زبان است . کد C# زیر را در نظر بگیرید :

```
int cornAmount = 0;
```

لیست زیر نشان می دهد ، این کد درباره چه چیزی است :
در سمت چپ علامت مساوی ؛ int cornAmount است . این متغییر جدید (cornAmount) به عنوان یک عدد صحیح اعلام شده است . بنابراین ، نحوه اعلام یک عدد صحیح ، استفاده از int و سپس اسم متغییر است .
= 0 ، یک مقدار به عدد صحیح واگذار می کند . در این مورد ؛ این مقدار صفر است .
کد با یک سمیکولون (;) پایان یافته است .

چند نکته در خصوص برنامه نویسی در C# :
فاصله ها معمولا مهم نیستند مگر اینکه بخشی از یک متن باشند . در بیشتر قسمت ها ، C# فاصله ها را نادیده می گیرد .
حرف کوچک و بزرگ ، باعث ایجاد تفاوت می شود . مثلا cornAmount و cornamount یکی نیستند .

قراردادهای نام گذاری

هنگام برنامه نویسی ، استفاده از یک قرارداد نام گذاری ، کمک کننده خواهد بود . چندین دلیل برای انجام این کار وجود دارد . اولا ، خواندن کدها را آسان تر می کند . همچنین به افرادی غیر از برنامه نویس اصلی کمک می کند ، به سرعت کدها را بفهمند . سومین مزیت استفاده از یک قرارداد نام گذاری این است که به شما کمک می کند سریع تر برنامه بنویسید .
در زیر چند مثال از قراردادهای استاندارد نام گذاری می بینید .

اسامی کلاس ها ، تابع ها و متدها

حرف اول هر کلمه از اسم کلاس ها ، تابع ها و متدها باید یک حرف بزرگ باشد . در مثال های زیر ، EatCorn ، Start و Update از این قرارداد پیروی می کنند :

```
public class EatCorn : MonoBehavior {

      void Start ( ) {
      }

      void Update ( ) {
      }
}
```

قرارداد *Camel case*

متغییرها و آرگومان ها از قراردادی معروف به camelCase استفاده می کنند . این قرارداد می گوید ، اولین کلمه دارای حرف کوچک باشد اما کلمات بعدی با حرف بزرگ شروع شوند . سه مثال زیر را ببینید :

```
pigletGrowth
coltCornHolding
chickenEatCorn
```

اختصار

هنگام برنامه نویسی باید از اختصار و کوتاه نویسی دوری کرد البته به جز چند استثنا . قبلا دیدیم که برای integer (اعداد صحیح) از int استفاده کردیم . این علاوه بر اینکه یک اختصار است ، یک کلمه کلیدی C# است . مثال های زیر زمانی را نشان می دهد که نباید از اختصار و کوتاه نویسی استفاده کرد :

مثال بد : p1CrnCon

مثال خوب : piglet1CornConsumed

مثال بد : cknay

مثال خوب : chickenArray

مثال بد : plyH20Hld

مثال خوب : playerWaterHeld

هدف این قرارداد ، خواناتر شدن اسکریپت هاست .

کاراکترهای خاص

باید از استفاده کاراکترهای خاص مثل خط تیره (-) و underscore (_) دوری کرد . اینها تعداد خطاهای ممکن در زمان اسکریپت نویسی را افزایش می دهند . بنابراین به جای chicken_Array از chickenArray استفاده کنید .

انواع داده

در C# بیش از ۱۲ نوع داده استفاده می شود و حتی خودتان نیز می توانید یک نوع داده مخصوص ایجاد کنید . لیست زیر ، انواع داده مورد استفاده در بازی ما خواهد بود :

نوع داده : بولی (bool)
توضیح : True یا False (صحیح یا غلط)

نوع داده : عدد صحیح (int)
توضیح : همه اعداد بین 2,147,483,648- تا 2,147,483,647

نوع داده : رشته متنی (string)
توضیح : متن هائی مثل "Hello Farmer Colt"

نوع داده : آرایه
توضیح : آرایه ها را می توانید به شکل داده های کنار هم در حافظه ، تصور کنید .

استفاده از MonoDevelop

هنگام اسکریپت نویسی ، باید از یک ویرایشگر استفاده کنیم . برنامه نویسان معمولا از یک ویرایشگر مخصوص برای اسکریپت نویسی استفاده می کنند . خبر خوب این است که یونیتی دارای MonoDevelop است . MonoDevelop ویرایشگری است که می تواند همه زبان های برنامه نویسی

مورد استفاده در یونیتی (C# ، جاوااسکریپت و Boo) را مدیریت کند . در بخش بعدی با MonoDevelop آشنا می شوید .

اسکریپت نویسی در یونیتی با استفاده از C#

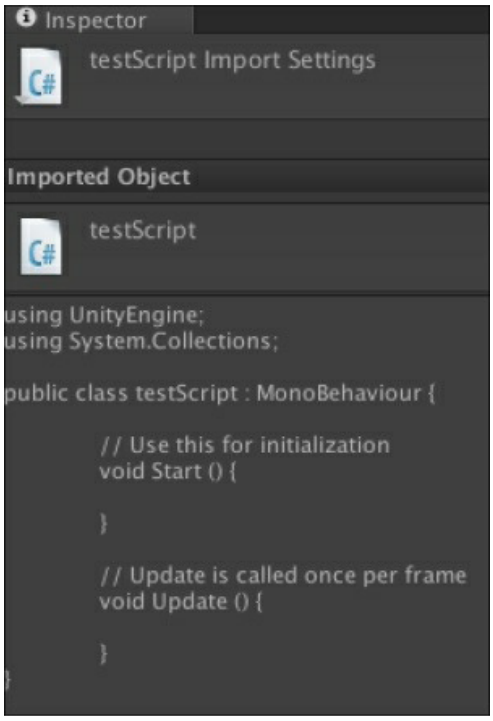
برای ایجاد یک اسکریپت جدید در یونیتی با استفاده از C# ، مراحل زیر را دنبال کنید :

۱. یونیتی را اجرا کنید .

۲. پروژه بازی را باز کنید .

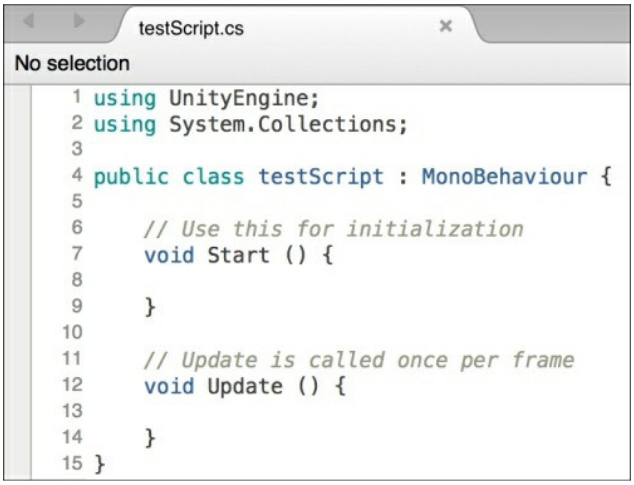
۳. به منو Assets | Create | C# Script بروید .

۴. اسکریپت را در نمای Project خواهید دید . آن را testScript بنامید . بعد از پایان کار ، این اسکریپت را حذف خواهیم کرد :



حالا در نمای Inspector ، اسکریپت قابل دیدن خواهد بود .

۵. در نمای Project و زیر Assets ، روی اسکریپت دوبار کلیک کنید . این کار باعث اجرای MonoDevelop شده و قادر به ویرایش اسکریپت خواهید بود :



۶. در MonoDevelop ، کد زیر را به خط 8 اضافه کنید :

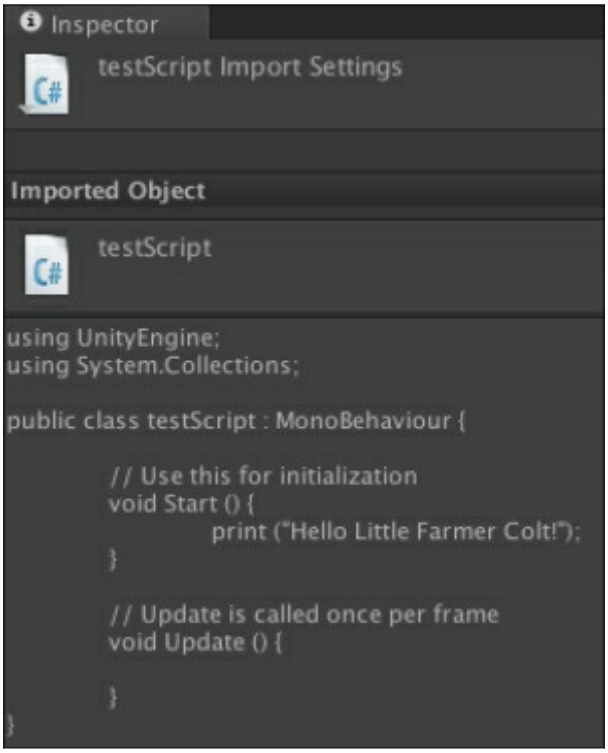
```
print("Hello Little Farmer Colt!");
```

۷. اسکریپت شما به صورت زیر می شود :

۸. در MonoDevelop به منو File | Save یا File | Save All بروید .

۹. پنجره MonoDevelop را ببندید .

۱۰. در نمای Inspector ، اسکریپت را بازدید کنید تا مطمئن شوید که تغییرات صورت گرفته ، ذخیره شده باشد :



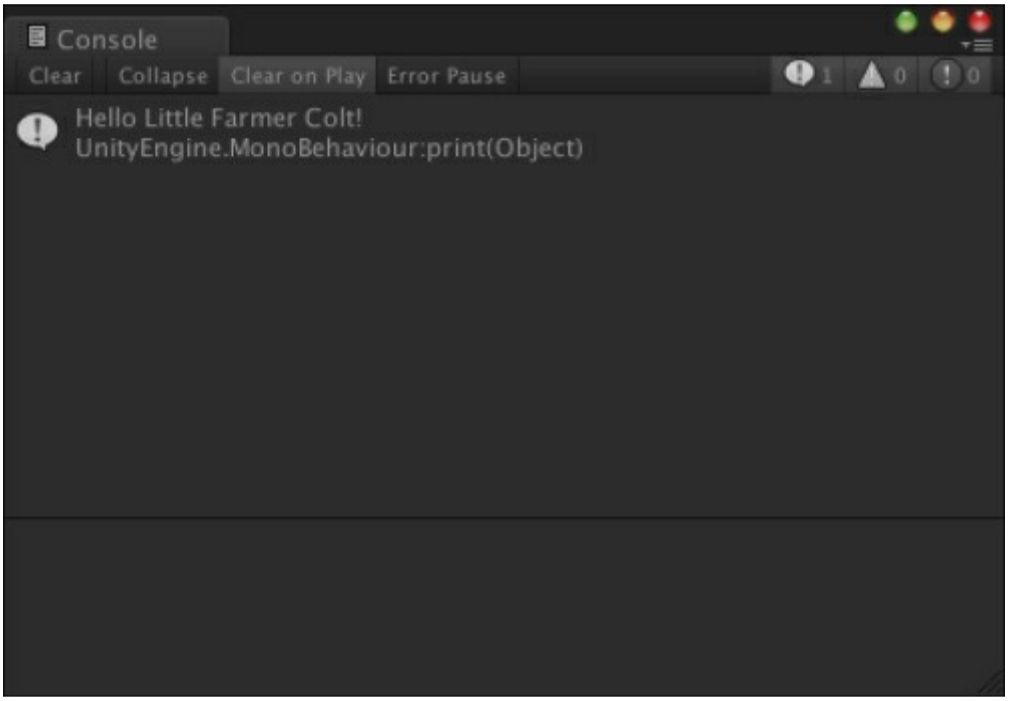
۱۱. در نمای Hierarchy ، روی گزینه Main Camera کلیک کنید .

۱۲. در نمای Project ، اسکریپت را کلیک کرده و به گزینه Main Camera که در نمای Hierarchy موجود است ، درگ کنید . این کار ، اسکریپت را به گزینه Main Camera متصل می کند . اسکریپت ها برای اینکه در بازی مورد استفاده قرار گیرند ، باید به یک شئ بازی متصل شوند .

۱۳. با انتخاب گزینه Main Camera در نمای Hierarchy ، می بینید که اسکریپت در نمای Inspector لیست می شود :

۱۴. حالا روی دکمه play کلیک کنید . اسکریپت اجرا شده و پیغام خروجی روی خط فرمان یونیتی ، نوشته می شود . این خط فرمان در گوشه پائینی سمت چپ محیط کاری یونیتی واقع شده است :

در ضمن با رفتن به منو Window | Console نیز می توانید به خط فرمان خروجی دسترسی پیدا کنید . این کار به شکل زیر ، خط فرمان را در یک پنجره جداگانه باز می کند :



مراحل حذف این اسکریپت :

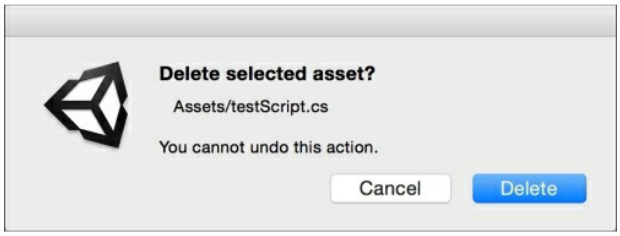
۱. در نمای Hierarchy ، شئ Main Camera را انتخاب کنید .

۲. در نمای Inspector ، روی چرخ دنده سمت راست اسکریپت کلیک کنید .

۳. Remove Component را انتخاب کنید .

۴. در نمای Project ، روی اسکریپت راست کلیک کرده و Delete را انتخاب کنید .

۵. به شکل زیر روی دکمه Delete کلیک کنید :



تابع های Start و update

احتمالا در هنگام ایجاد اولین اسکریپت C# ، متوجه شدید که اسکریپت دو تابع دارد :

```
void Start () { }
void Update () { }
```

ما کدمان را در تابع Start قرار دادیم تا اولین باری که شئ متصل به آن ، نمونه سازی می شود ، این کد اجرا شود . هنگامی که بازی شروع می شود ، شئ Main Camera نمونه سازی می شود .

اگر دستوراتمان را در تابع Update قرار دهیم ، در هر فریم ، اجرا خواهند شد . بازی ما با سرعت چندین فریم در هر ثانیه اجرا می شود بنابراین قراردادن دستور print در این تابع باعث ایجاد تعداد زیادی از خروجی ها و پردازش های غیرضروری خواهد شد .

مثالی از اسکریپت نویسی

اجازه دهید اسکریپتی طولانی تر را ببینیم . اسکریپتی که در کدهای زیر می بینید ، به منظور گرفتن ورودی کاربر از طریق دکمه های صفحه کلید ، استفاده می شود . اشکالی ندارد ، اگر این اسکریپت را نمی فهمید . ما فقط آن را برای آشنا شدن شما با اسکریپت های یونیتی ، در اینجا آورده ایم . لازم نیست آن را تایپ کنید یا به بازیتان اضافه نمائید :

```
using UnityEngine;
using System.Collections;

public class ButtonInput : MonoBehaviour
{
	public GUITexture graphic;
	public Texture2D standard;
	public Texture2D downgfx;
	public Texture2D upgfx;
	public Texture2D heldgfx;

	void Start()
	{
		graphic.texture = standard;
	}

	void Update ()
	{
		bool down = Input.GetButtonDown("Jump");
		bool held = Input.GetButton("Jump");
		bool up = Input.GetButtonUp("Jump");

		if(down)
		{
			graphic.texture = downgfx;
		}
		else if(held)
		{
			graphic.texture = heldgfx;
		}
		else if(up)
		{
			graphic.texture = upgfx;
		}
		else
		{
			graphic.texture = standard;
```

```
            }

            guiText.text = " " + down + "\n " + held + "\n " + up;
        }
}
```

این اسکریپت شامل کلاس ButtonInput است که دارای متدهای Start و Update است . تابع Start ، یک بار هنگامی که بازی در ابتدا شروع می شود ؛ اجرا می شود . متد Update در حالیکه بازی در حال اجراست ، در هر فریم از بازی اجرا می شود .

منابع اسکریپت در یونیتی

قبلا با فروشگاه منابع یونیتی آشنا شدیم و حتما می دانید که اسکریپت ها نیز یکی از منابع موجود در این فروشگاه است . هزاران اسکریپت رایگان و پولی در فروشگاه در دسترس شماست .
تصویر زیر ، نمونه ای از یک اسکریپت رایگان است که شرکت Unity Technologies سازنده آن است :

این منبع شامل شانزده lens flare است که می توانید از آنها در بازی هایتان استفاده کنید . چون این منبع رایگان است ، باید دکمه Open in Unity را کلیک کنید .
با استفاده از اسکریپت های این چنینی که از قبل آماده هستند ، می توانید زمان زیادی را صرفه جوئی کنید .
حالا که به صورت مقدماتی با زبان C# و نحوه استفاده از اسکریپت ها در یونیتی ، آشنا شدید ؛ وقت آن است که اینها را در بازیمان به کار گیریم . این کار را در بخش بعد انجام خواهیم داد .

اسکریپت نویسی بازی

قبل از اسکریپت نویسی بازیمان ، باید کمی نقشه بکشیم . باید ببینیم برای چه چیزهائی اسکریپت می نویسیم و چه رویدادهائی باید این اسکریپت ها را راه اندازی کنند . همچنین باید تعیین کنیم که چه نوعی از داده ها را نیاز داریم . چون بازی ما خیلی ساده است ، انجام این امور ، زیاد مشکل نخواهد بود .

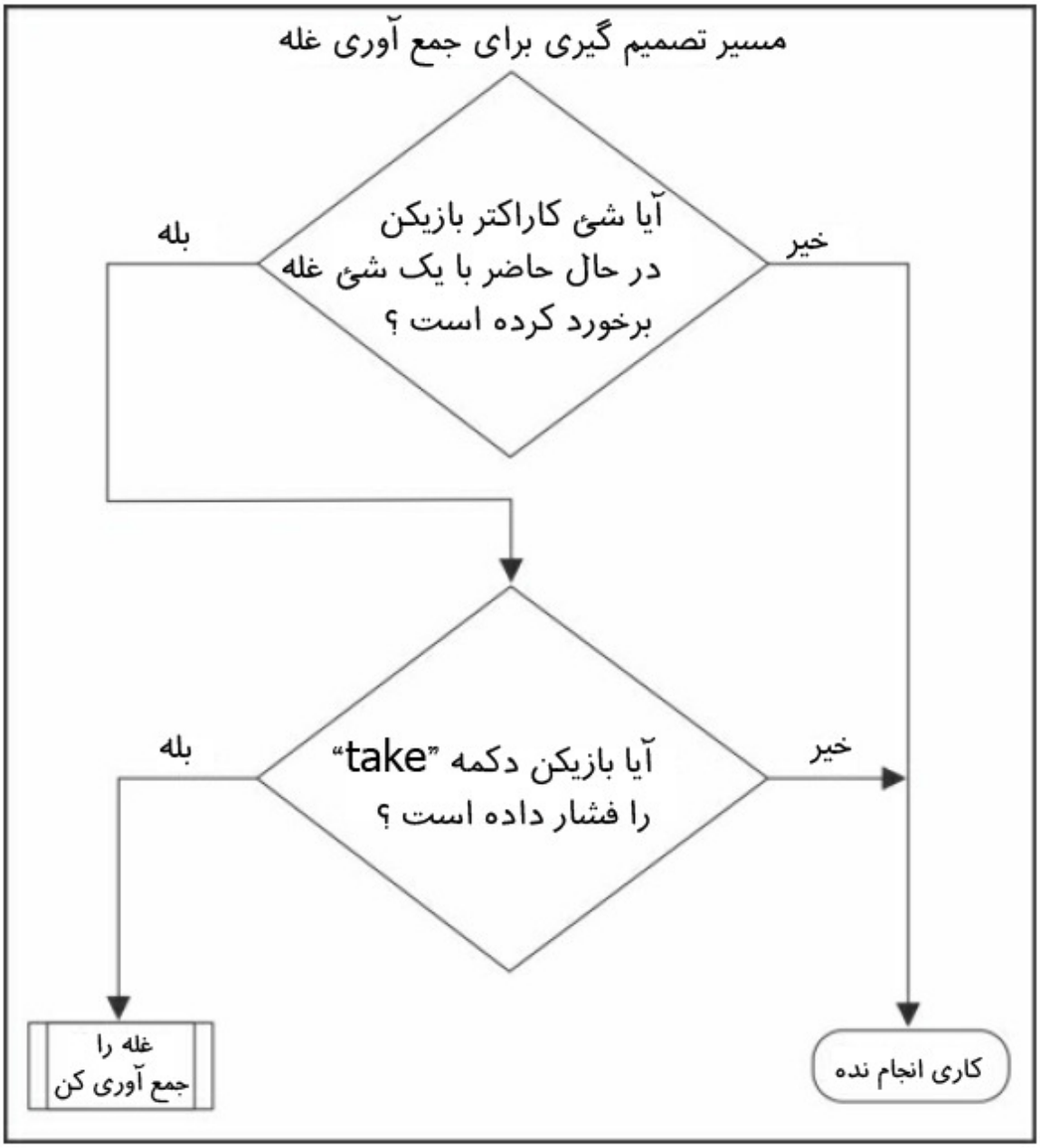
طرح ریزی اسکریپت ها

اجازه دهید با مرور طراحی بازیمان ، تعیین کنیم که چه چیزهائی باید اسکریپت نویسی شوند . می دانیم که عمده ترین کاری که کاربر انجام می دهد ، هدایت و حرکت دادن کاراکتر بازیکن در محیط بازی است . همانطور که قبلا دیدید ، این کار توسط یونیتی برای ما انجام می شود و نیازی به برنامه نویسی ندارد .

ما باید برای رویدادهای جمع آوری آب و غله توسط بازیکن و تغذیه حیوانات ، طرح ریزی کنیم .

جمع آوری غله

ما بازی را طوری طراحی کرده ایم که کاراکتر بازیکن بتواند غله را جمع آوری کرده و حیوانات مزرعه را با آن تغذیه نماید . شیوه انجام این کار این است که تشخیص دهیم کی کاراکتر بازیکن و اشیاء غله با هم برخورد می کنند . هنگامی که این برخورد اتفاق افتاد ، اگر بازیکن از دستور take استفاده کند , غله جمع آوری خواهد شد . نمودار زیر این کار را بهتر نشان می دهد :



پس سه اسکریپت نویسی باید انجام دهیم :

۱. اضافه کردن یک collider (برخوردکننده) برای اینکه بفهمیم کی کنترل کننده بازیکن و اشیاء غله باهم برخورد کرده اند .

۲. دادن توانائی فراخواندن دستور take به بازیکن . ما این را برای کلید T صفحه کلید ، برنامه نویسی می کنیم . اگر در حال ساخت این بازی برای دستگاه های موبایل هستید ، باید به دنبال راه حل دیگری غیر از صفحه کلید بگردید !

۳. تعیین اینکه آیا در هنگامی که برخورد اتفاق افتاده است ، بازیکن از دستور take استفاده کرده یا نه ؟

۴. دنبال کردن اینکه بازیکن چقدر غله دارد .

جمع آوری آب

جمع آوری آب نیز شبیه به جمع آوری غله است . ما از یک listener برای تشخیص برخورد و تعیین اینکه آیا بازیکن دستور take را به کارگیری کرده است یا خیر ، استفاده می کنیم . سپس باید مقدار آبی را که بازیکن دارد ، ردگیری و دنبال کنیم .

برای اسکریپت نویسی جمع آوری آب :

۱. اضافه کردن collider برای تشخیص برخورد

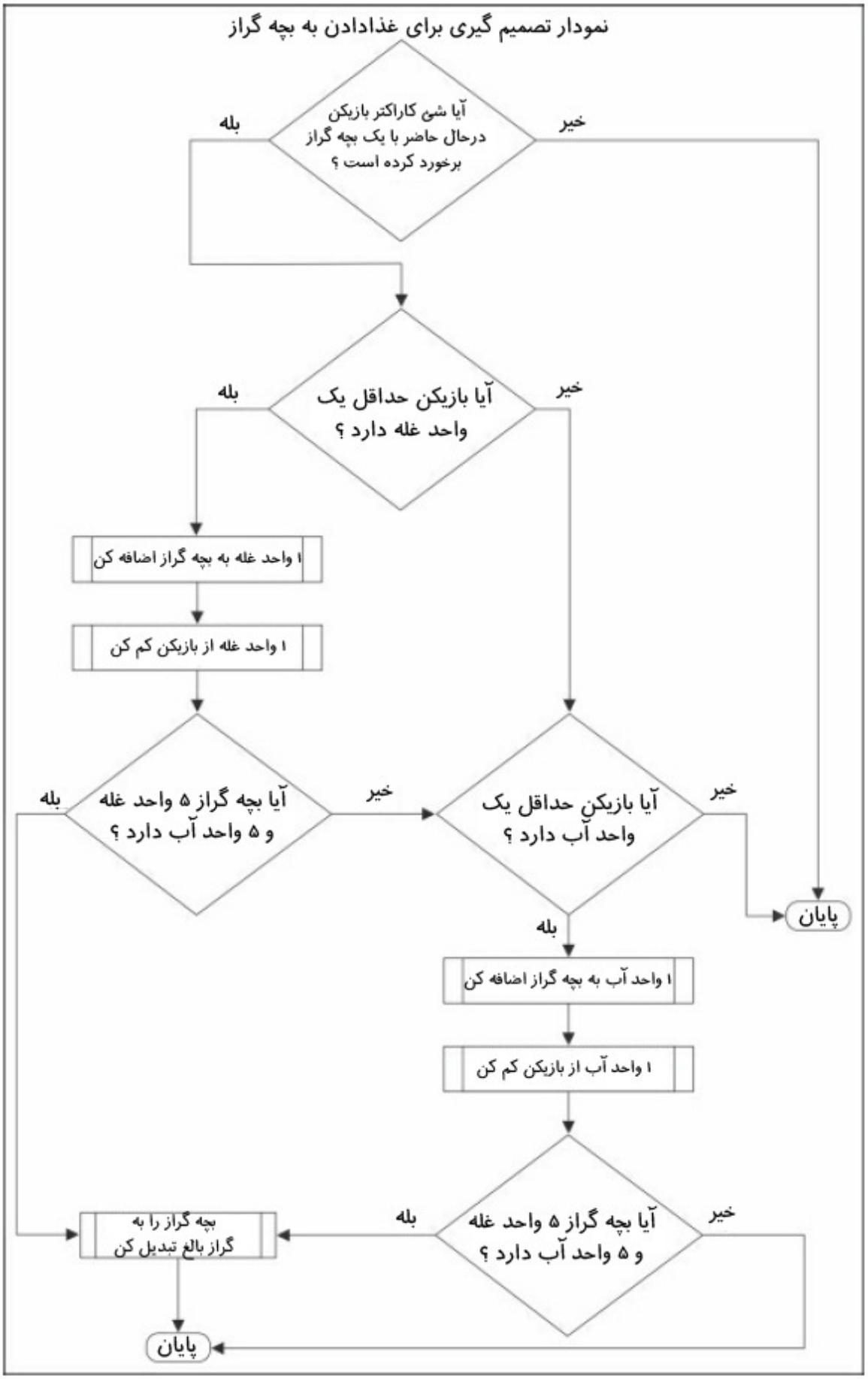
۲. تعیین اینکه آیا بازیکن درهنگام برخورد ، از دستور take استفاده کرده است یا نه .

۳. دنبال کردن اینکه بازیکن چقدر آب دارد .

غذادادن به بچه گرازها

بازی با ۱۰ بچه گراز در مزرعه کشاورز شروع می شود . برای اینکه اینها رشد کنند ، به آب و غله نیاز دارند . بعد از اینکه بچه گرازها ۵ واحد آب و ۵ واحد غله کسب کردند ، به گراز بالغ تبدیل می شوند .

نمودار زیر فرایند انجام این کار را نشان می دهد :


نمودار تصمیم گیری برای غذادادن به بچه گراز
آیا شئ کاراکتر بازیکن درحال حاضر با یک بچه گراز برخورد کرده است ؟
بله
خیر
آیا بازیکن حداقل یک واحد غله دارد ؟
بله
خیر
۱ واحد غله به بچه گراز اضافه کن
۱ واحد غله از بازیکن کم کن
آیا بچه گراز ۵ واحد غله و ۵ واحد آب دارد ؟
بله
خیر
آیا بازیکن حداقل یک واحد آب دارد ؟
خیر
پایان
بله
۱ واحد آب به بچه گراز اضافه کن
۱ واحد آب از بازیکن کم کن
آیا بچه گراز ۵ واحد غله و ۵ واحد آب دارد ؟
بله
خیر
بچه گراز را به گراز بالغ تبدیل کن
پایان

غذا دادن به جوجه ها

غذا دادن به جوجه ها دقیقا مثل بچه گرازها انجام می شود . جوجه ها بعد از به دست آوردن ۵ واحد آب و غله ، تبدیل به مرغ بالغ می شوند .

برای هردو گروه از حیوانات (مرغ و گراز) ؛ به یک شیوه تبدیل از بچه به بالغ را انجام می دهیم . هنگامی که حیوان ۵ واحد آب و غله به دست آورد ؛ یک گراز بالغ را در همان مختصات X ، Y و Z بچه گراز نمایش می دهیم . سپس بچه گراز را مخفی می کنیم . همین شیوه برای جوجه ها و مرغ ها نیز انجام می شود .

غذا دادن گراز های بالغ

هنگامی که یک بچه گراز به گراز بالغ تبدیل شد ، گراز دارای صفر واحد آب و غله خواهد بود . گراز به ۱۰ واحد آب و ۱۰ واحد غله نیاز دارد تا کاملا رشد کند . گرازها می توانند بیش از ۱۰ واحد آب و غله مصرف کنند بنابراین ما آنها را محدود نخواهیم کرد .

به دست آوردن تعداد مشخصی از گرازها و مرغ های رشد یافته ، یکی از ماموریت های بازیکن خواهد بود .

برای اسکریپت نویسی غذادادن به گرازها ، آنچه را که برای بچه گرازها انجام دادیم ، دنبال می کنیم .

غذا دادن به مرغ های بالغ

تغذیه مرغ ها نیز به همان شیوه گرازهای بالغ کار می کند . مرغ ها نیز بعد از به دست آوردن ۱۰ واحد آب و غله کاملا رشد می کنند . بازهم محدودیتی در زمینه مصرف آب و غله برای مرغ ها نیز وجود ندارد .

داده های لازم

مهم است که قبل از شروع برنامه نویسی ، درباره داده هائی که بازی ما نیاز دارد ، فکر کنیم . ما از یک سطح بالا درباره این داده ها تصور خواهیم کرد . مثلا می دانیم که باید اطلاعاتی را درباره بچه گرازها ،

گرازهای بالغ ، جوجه ها و مرغ های بالغ نگهداری کنیم . همچنین باید بدانیم که اینها چقدر آب و غله مصرف می کنند و اینکه کشاورز جوان چقدر آب و غله جمع آوری کرده و مجموع ذخیره او چقدر است .

ما نگاهی دقیق تر به داده های مورد نیازمان خواهیم داشت و تصمیم می گیریم که متغییرهایمان را چه بنامیم .

درباره گرازها

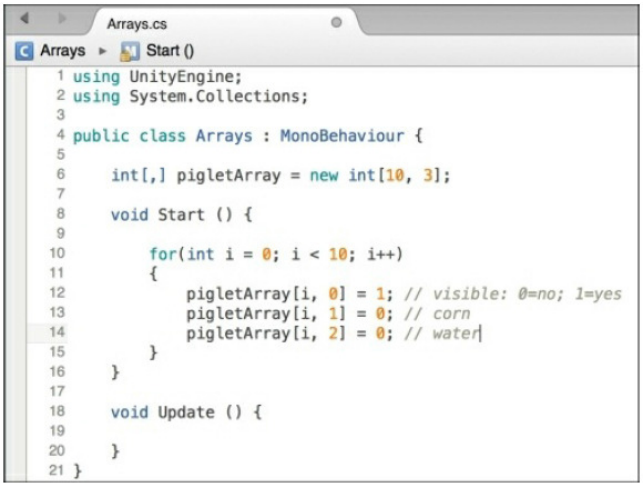
در بازی ما ۱۰ بچه گراز و ۱۰ گراز بالغ وجود دارد . برای هر یک از این حیوانات ، باید مقدار واحدهائی از آب و غله هر حیوان را دنبال کنیم . جدول زیر محدوده هر یک از آیتم های داده ای را نشان می دهد :

Animal	Visible	Corn	Water
piglet1	Y or N	0, 1, 2, 3 . . .	0, 1, 2, 3 . . .
pig1	Y or N	0, 1, 2, 3 . . .	0, 1, 2, 3 . . .

آیتم visible (قابل دیدن) از نوع بولی و corn (غله) و water (آب) از نوع عدد صحیح خواهند بود . ما از دو آرایه استفاده می کنیم . یک آرایه مخصوص بچه گرازها خواهد بود و دیگری مخصوص گرازهای بالغ . چون آرایه ها از یک نوع داده استفاده می کنند ، ما به جای نوع داده بولی از 0 و 1 برای نمایش مخفی بودن و آشکار بودن استفاده می کنیم . ستون اول ، شماره حیوان است بنابراین مقادیر آن 0 ، 1 ، 2 ، 3 ، 4 ، 5 ، 6 ، 7 ، 8 و 9 خواهد بود .

ستون دوم نشان می دهد که آیا بچه گراز قابل دیدن است یا خیر . 0 نشان می دهد که بچه گراز قابل دیدن نیست و 1 نشان می دهد که بچه گراز قابل دیدن است . ستون های سوم و چهارم به ترتیب تعداد واحدهای غله و آب را نشان می دهند .

اسکریپت ایجاد آرایه بچه گراز را ببینید :



خط 6 نحوه ایجاد یک آرایه با ۱۰ سطر و ۳ ستون را نشان می دهد . یادآوری می کنم که آرایه ها از صفر شروع می شوند بنابراین یک آرایه ۱۰ در ۳ به شکل زیر خواهد بود :

	0	1	2
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

از خط ۱۰ تا ۱۵ ، از یک حلقه for برای مقداردهی اولیه آرایه استفاده کرده ایم . ما با صفر ، حرکت از میان سطرهای آرایه را شروع و با ۹ پایان می دهیم . خطهای ۱۲ ، ۱۳ و ۱۴ مقادیر را داخل آرایه قرار می دهند .در اولین ستون مقدار 1 قرار داده می شود تا نشان دهد که بچه گراز ، قابل دیدن است . دو ستون بعدی نشان می دهند که صفر واحد غله و آب وجود دارد .

برای دیدن این موضوع در عمل ، کد زیر را بین خط ۱۴ و ۱۵ قرار دهید :

```
Debug.Log ("Piglet " + i + "\t" + pigletArray[i,0] + "\t" + pigletArray[i, 1] +
"\t" + pigletArray[i, 2]);
```

هنگام ذخیره اسکریپت ، باید آن را به داخل گزینه Main Camera درگ کنید . حالا اجازه دهید بازی را اجرا و خروجی را ببینیم . می بینید که ۱۰ بچه گراز وجود دارد که هر کدام در ستون اول یک "1" دارند و دو ستون بعدی آنها صفر است :

حالا آرایه گرازهای بالغ را هم اضافه می کنیم . برای این کار ، آرایه بچه گرازها را کپی می کنیم .
pigletArray را به pigArray تغییر می دهیم و چون در ابتدای بازی ، گرازهای بالغ قابل دیدن نیستند ؛ ستون دوم را صفر قرار می دهیم :

Arrays.cs

Arrays ▸ Start ()

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Arrays : MonoBehaviour {
5
6      int[,] pigletArray = new int[10, 3];
7      int[,] pigArray = new int[10, 3];
8
9      void Start () {
10
11         for(int i = 0; i < 10; i++)
12         {
13             // piglet Array initial values
14             pigletArray[i, 0] = 1; // visible: 0=no; 1=yes
15             pigletArray[i, 1] = 0; // corn
16             pigletArray[i, 2] = 0; // water
17
18             // pig Array initial values
19             pigArray[i, 0] = 0; // visible: 0=no; 1=yes
20             pigArray[i, 1] = 0; // corn
21             pigArray[i, 2] = 0; // water
22         }
23     }
24
25     void Update () {
26
27     }
28 }
```

مرغ ها و جوجه ها

جوجه ها و مرغ ها نیز همان داده های شبیه به گرازها را نیاز دارند :

Arrays.cs

Arrays ▸ No selection

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class Arrays : MonoBehaviour {
5
6      int[,] pigletArray = new int[10, 3];
7      int[,] pigArray = new int[10, 3];
8      int[,] chickArray = new int[10, 3];
9      int[,] chickenArray = new int[10, 3];
10
11     void Start () {
12
13         for(int i = 0; i < 10; i++)
14         {
15             // piglet Array initial values
16             pigletArray[i, 0] = 1; // visible: 0=no; 1=yes
17             pigletArray[i, 1] = 0; // corn
18             pigletArray[i, 2] = 0; // water
19
20             // adult pig Array initial values
21             pigArray[i, 0] = 0; // visible: 0=no; 1=yes
22             pigArray[i, 1] = 0; // corn
23             pigArray[i, 2] = 0; // water
24
25             // baby chick Array initial values
26             chickArray[i, 0] = 1; // visible: 0=no; 1=yes
27             chickArray[i, 1] = 0; // corn
28             chickArray[i, 2] = 0; // water
29
30             // adult chicken Array initial values
31             chickenArray[i, 0] = 0; // visible: 0=no; 1=yes
32             chickenArray[i, 1] = 0; // corn
33             chickenArray[i, 2] = 0; // water
34         }
35     }
36
37     void Update () {
38
39     }
40 }
```

حالا ۴ آرایه داریم ، یکی برای هر نوع حیوان !

غله و آب

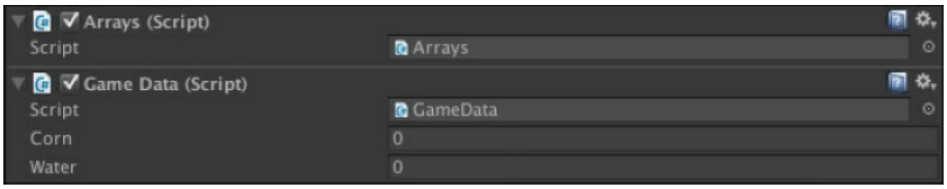
محیط بازی ما مقدار نامحدودی از آب و غله دارد . هنگامی که بازیکن غله جمع آوری می کند ، از مزرعه هیچ غله ای کم نمی شود . همچنین هنگامی که بازیکن آب جمع آوری می کند ، از رودخانه هیچ آبی کم نمی شود . بنابراین برای غله و آب فقط نیاز به ایجاد اعداد صحیح داریم . اجازه دهید یک اسکریپت جدید ایجاد و آن را GameData بنامیم . سپس آن را به Main Camera الصاق کنید :

GameData.cs

No selection

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class GameData : MonoBehaviour {
5
6     public int corn = 0;
7     public int water = 0;
8
9     void Start () {
10
11    }
12
13    void Update () {
14
15    }
16 }
```

در خط ۶ و ۷ برای آب و غله اعداد صحیحی به صورت public ایجاد کرده ایم . ما متغییرها را اعلام و در همان نقطه ، مقدار اولیه را به آنها واگذار کردیم . با اعلام این متغییرها به صورت public ، می توانیم در نمای Inspector ، مقادیر آنها را مشاهده و تغییر دهیم :



می بینید که در نمای Inspector ، مقادیر arrays قابل دسترسی نیستند اما به مقادیر GameData می توان دسترسی داشت . علت این است که متغییرهای موجود در GameData (corn و water) به صورت public اعلام شده اند . اما متغییرهای تعریف شده در اسکریپت Arrays به صورت private اعلام شده اند .

مقداردهی اولیه داده ها

ما در این بازی ، سه گروه اصلی از داده ها داریم . گروه اول مخصوص بچه گرازها و گرازهای بالغ است . گروه دوم مخصوص جوجه ها و مرغ هاست . هر دو این گروه ها در اسکریپت Arrays کدنویسی شده اند . داخل این اسکریپت ، اعلام داده ها و مقداردهی اولیه آنها در تابع Start انجام شده که باعث می شود ، هنگامی که شئ الصاق شده به این اسکریپت مقداردهی اولیه می شود ، این بخش از اسکریپت اجرا شود . چون این اسکریپت به گزینه Main Camera الصاق شده ، هنگامی که بازی در ابتدا بارگذاری می شود ، اسکریپت اجرا خواهد شد .

سومین گروه از داده ها یعنی متغییرهای corn و water در تابع Start اسکریپت GameData تعریف شده اند که این اسکریپت نیز به Main Camera الصاق شده است .

اشاره

نمای Inspector اسامی اسکریپت ها را به طور متفاوت با نمای Project نشان می دهد . مثلا ما اسکریپت GameData را ایجاد کردیم و در نمای Project به درستی نمایش داده می شود . در نمای Inspector ، یک فاصله بین دو کلمه نشان داده می شود . تمام متغییرهای نشان داده شده در نمای Inspector نیز با فاصله نمایش داده می شوند . نگران نباشید . این ضربه ای به بازی نمی زند !

اسکریپت های منتخب

می توانید بازی را به صورت کامل از صفحه وب کتاب دانلود کرده و همه اسکریپت های بازی را مرور کنید . در این قسمت ، من چند تا از اسکریپت های بازی را همراه با توضیحاتی آورده ام تا به شما در خواندن اسکریپت ها ، کمک شود .

اسکریپت منتخب – متغییرهای جهانی

اسکریپت زیر ، اسکریپت GameData است که قبلا ایجاد کردیم اما با چند تغییر مهم . به یاد دارید که این اسکریپت را در نمای Hierarchy به Main Camera الصاق کردیم . اسکریپت ها باید به اشیاء بازی الصاق شوند وگرنه اجرا نخواهند شد .

این اسکریپت متغییرهای water و corn را نصب کرده طوری که بتوانند به صورت جهانی مورد استفاده قرار گیرند . هر متغییر در ابتدا به شکل private static int و همراه با پیشوند یک _ به اسم متغییر ، اعلام می شود . اینها متغییرهای اصلی ما هستند . می توانیم از GameData.water و GameData.corn برای به دست آوردن مقدار فعلی این متغییرها ، استفاده کنیم .

برای تغییر این متغییرها ، می توانیم با استفاده از کدهائی مثل `GameData.water = GameData.water +2;` یا `GameData.water +=2;` ، آنها را افزایش یا کاهش دهیم :

```
using UnityEngine;
using System.Collections;

public class GameData : MonoBehaviour {

	private static int _water = 0;
	private static int _corn = 0;

	public static int water
	{
		get { return _water; }
		set { _water = value; }
	}

	public static int corn
	{
		get { return _corn; }
		set { _corn = value; }
	}
}
```

اسکریپت منتخب – فراخوانی انیمیشن **take**

هدف ما این است که بازیکن را قادر کنیم از کلید T استفاده کرده و انیمیشن take را فراخوانی کند . در اسکریپت زیر چند کار انجام می شود و با استفاده از یک if و ارزیابی ((Input.GetKeyDown (KeyCode.T) تعیین می کنیم که آیا کاربر کلید T صفحه کلید را فشرده یا نه .

اگر بازیکن کلید T را فشرده باشد ، انیمیشن take پخش می شود . برای این کار ابتدا anim را به عنوان یک متغییر نوع Animator اعلام می کنیم . سپس یک عدد صحیح به نام takeHash ایجاد و با ("Take") Animator.StringToHash = مقدار آن را با انیمیشن take نصب می کنیم . حالا می توانیم با anim.Play (takeHash) به انیمیشن مراجعه کنیم :

```
using UnityEngine;
using System.Collections;

public class Take : MonoBehaviour {

	Animator anim;
	int takeHash = Animator.StringToHash ("Take");

	void Start () {
		anim = GetComponent<Animator> ();
	}

	void Update () {
		if (Input.GetKeyDown (KeyCode.T)) {
			anim.Play (takeHash);
```

```
                GameData.water = GameData.water +1;
                Debug.Log (GameData.water);
            }
        }
}
```

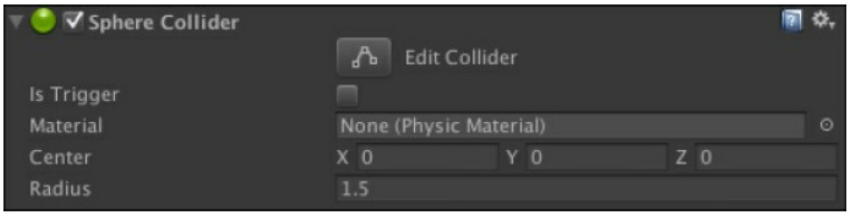
اسکریپت منتخب – غذادادن به حیوانات مزرعه

این کار هنگامی انجام می شود که کاراکتر بازیکن یعنی Colt با یک حیوان برخورد کند . برای انجام این کار ما یک برخوردکننده کره شکل (Sphere Collider) به هر حیوان اضافه می کنیم . برای این کار مراحل زیر را انجام دهید :

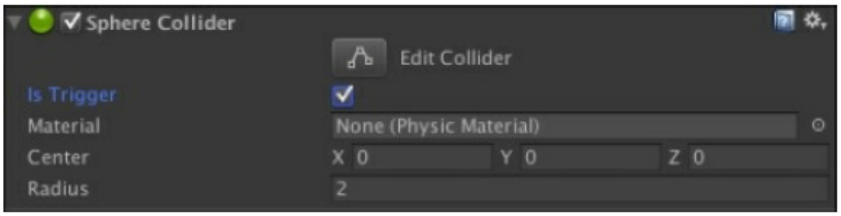
۱. در نمای Hierarchy ، حیوان مزرعه را انتخاب کنید .

۲. به منوی Component | Physics | Sphere Collider بروید . این کار یک کره در اطراف حیوان ایجاد می کند .

۳. در نمای Inspector ، شعاع این کره (Radius) را به شکل زیر طوری تنظیم کنید که ، حیوان در داخل کره قرار گیرد :



۴. سپس در قسمت Sphere Collider مربوط به پنل Inspector ، باید تیک گزینه Is Trigger را فعال کنیم :



۵. حالا این collider را اسکریپت نویسی می کنیم . از منو Assets | Create | C# Script یک اسکریپت جدید ایجاد کنید و آن را FeedChicken بنامید .

۶. اسکریپت را به شکل زیر ویرایش کنید :

```
using UnityEngine;
using System.Collections;

public class FeedChicken : MonoBehaviour {
```

```
	void OnTriggerEnter(Collider other) {
		if (GameData.water > 0) {
			GameData.water = GameData.water - 1;
			// add water to chicken's holdings
		}
		if (GameData.corn > 0) {
			GameData.corn = GameData.corn - 1;
			// add corn to chicken's holdings
		}
	}
}
```

۷. آخرین کار واگذاری این اسکریپت به یکی از اشیاء بازی است . در نمای Hierarchy و درحالیکه شئ حیوان انتخاب شده است ، منو Component | Scripts | Feed Chicken را انتخاب کنید . این کار اسکریپت را به شئ بازی واگذار می کند . قبلا این کار را با درگ کردن اسکریپت از نمای Project به داخل شئ بازی در نمای Hierarchy انجام می دادیم . هر دو این شیوه ها به خوبی کار می کنند :

اشاره : می توانید یکی از چهار حیوان مزرعه را برنامه نویسی کرده و برای بقیه ، این کدها را کپی کنید تا زمانتان ذخیره شود .

سازمان دهی اسکریپت ها

اگر در حال کار بر روی یک پروژه بازی بزرگ باشید ، می فهمید که مدیریت تعداد زیادی از اسکریپت ها سخت می شود . چند تاکتیک برای حل این مشکل وجود دارد .

اولین و شاید آسانترین آنها ، ایجاد یک پوشه اسکریپت در نمای Project و نگهداری همه اسکریپت ها در آنجاست .

دومین شیوه ، اضافه کردن متنی مثل _script به ابتدای اسم فایل اسکریپت هاست تا شناسائی آنها آسان تر شود . با این شیوه ، اسکریپت FeedChicken.cs به شکل script_FeedChicken.cs نام گذاری خواهد شد .

در نمای Project ، چهار Favorites وجود دارد که هرکدام با یک آیکون جستجو نمایش داده شده اند . کلیک بر روی All Scripts آنها ، لیستی از همه اسکریپت های موجود در پروژه را باز خواهد کرد .

خلاصه

در این فصل درباره اسکریپت نویسی با C# در یونیتی ، آموزش دیدید . به صورت خیلی مختصر با اصول اولیه C# شامل گرامر و قواعد نام گذاری آشنا شدید . در زمینه نحوه استفاده از MonoDevelop و

شیوه ویرایش اسکریپت ها با استفاده از این ابزار که همراه یونیتی وجود دارد ، آشنا شدید . این فصل نحوه ایجاد اسکریپت ها و الصاق آنها به اشیاء بازی را نشان داد . شما را درخصوص نوشتن اولین اسکریپت C# برای بازیمان راهنمائی کردیم .

در فصل بعد ، رابط کاربری گرافیکی را به بازیمان اضافه خواهیم کرد . ما سیستم جدید Unity 5 UI را بررسی می کنیم . ما یک منو تمام صفحه ، یک HUD و یک minimap ایجاد و اسکریپت نویسی می کنیم . در انتهای فصل ، شما یک بازی پیشرفته همراه با مجموعه کاملی از خصوصیات رابط کاربری گرافیکی ، در اختیار خواهید داشت .

فصل ۶ : اضافه کردن رابط کاربری گرافیکی

بازی ما تقریبا کامل شده است . یکی از خصوصیات بازی که بازیکنان انتظار آن را دارند ، رابط کاربری گرافیکی [graphical user interface (GUI)] است . به عبارت دیگر ، مجموعه ای از تصاویر ، متن ها و دکمه ها که بازیکن در طول بازی با آنها سروکار دارد . ما نیز تعدادی از اجزای کلیدی GUI را به بازیمان اضافه می کنیم .

این فصل با مقدمه ای درباره سیستم رابط کاربری یونیتی [Unity 5 User Interface (UI)] شروع می شود . همچنین اجزای GUI زیر را نیز ایجاد می کنیم :

Heads-up display (HUD)
Minimap
منو تمام صفحه

قبل از شروع ساختن GUI ، اهمیت اجزای GUI و ابزارهای UI یونیتی را بررسی خواهیم کرد .

بعد از مطالعه این فصل :

با اهمیت استفاده از GUI آشنا می شوید .

با سیستم UI یونیتی ۵ آشنا می شوید .

قادر به ایجاد HUD خواهید بود .

قادر به ایجاد minimap ها خواهید بود .

قادر به ایجاد منوهای تمام صفحه خواهید بود .

مقدمه ای درباره GUI

GUI مجموعه ای از آیتم های تصویری مثل متن ، دکمه ها و تصاویر است که تعامل کاربر با نرم افزار را آسان تر می کند . GUI ها همچنین به منظور فراهم کردن بازخورد به بازیکنان ، مورد استفاده قرار می گیرند . در مورد بازی ما ، GUI به بازیکنان اجازه می دهد تا با بازی ما تعامل داشته باشند . بدون GUI ، کاربر نحوه استفاده از بازی را نمی داند . تصویر زیر نشان دهنده رابط های کاربری قدیمی است :

ما همیشه از GUI ها استفاده می کنیم ولی ممکن است زیاد به آنها توجه نکنیم مگر اینکه خیلی ضعیف طراحی شده باشند . اگر همیشه سعی می کنید بفهمید که چگونه از یک برنامه نصب شده روی گوشیتان استفاده کنید یا نحوه اجرای یک کار خاص را در یک نرم افزار نمی فهمید ، احتمالا با یک GUI که خیلی ضعیف طراحی شده است ، مواجهید !

کاربرد GUI

هدف ما ایجاد یک GUI برای بازیمان است که اطلاعات بازی را به کاربران ارائه کند و اجازه تعامل بین کاربر و بازی را بدهد . پس GUI ها دو هدف اصلی دارند : بازخورد و کنترل . بازخورد توسط بازی برای کاربر ایجاد می شود و کنترل به کاربر داده می شود و توسط ورودی کاربر مدیریت می شود . اجازه دهید کمی دقیق تر به این دو نگاه کنیم .

بازخورد

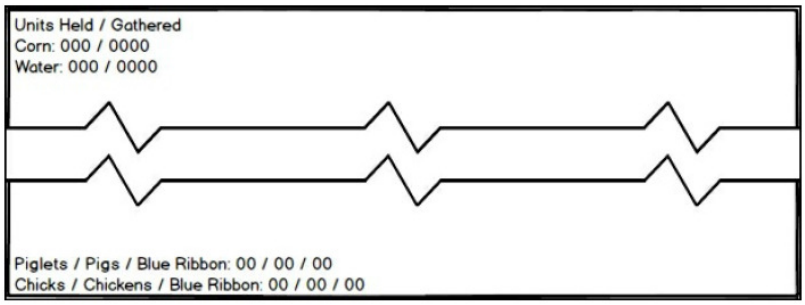
بازخورد می تواند به فرم های زیادی بیاید . رایج ترین فرم های بازخورد بازی ، تصویری و صوتی هستند . بازخورد تصویری می تواند به سادگی نمایش یک متن روی صفحه بازی باشد . مثلا امتیاز فعلی بازیکن که همیشه روی صفحه بازی نمایش داده می شود . بازخورد تصویری می تواند فاقد متن باشد مثلا دود ، آتش ، انفجارها و افکتهای گرافیکی دیگر.
بازخورد صوتی می تواند به سادگی صدای یک کلیک باشد ، مثلا زمانیکه کاربر یک دکمه را کلیک می کند یا به پیچیدگی صدای رادار باشد ، مثلا زمانیکه دشمنان روی ردیاب صوتی تشخیص داده شده باشند .

ما اجزای GUI زیر را به منظور تولید بازخورد در بازیمان داریم :

تعداد واحدهای غله ای که بازیکن درحال حاضر دارد .
مجموع تعداد واحدهای غله ای که بازیکن جمع آوری کرده است .

تعداد واحدهای آبی که بازیکن درحال حاضر دارد .
مجموع تعداد واحدهای آبی که بازیکن جمع آوری کرده است .
تعداد بچه گراز ها ، گرازها و گراز های بالغ و رشد یافته ای که در حال حاضر در مزرعه وجود دارد .
تعداد جوجه ها ، مرغ ها و مرغ های بالغ و رشد یافته ای که در حال حاضر در مزرعه وجود دارد .

تصویر زیر نشان می دهد که چینش اجزای GUI بازی ما به چه شکلی خواهد بود :



کنترل

دومین کاربرد GUI ، دادن کنترل بازی به بازیکنان است . انواع زیادی از کنترل ها وجود دارد . ساده ترین آنها استفاده از دکمه ها یا منوهای بازی است .
اغلب بازی های کنسول فاقد رابط GUI خوب هستند مخصوصا در زمینه کنترل بازی . اگر بازی های کنسول را بازی کرده باشید ، احتمالا وقت زیادی را برای یاد گرفتن نحوه کنترل بازی گذاشته اید . چطور باید بپرید ؟ اسلحه ها را عوض کنید ، خم شوید ، نارنجک پرتاب کنید و ...
با این اوصاف ، کنترل علاوه بر گرافیکی بودن ، اغلب فیزیکی است . اجزای فیزیکی کنترل شامل صفحه کلید ، ماوس ، دسته های بازی و ... هستند .

ما از اجزای GUI زیر به منظور کنترل بازیمان استفاده می کنیم :

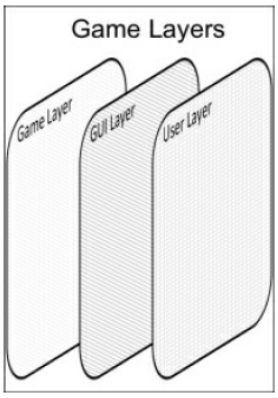
کلیدهای جهتی صفحه کلید ، کارکتر اصلی را کنترل می کنند .
کلید space صفحه کلید باعث پرش کاراکتر اصلی می شود .
کلید T صفحه کلید ، انیمیشن take و عملکرد آن را فراخوانی می کند .
از دکمه ها در منو تمام صفحه بازی استفاده می کنیم .

بازخورد و کنترل

اجزای بازخورد و کنترل GUI معمولا جفتی کار می کنند . هنگامی که یک دکمه را کلیک می کنید ، معمولا هر دو افکت صوتی و تصویری را در بر دارد و همچنین عملی انجام می شود . هنگامی که شما روی یک صندوق طلا کلیک می کنید (کنترل) ، صندوق باز می شود (بازخورد تصویری) و شما صدای لولاهای چوبی و قدیمی صندوق را می شنوید (بازخورد صوتی) .

لایه های بازی

در یک سطح ابتدائی ، برای هر بازی سه لایه وجود دارد . سطح اصلی ، لایه بازی (Game Layer) است . این بازی شماست و در این کتاب ، بازی **Little Farmer Colt** است . لایه فوقانی ، لایه کاربر (User Layer) است . این شخص واقعی است که در حال بازی کردن است . پس لایه ای در این میان وجود دارد : لایه رابط گرافیکی کاربر (GUI Layer) که به عنوان یک واسطه بین بازی و بازیکن عمل می کند . تصویر زیر این سه لایه را نشان می دهد :



مشخص می شود که طراحی و پیاده سازی یک GUI خوب و کاربردی برای بازی و موفقیت آن خیلی مهم است .

سیستم Unity 5 UI

سیستم UI یونیتی به تازگی مهندسی و ساخته شده و قدرتمندتر از همیشه است . شاید مهمترین مفهومی که باید متوجه شوید ، شئ Canvas است . همه آیتم های UI داخل یک canvas قرار می گیرند . پروژه ها و صحنه ها می توانند بیش از یک canvas داشته باشند . canvas را می توانید به شکل یک محفظه برای اجزای UI تصور کنید .

Canvas

برای ایجاد یک canvas ، منو GameObject | UI | Canvas را انتخاب کنید . از منو GameObject | UI می توانید ببینید که ۱۱ نوع مختلف از آیتم های UI به شکل زیر وجود دارد :

شما می توانید اولین آیتم UI خودتان مثلا یک دکمه را ایجاد کنید تا یونیتی به صورت خودکار یک canvas را برای شما ایجاد کند و آن را به نمای Hierarchy اضافه می کند . برای ایجاد آیتم UI بعدی ، canvas را در نمای Hierarchy انتخاب و پررنگ کرده و سپس به منو GameObject | UI رفته تا آیتم جدید UI را انتخاب کنید .

جدول زیر توضیح کوتاهی درباره هر یک از اجزای UI می دهد :

Panel : یک شئ قاب شکل

Button : یک دکمه استاندارد که می تواند کلیک شود .

Text : متنی همراه با فرمت استاندارد متن

Image : تصاویر می توانند ساده باشند ، به صورت تکه تکه و ... باشند .

Raw image : فایل بافت

Slider : کنترل Slider همراه با مقادیر ماکزیمم و مینیمم

Scrollbar : کنترل Scrollbar همراه با مقادیری بین 0 و 1

Toggle : یک کنترل استاندارد checkbox

Input field : فیلد ورودی متن

Canvas : نگهدارنده ای برای آیتم های UI

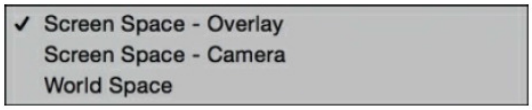
Event system : به ما اجازه راه اندازی اسکریپت ها از آیتم های UI را می دهد . هنگامی که یک canvas ایجاد می کنید ، یک Event system به صورت خودکار ایجاد می شود .

می توانید چندین canvas در بازی هایتان داشته باشید . با ساختن بازی های بزرگ تر ، نیاز به استفاده از بیش از یک canvas خواهید داشت .

شیوه رندر

در نمای Inspector چند تنظیمات درخصوص شئ canvas وجود دارد . اولی شیوه رندر (Render Mode) است . به شکل زیر سه تنظیم وجود دارد :

Screen Space – Overlay
Screen Space – Camera
World Space



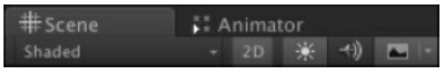
ما در بازیمان ، از این شیوه رندر استفاده می کنیم . در این شیوه رندر ، هنگامی که کاربر اندازه یا رزولیشن صفحه بازی را تغییر می دهد ، canvas به صورت اتوماتیک تغییر اندازه می یابد . دومین شیوه رندر ، Screen Space – Camera ، دارای ویژگی است که تعیین می کند canvas تا چه فاصله ای از دوربین رندر شود . سومین شیوه رندر World Space است . این شیوه به شما بیشترین کنترل را می دهد و می تواند مثل هر شئ دیگر بازی ، دستکاری شود .
پیشنهاد می کنم شیوه های مختلف رندر را آزمایش و تجربه کنید تا بفهمید کدام را می پسندید و کی باید از هر کدام استفاده کنید .

ایجاد یک GUI

ایجاد یک GUI در یونیتی کار ساده ای است . ما باید ابتدا یک canvas ایجاد کنیم یا هنگامی که اولین آیتم UI را ایجاد می کنیم ، یونیتی canvas را برای ما به صورت اتوماتیک ایجاد می کند . سپس آیتم های

دلخواه UI را به canvas اضافه می کنیم . هنگامی که همه آیتم های لازم در canvas قرار گرفت ، می توانید آنها را مرتب و فرمت بندی کنید .

هنگام قرار دادن آیتم های UI روی canvas ، بهتر است در نمای Scene به مد 2D بروید . این کارها را کمی آسان تر می کند :



اشاره

اگر از ورژن های قبلی یونیتی استفاده کرده باشید ، در خصوص ایجاد و ارجاع به آیتم های GUI ، چند چیز تغییر کرده است . مثلا باید قبل از ارجاع به آیتم های UI از دستور `UnityEngine.UI;` استفاده کنید . همچنین به جای استفاده از `public GUIText waterHeld;` برای ارجاع به یک GUI متنی ، باید از `public Text waterHeld;` استفاده کنید .

HUD

HUD اطلاعاتی گرافیکی و متنی است که همیشه در دسترس کاربر است . کاربر کار خاصی انجام نمی دهد به جز اینکه به یک قسمت مشخص از صفحه نگاه می کند تا آنچه نمایش داده شده را بخواند . مثلا اگر یک بازی مسابقه ای ماشین انجام می دهید ؛ ممکن است یک کیلومتر شما ، سرعت سنج ، قطب نما ، سطح باک بنزین ، فشار هوا و نمایشگر های تصویری دیگری همیشه روی صفحه وجود داشته باشند .

بازی ما ، روی صفحه دارای اجزای زیر است :

تعداد واحدهای غله ای که بازیکن درحال حاضر دارد .
مجموع تعداد واحدهای غله ای که بازیکن جمع آوری کرده است .
تعداد واحدهای آبی که بازیکن درحال حاضر دارد .
مجموع تعداد واحدهای آبی که بازیکن جمع آوری کرده است .
تعداد بچه گراز ها ، گرازها و گراز های بالغ و رشد یافته ای که در حال حاضر در مزرعه وجود دارد .
تعداد جوجه ها ، مرغ ها و مرغ های بالغ و رشد یافته ای که در حال حاضر در مزرعه وجود دارد .

ایجاد یک HUD

برای ایجاد HUD بازیمان ، مراحل زیر را دنبال کنید :

۱. پروژه بازی را باز کرده و صحنه را بارگذاری کنید .

۲. منو GameObject | UI | Text را انتخاب کنید . این کار باعث می شود یک شئ Canvas به نمای Hierarchy اضافه شود که دارای یک آیتم متنی خواهد بود . این آیتم را به شکل زیر HUD_Line_1 بنامید :

۳. در نمای Hierarchy ، گزینه HUD_Line_1 را انتخاب کنید . سپس در نمای Inspector ، Text را به "Units Held / Gathered" تغییر دهید .

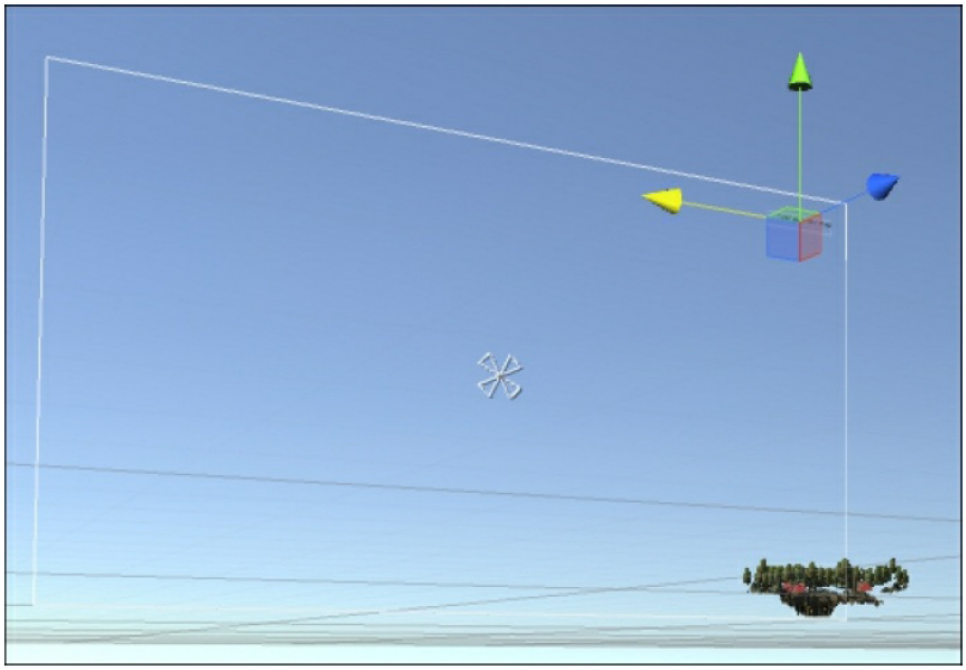
۴. در نمای Inspector ، مقدار Font Size را به 24 تغییر دهید .

۵. در نمای Inspector ، گزینه Horizontal Overflow را به شکل زیر از Wrap به Overflow تغییر دهید :

۶. در نمای Scene به سمت بیرون زوم کنید تا بتوانید Canvas را ببینید . از ابزارهای تغییر شکل استفاده کرده و آیتم HUD_Line_1 را به گوشه بالائی سمت چپ صفحه ببرید .

اشاره

بسته به چگونگی نمایش صحنه در نمای Scene ، ممکن است نیاز به استفاده از ابزار hand برای چرخاندن صحنه داشته باشید . بنابراین اگر متن GUI به صورت وارونه نمایش داده می شود ، صحنه را بچرخانید تا درست شود .



۷. منو GameObject | UI | Text را انتخاب کنید . این کار ، دومین آیتم متنی را به GUI اضافه می کند . آن را HUD_Line_2 بنامید .

۸. در نمای Hierarchy ، آیتم HUD_Line_2 را انتخاب کنید . سپس در نمای Inspector ، Text را به "Corn: 000 / 0000" تغییر دهید .

۹. در نمای Inspector ، مقدار Font Size را به 24 تغییر دهید .

۱۰. در نمای Inspector ، گزینه Horizontal Overflow را Wrap به Overflow تغییر دهید .

۱۱. در نمای Scene به سمت بیرون زوم کنید تا بتوانید Canvas را ببینید . از ابزارهای تغییر شکل برای قرار دادن آیتم HUD_Line_2 درست در زیر آیتم HUD_Line_1 استفاده کنید .

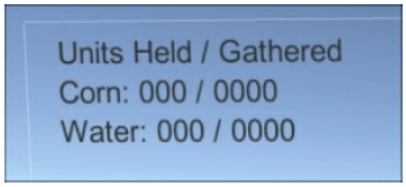
۱۲. منو GameObject | UI | Text را انتخاب کنید و آیتم اضافه شده را HUD_Line_3 بنامید .

۱۳. در نمای Hierarchy آیتم HUD_Line_3 را انتخاب کرده و سپس در نمای Inspector ، گزینه Text را به "Water: 000 / 0000" تغییر دهید .

۱۴. در نمای Inspector ، مقدار Font Size را به 24 تغییر دهید .

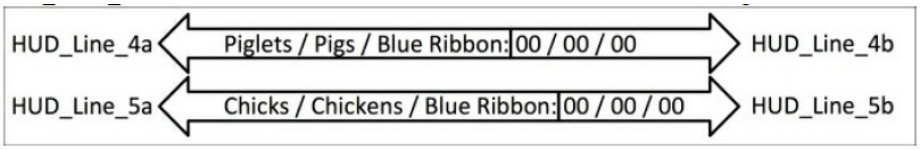
۱۵. در نمای Inspector ، گزینه Horizontal Overflow را Wrap به Overflow تغییر دهید .

۱۶. در نمای Scene به سمت بیرون زوم کنید تا بتوانید Canvas را ببینید . از ابزارهای تغییر شکل برای قرار دادن آیتم HUD_Line_3 درست در زیر آیتم HUD_Line_2 استفاده کنید . حالا Canvas شما باید مثل تصویر زیر باشد :

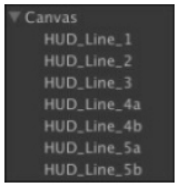


۱۷. سپس با استفاده از منو GameObject | UI | Text ، ۴ آیتم دیگر ایجاد می کنیم .

۱۸. این ۴ آیتم متنی جدید را HUD_Line_4a ، HUD_Line_4b ، HUD_Line_5a و HUD_Line_5b بنامید . این ۴ آیتم برای ساختن عملکرد زیر استفاده می شود :



حالا شئ Canvas در نمای Hierarchy به شکل زیر می شود :



توجه

تصویر قبل ، اشیاء فرزند Canvas را نشان می دهد . همچنین بیانگر ترتیب رندر آنها نیز هست یعنی با چه ترتیبی روی canvas نمایش داده خواهند شد . داشتن اشیاء بازی که روی هم بیافتند یا دیگری را بپوشانند ، ممکن است . این موضوع را در هنگام قرار دادن آیتم های UI باید مورد توجه قرار دهید .

۱۹. تغییرات زیر را بر روی هر یک از ۴ آیتم متنی که قبلا ایجاد کردیم ، انجام دهید :

تغییر مقدار Font Size به 24

تغییر گزینه Horizontal Overflow از Wrap به Overflow

جابه جا کردن اشیاء طوری که به شکل زیر چینش شوند :

۲۰. صحنه و پروژه را ذخیره کنید .

۲۱. بازی را play کرده و ببینید HUD چطور ظاهر می شود .

حالا آیتم های متنی GUI داریم که می توانیم در اسکریپت ها به آنها ارجاع کنیم بنابراین می توانیم بازخوردی را که از طریق HUD برای کاربر فراهم می شود ، کنترل کنیم .

Mini-maps

mini-maps (نقشه های کوچک) برای بازیکنان بازی ، یک وسیله کمکی تصویری فراهم می کند که به آنها کمک می کند دورنما و جهتشان را در بازی حفظ کنند . بسته به بازی ، mini-maps می تواند برای اهداف متفاوتی به کار گیری شود . مثلا توانائی دیدن نقشه ای که مشرف بر اردوگاه دشمنان باشد ؛ نمائی از نقشه بازی همراه با شاخص هائی که نشان دهنده دوستان و دشمنان بازی باشند و ...
هر چند ما برای بازی خود ، نیازی به mini-map نداریم ؛ اما برای نشان دادن نحوه ایجاد آنها به شما ، آن را برای بازیمان ایجاد می کنیم .

ایجاد یک mini-map

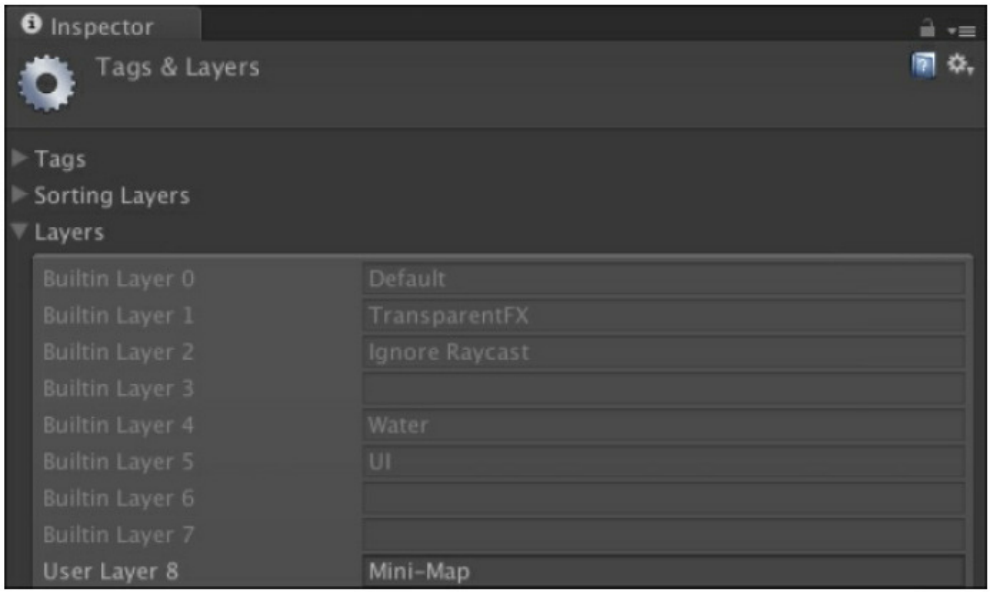
مراحل زیر را به منظور ایجاد یک mini-map برای بازیمان ، دنبال کنید :

۱. منو GameObject | Camera را انتخاب کنید .

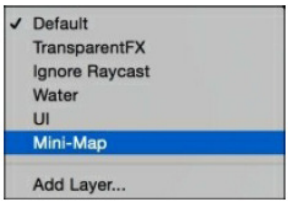
۲. در نمای Hierarchy ، اسم را از Camera به Mini-Map تغییر دهید .

۳. در حالیکه این دوربین mini-map انتخاب شده باشد ، به نمای Inspector رفته و دکمه Layer را کلیک کنید سس از منو باز شده Add Layer را انتخاب کنید .

۴. در User Layer بعدی و در دسترس ، اسم Mini-Map را وارد کنید :

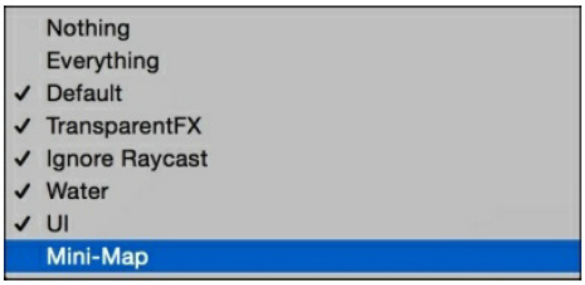


۵. در نمای Hierarchy ، گزینه Mini-Map را انتخاب کرده و سپس Layer | Mini-Map را انتخاب کنید . حالا دوربین mini-map جدید به لایه Mini-Map واگذار شده است :



۶. سپس اطمینان حاصل می کنیم که دوربین اصلی ، دوربین Mini-Map را رندر نکند . در نمای Hierarchy ، گزینه Main Camera را انتخاب کنید .

۷. در نمای Inspector ، گزینه Culling Mask را انتخاب کرده و از منو زیر ، تیک Mini-Map را بردارید تا از حالت انتخاب شده خارج شود :



۸. حالا آماده تکمیل تنظیمات دوربین mini-map هستیم . در نمای Hierarchy ، گزینه Mini-Map را انتخاب کنید .

۹. در نمای Scene و با استفاده از ابزارهای تغییر شکل ، شئ دوربین را طوری تنظیم کنید که در بالای مزرعه Colt قرار گیرد . در نمای Scene به مقدار لازم به داخل و خارج زوم کنید تا مطمئن شوید که دوربین فقط مزرعه Colt را می بیند .

۱۰. در نمای Inspector و زیر Camera ، تنظیمات را به شکل زیر وارد کنید :

Setting	Value
Clear flags	Depth only
Culling mask	Everything
Projection	Orthographic
Size	25
Clipping planes	Near 0.3; Far 1000
Viewpoint rect	X 0.75; Y 0.75; W 1; H 1
Depth	1
Rendering path	User player settings
Target texture	None
Occlusion culling	Selected
HDR	Not selected

۱۱. درحالیکه دوربین Mini-Map هنوز انتخاب شده باشد ، در نمای Inspector روی هر یک از کامپونت های Flare Layer ، GUI Layer و Audio Listener راست کلیک کنید و Remove Component را انتخاب کنید .

۱۲. صحنه و پروژه را ذخیره کنید .

۱۳. حالا آماده تست mini-map هستیم . بازی را play کرده و می بینید که حتی با حرکت Colt به سمت انبار Pa Poo ، مزرعه Colt در mini-map قابل دیدن است :

Mini-maps می تواند یکی از اجزای قدرتمند بازی ها باشد . اگر قصد استفاده از Mini-maps در بازی هایتان را دارید ، نکات زیر را در نظر داشته باشید :

۱. مطمئن شوید که اندازه mini-map ، محدوده زیادی از محیط بازی شما را نگیرد . هیچ چیز بدتر از این نیست که از سمت دشمنی مورد شلیک قرار بگیرید که او را به خاطر وجود mini-map ندیده اید !

۲. mini-map باید یک هدف داشته باشد و فقط هنگامی که مورد نیاز باشد ، استفاده شود مثلا به بازیکن کمک کند با آگاهی از شرایط بازی ، تصمیمی را بگیرد . در بازی ما ، بازیکن می تواند در حالیکه برای جمع آوری آب و غله از مزرعه خارج شده است ، مراقب حیوانات مزرعه Colt نیز باشد و از آنها چشم برندارد !

۳. آیتم های روی mini-map باید کاملا قابل دیدن باشند . اکثر بازی ها از نقطه های قرمز رنگ به عنوان دشمنان استفاده می کنند یا از نقطه های آبی به عنوان دوستان و از نقطه های زرد به عنوان نیروهای بی طرف . این نوع رنگ بندی ها ، با یک مرور کلی ، اطلاعات زیادی را برای کاربران فراهم می کنند .

۴. مطلوب این است که کاربر باید بتواند mini-map را به یکی از گوشه های دلخواهش در صفحه انتقال دهد و یا آن را کاملا از روی صفحه حذف نماید . در بازی ما ، mini-map در گوشه بالائی سمت راست صفحه بازی قرار داده شد طوری که اشیاء HUD سر راهش نباشند .

منوهای بازی

بسیاری از بازی ها هنگامی که در ابتدا اجرا می شوند ، با یک منو تمام صفحه (full-screen menu) شروع می شوند . آیتم هائی که معمولا روی این صفحه ها وجود دارند شامل متن ها ، تصاویر و دکمه ها

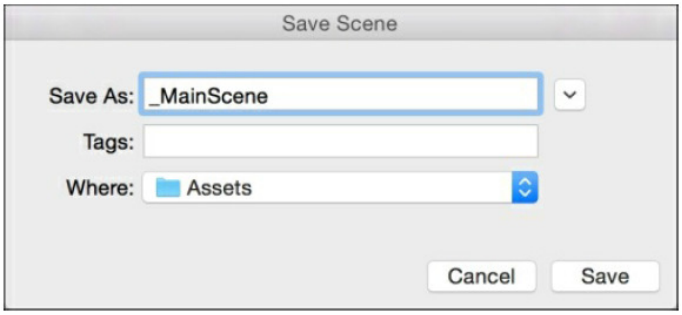
هستند . متن معمولا عنوان بازی را نمایش می دهد و تصاویر نیز مربوط به بازی هستند . یک یا چند دکمه نیز از آیتم های معمولی UI روی صفحه اصلی بازی هستند . یک بازی ساده ممکن است فقط یک دکمه Play داشته باشد درحالیکه بازی های پیشرفته تر شامل دکمه های بیشتری هستند .

با انجام مراحل زیر ، یک صحنه جدید برای بازیمان ایجاد می کنیم . این صحنه اصلی است که دکمه play را نمایش می دهد . هنگامی که کاربر این دکمه را کلیک می کند ، صحنه مزرعه کشاورز بارگذاری خواهد شد :

۱. صحنه فعلی و پروژه را ذخیره کنید .

۲. برای ایجاد یک صحنه جدید ، منو File | New Scene را انتخاب کنید .

۳. این صحنه را با نام _MainScene ذخیره کنید :



اشاره

من اسم همه صحنه ها را با یک _ شروع می کنم تا همیشه در بالای نمای Project نشان داده شوند . اگر شما نیز صحنه های زیادی دارید ، می توانید در نمای Project یک پوشه Scenes ایجاد کنید . برای بازی های بزرگ تر ، صحنه ها را با اعداد ترتیبی مثلا به شکل 01_Main ، 02_Jungle ، 03_Cave و ... نام گذاری کنید .

۴. حالا باید در نمای Hierarchy ، اطلاعات _MainScene همراه با دوربین اصلی آن را ببینید . نگران نباشد ، همه اطلاعات _FarmScene هنوز سالم و دست نخورده وجود دارند .

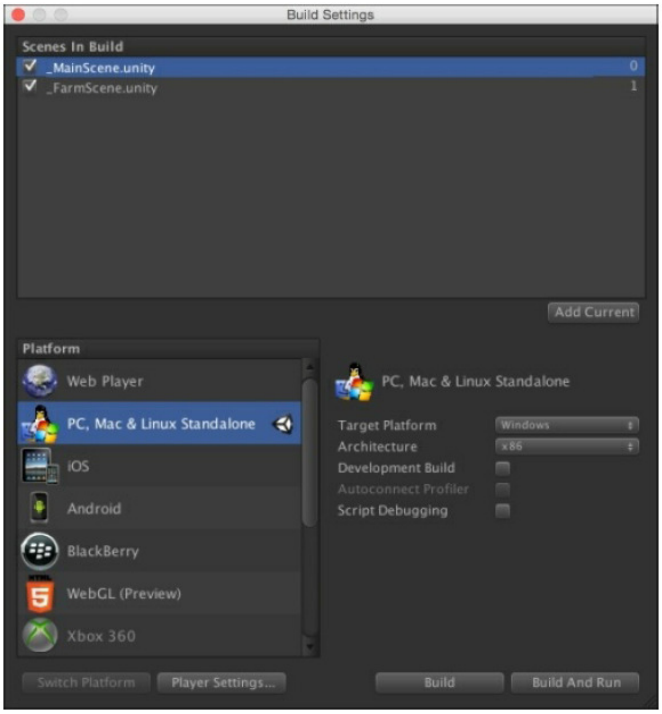
اشاره

می توانید در نمای Project ، با دوبار کلیک کردن بر روی اسم صحنه ، بین صحنه ها جابه جا شوید .

۵. با انتخاب منو File | Build Settings به پنجره build settings دسترسی پیدا کنید .

۶. از نمای Project ، هر دو صحنه را به قسمت Scenes in Build درگ کنید .

۷. صحنه ها را طوری چینش کنید که صحنه _MainScene ، صحنه شماره 0 باشد و _FarmScene صحنه شماره 1 :



۸. پنجره Build Settings را ببندید .

۹. پروژه را ذخیره کنید .

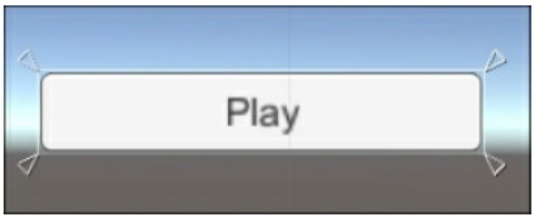
حالا یک دکمه به این صحنه جدید اضافه می کنیم تا هنگامی که کلیک شود ، صحنه FarmScene_ بارگذاری گردد .

۱۰. منو GameObject | UI | Button را انتخاب کنید . این کار یک canvas همراه با یک دکمه و یک شئ EventSystem اضافه می کند .

۱۱. در نمای Hierarchy ، دکمه را باز کنید تا مولفه text را ببینید . همه دکمه ها یک مولفه text دارند :

۱۲. در نمای Hierarchy ، دکمه را به Play_Button تغییرنام دهید .

۱۳. در نمای Inspector ، متن مولفه text را به Play تغییر دهید . حالا باید در وسط canvas یک دکمه Play داشته باشید :



۱۴. برای ایجاد یک اسکریپت جدید ، منو Assets | Create | C# Script را انتخاب کنید . هنگامی که بازیکن دکمه play را کلیک کند ، این اسکریپت اجرا خواهد شد .

۱۵. این اسکریپت جدید را به LoadSceneOnClick تغییر نام دهید .

۱۶. به منظور ویرایش این اسکریپت در MonoDevelop ، روی آن دوبار کلیک کنید .

۱۷. کدهای زیر را وارد کنید :

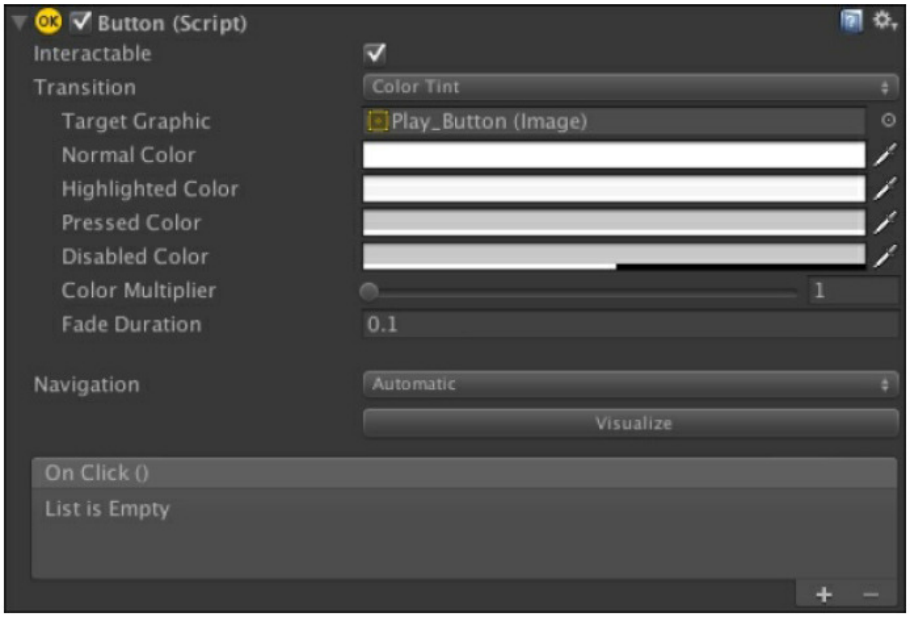
```
using UnityEngine;
using System.Collections;

public class LoadSceneOnClick : MonoBehaviour {

      public void LoadScene(int scene)
      {
            Application.LoadLevel (scene);
      }

}
```

۱۸. اسکریپت را ذخیره کرده و به یونیتی برگردید .

۱۹. این اسکریپت را به canvas درگ کنید .

۲۰. در نمای Hierarchy ، گزینه Play_Button را انتخاب کنید .

۲۱. در نمای Inspector ، مثل تصویر زیر ، در پائین کامپوننت Button (Script) ، تابع On Click() را می بینید :

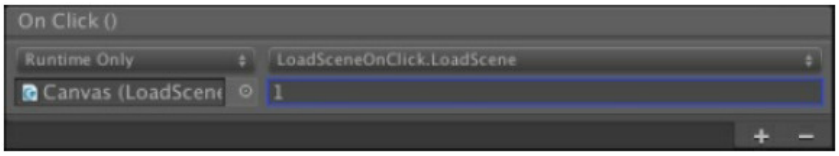


۲۲. روی علامت + کلیک کنید تا یک تابع به لیست On Click() اضافه کنیم .

۲۳. از نمای Hierarchy ، قسمت Canvas را به ناحیه None (Object) نمای Inspector درست زیر برچسب On Click () ، درگ کنید .

۲۴. با استفاده از منو No Function در ناحیه On Click () ، گزینه LoadSceneOnClick | LoadScene (int) را انتخاب کنید .

۲۵. آخرین کار ، تغییر پارامتری است که به تابع LoadScene (int) پاس می دهیم . در قسمت On Click () ، به شکل زیر ، پارامتر را از 0 به 1 تغییر دهید :



۲۶. صحنه و پروژه را ذخیره کنید .

حالا هنگامی که بازی را play کنید ، چند تغییر خواهید دید . اولا یک صحنه جدید همراه با یک دکمه Play نمایش داده می شود . هنگامی که کاربر این دکمه را کلیک می کند ، صحنه مزرعه کشاورز بارگذاری شده و بازی به صورت عادی قابل اجرا می شود .

می توانیم با اضافه کردن دکمه خروج از بازی ، منو را کامل تر کنیم . همچنین با اضافه کردن یک تصویر پس زمینه ، منو را زیباتر کنید .

خلاصه

در این فصل ، در خصوص سیستم UI موجود در Unity 5 آموزش دیدید . با اهمیت بودن GUI ها در بازی آشنا شدید . در زمینه ایجاد HUD و mini-map تجربه کسب کردید . همچنین در خصوص ایجاد یک دکمه و اسکریپت نویسی آن به منظور بارگذاری یک صحنه آشنا شدید .

در فصل بعد ، افکت های صوتی را به بازی اضافه می کنیم و با اضافه کردن سایه ها ، باعث واقعی تر شدن بازی می شویم . همچنین چند جلوه ویژه نورپردازی را نیز اضافه خواهیم کرد . در انتها نیز راه های بهینه سازی بازی را از طریق تنظیمات رندرینگ و بهینه سازی کدها ، بررسی خواهیم کرد .

فصل ۷ : بهینه سازی بازی

ما بازیمان را کامل کرده ایم و می توانیم بدون هیچ خطائی آن را اجرا کنیم . می توانید پروژه نهائی این بازی را از صفحه وب کتاب دانلود کنید .
در این فصل ، نحوه اضافه کردن جلوه های صوتی و تصویری مثل افکت های صوتی ، سایه ها و افکت های نورپردازی را به شما نشان می دهیم .

بعد از مطالعه این فصل :

قادر به اضافه کردن افکت های صوتی خواهید بود .
قادر به اضافه کردن سایه به منابع بازی خواهید بود .
قادر به ایجاد افکت های نورپردازی خواهید بود .
نحوه ویرایش دوربین ها به منظور تغییر تنظیمات رندرینگ را می فهمید .
قادر به بهینه سازی اسکریپت ها خواهید بود .

صدا و تصویر

اجازه دهید یک آزمایش سریع را انجام دهیم . پشت کامپیوتر ، کنسول بازی یا موبایلتان بنشینید و بازی مورد علاقه تان را اجرا کنید . ابتدا صدا را قطع کنید . حالا تنظیمات ویدئویی را طوری تغییر دهید که نور بازی کم شود یا اگر ممکن است به صورت سیاه و سفید شود . حالا بازی را اجرا کنید . می بینید که بازی مثل قبل جذاب نیست . حالا دوباره تنظیمات بازی را به شکل اول برگردانید . با اجرای مجدد بازی متوجه تفاوت های بودن و نبودن جلوه های صوتی و تصویری خواهید شد !
می بینید که جلوه های صوتی تصویری چقدر برای بازی ها مهم هستند . یک انفجار بدون صدا را تصور کنید ! در این قسمت به اجرای افکت های صوتی تصویری در یونیتی می پردازیم .

امکانات صوتی یونیتی

یونیتی پشتیبانی بزرگی برای پخش صدا در بازی دارد . قبل از پخش صدا در بازیمان ، چند مفهوم ابتدائی را باید بفهمید . اولا ، صداها از یک فایل منبع سرچشمه می گیرند . یونیتی از چندین فرمت فایل صوتی شامل AIFF ، MPEG1 ، MPEG2 ، MPEG3 ، OGG و WAV پشتیبانی می کند . یک فرمت لزوما بهتر از بقیه نیست . چیزی که مهم است این است که آیا فایل به شکل اصلی است یا به صورت فشرده شده .
فایل های اصلی آنهائی هستند که بدون کدگذاری فشرده شده و متراکم ، در پروژه بازی شما ذخیره می شوند .

فایل های فشرده شده ، فضای ذخیره کمتری اشغال می کنند اما در زمان اجرا ، نیاز به بازگشائی و خارج شدن از حالت فشرده دارند که بسته به سخت افزاری که استفاده می کنید ممکن است قدرت پردازش زیادی را مصرف کرده و باعث شوند بازی شما تاخیر در اجرا (lag) را تجربه کند .

Audio listener

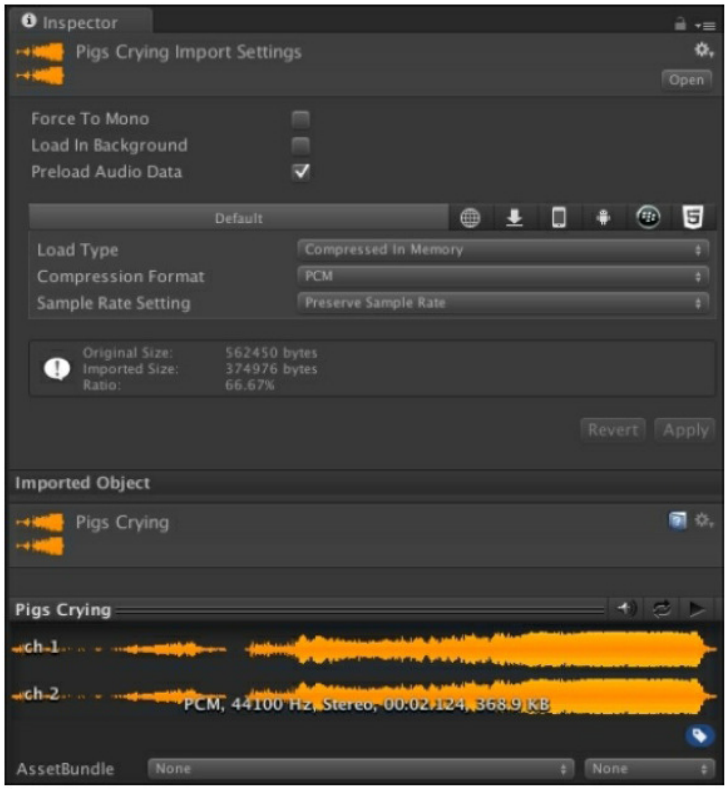
یونیتی ، Audio Listener (شنونده صدا) را به عنوان یکی اجزای دوربین اصلی پروژه ، ایجاد می کند . هنگامی که در نمای Hierarchy ، دوربین اصلی را انتخاب می کنید ؛ می بینید که در نمای Inspector ، گزینه Audio Listener لیست شده است :

Audio listener مولفه ای ساده بدون هیچ گونه مشخصاتی است . معمولا در یک زمان ، شما فقط یک Audio listener دارید . برای بازی ما ، audio listener را به دوربین اصلی الصاق می کنیم که همان طور که به یاد دارید ؛ ما دوربین اصلی را به شئ کنترل کننده سوم شخص متصل کردیم . به این شکل ، کاربر می تواند صدا را از دستگاه های خروجی صوتی مثل بلندگوهای کامپیوتر بشنود .

کلیپ های صوتی

در یونیتی ، ما به فایل منبع صوتی ، به عنوان یک نمونه از کلیپ صوتی نگاه می کنیم . کلیپ های صوتی ، انواع مختلفی دارند .

ورژن نهائی بازی Little Farmer Colt ما یک منبع صوتی Pigs Crying (صدای نعره بچه گراز) دارد . فایل منبع به فرمت WAV است . هنگامی که در نمای Inspector دیده شود ، چند تنظیم قابل تغییر وجود دارد :



توجه

یونیتی mono و stereo را پشتیبانی می کند . صدای mono هنگامی است که یک کانال صدا مورد استفاده قرار گرفته باشد در حالیکه stereo دو کانال دارد . با صدای mono ، یک صدا به هر دو بلندگوی چپ و راست فرستاده می شود . با صدای Stereo ، در یک زمان کانال های متفاوتی به هر بلندگو فرستاده می شود .

منبع صوتی

یکی از چیزهائی که باید انجام دهیم ، اضافه کردن یک منبع صوتی به دوربین اصلی است . این کار را با مراحل زیر انجام می دهیم :

۱. در نمای Hierarchy ، گزینه Main Camera را انتخاب کنید .

۲. در نمای Inspector ، دکمه Add Component را کلیک کنید .

۳. Audio | Audio Source را انتخاب کنید . این کار به شکل زیر یک Audio Source به دوربین اصلی اضافه می کند :

۴. سپس فایل صوتی Pigs Crying را به فیلد AudioClip مربوط به Audio Source در نمای Inspector درگ کنید .

۵. تیک گزینه Play on Awake را بزنید . این کار باعث می شود به محض اینکه دوربین اصلی نمونه سازی شود ، کلیپ صوتی نیز پخش شود .

۶. تیک گزینه Loop را بزنید . این کار باعث می شود کلیپ صوتی دائما پخش شود .

حالا می توانید بازی را اجرا کرده و صدای بچه گراز را بشنوید .

یکی از تنظیمات مهم در نمای Inspector ، گزینه priority (اولویت) منبع صوتی است . به صورت پیش فرض ، این گزینه روی 128 تنظیم شده است . هنگامی که در یونیتی چندین صدا برای پخش شدن وجود داشته باشد ، از اولویت هرکدام استفاده شده و مشخص می شود که کدام صدا پخش شود . عدد کوچکتر ، اولویت بالاتری خواهد داشت :

پیاده سازی افکت های صوتی

علاوه بر الصاق صداها به یک منبع صوتی و انتخاب Play on Awake و Loop ، می توانیم از طریق اسکریپت ها نیز صداها را پیاده سازی نمائیم . برای این کار تیک گزینه های Play on Awake و Loop را بردارید . سپس مراحل زیر را انجام دهید :

۱. اسکریپت Take.cs را به شکل زیر تغییر دهید . ۴ خط کد جدید اضافه شده که با توضیح // new line آمده اند :

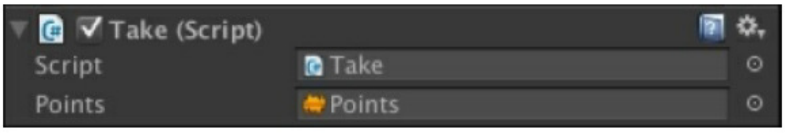
```
using UnityEngine;
using System.Collections;
```

```
public class Take : MonoBehaviour {

	Animator anim;
	int takeHash = Animator.StringToHash ("Take");

	public AudioClip points; // new line
	private AudioSource source; // new line

	void Start () {
		anim = GetComponent<Animator> ();
		source = GetComponent<AudioSource> (); // new line
	}

	void Update () {
		if (Input.GetKeyDown (KeyCode.T)) {
			anim.Play (takeHash);
			GameData.water = GameData.water +1;
			source.PlayOneShot (points); // new line
		}

	}
}
```

کدهای جدید ، هربار که بازیکن کلید T را فشار می دهد ، یک افکت صوتی اضافه می کنند .

۲. سپس ، در نمای Hierarchy ، گزینه ThirdPersonController را انتخاب کنید .

۳. در نمـــای Inspector ، فایـــل صـــوتی Points را از نمـــای Project بـــه فیلـــد Points مربـــوط بـــه ناحیه Take (Script) در نمای Inspector ، درگ کنید :



۴. درحالیکه ThirdPersonController در نمای Hierarchy انتخاب شده باشد ، در نمای Inspector ، دکمه Add Component را کلیک کنید .

۵. Audio | Audio Source را انتخاب کنید .

۶. تیک گزینه Play on Awake منبع صوتی را بردارید . این کار باعث می شود هنگامی که بازی بارگذاری می شود ، کلیپ به صورت اتوماتیک پخش نشود .

حالا هربار که انیمیشن take کاراکتر اصلی پخش می شود ، کلیپ صوتی "points" نیز پخش خواهد شد .

افکت های تصویری

در این بخش مثال های زیر را بررسی می کنیم :

گوی نورانی

سایه

دنباله

گوی نورانی

تابلو موجود روی انبار Colt را ببینید :

اجازه دهید یک گوی نورانی به آن اضافه کنیم :

۱. در نمای Project ، گزینه Materials را انتخاب کنید .

۲. منو Assets | Create | Material را انتخاب کنید .

۳. متریال جدید را Mat_Yellow بنامید .

۴. در حالیکه Mat_Yellow در نمای Project انتخاب شده باشد ، رنگ را از سفید به زرد تغییر دهید .

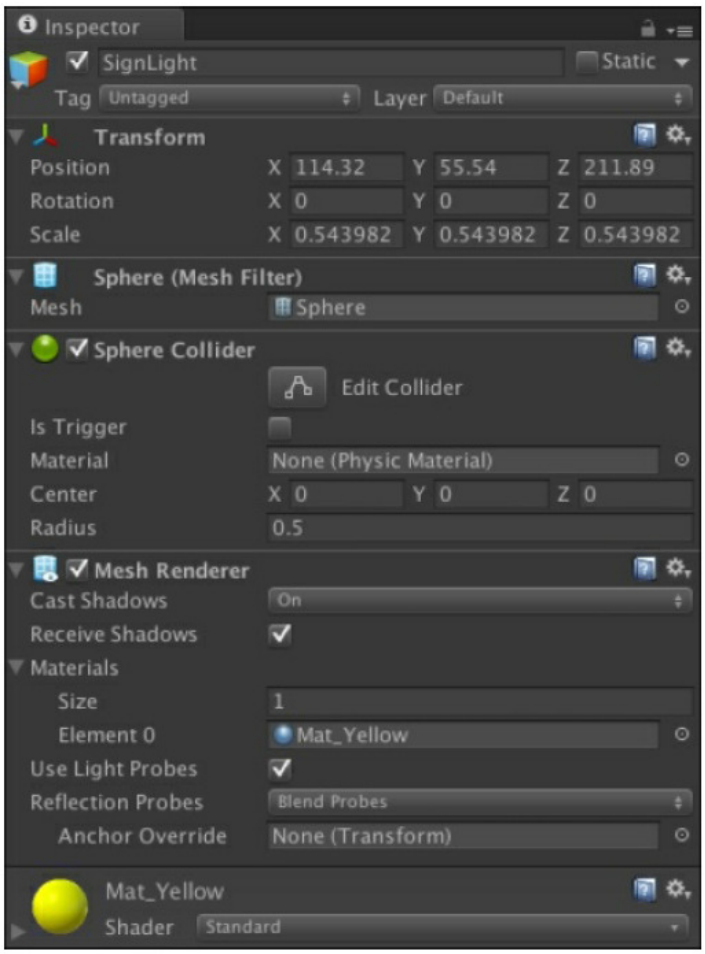
۵. منو GameObject | 3D Object | Sphere را انتخاب کنید .

۶. کره را SignLight بنامید .

۷. کره SignLight را به بالای تابلو انبار Colt ببرید و همسطح در انبار ، قرار دهید .

۸. کره را تغییر اندازه دهید تا روی در به شکل مناسب ظاهر شود .

۹. در حالیکه کره SignLight در نمای Hierarchy انتخاب شده باشد ، متریال Mat_Yellow را به فیلد Mesh Renderer | Materials | Element 0 موجود در نمای Inspector درگ کنید :



حالا گوی ایجاد شده است و آماده ایجاد نور هستیم .

۱۰. منو GameObject | Light | Spotlight را انتخاب کنید .

۱۱. در نمای Hierarchy ، قسمت Spotlight را درگ کنید تا به شکل زیر ، زیرمجموعه قسمت SignLight شود :

۱۲. در نمای Scene ، قسمت Spotlight را حرکت دهید تا در داخل کره ظاهر شود .

۱۳. Spotlight را بچرخانید تا درخشش نور به طور مناسب روی تابلو انجام شود .

۱۴. درحالیکه Spotlight انتخاب شده است ، در نمای Inspector ، گزینه Range را به 3 تغییر دهید .

۱۵. در نمای Inspector ، رنگ را به همان رنگ زردی که برای متریال Mat_Yellow استفاده کردید ، تغییر دهید .

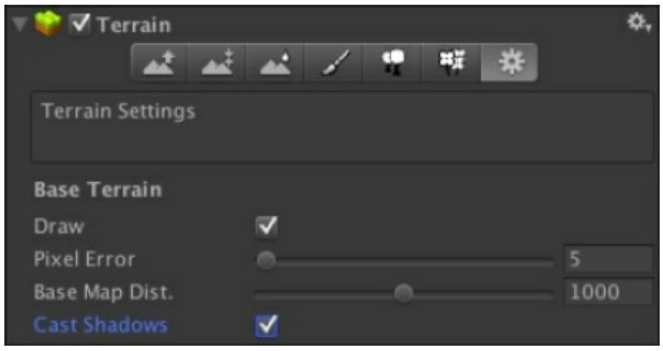
۱۶. تیک گزینه Draw Halo را بزنید .

۱۷. صحنه و پروژه را ذخیره کنید .

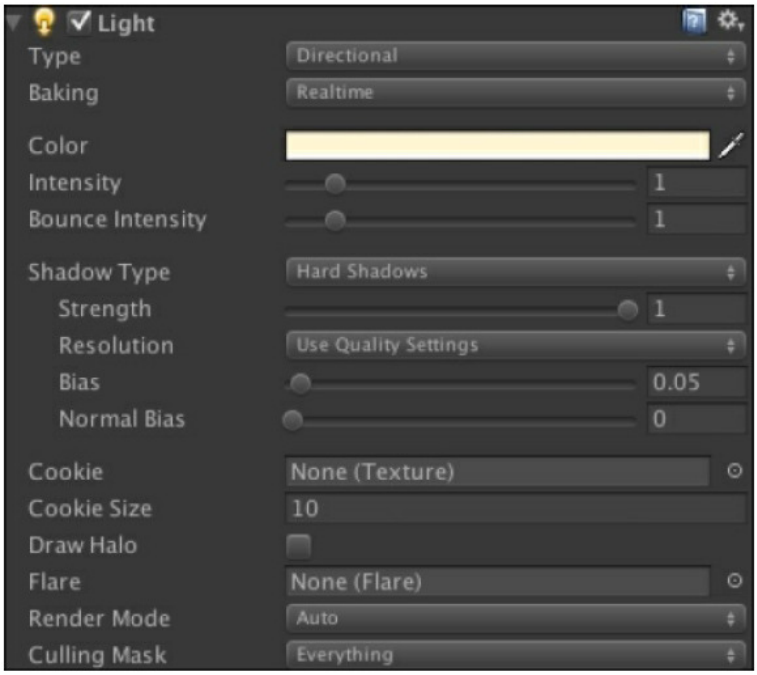
۱۸. با play کردن بازی ، این افکت را خواهید دید :

سایه

حتما توجه کرده اید که همه درخت های بازی ما سایه دارند اما حیوانات ، بازیکن و انبارها دارای سایه نیستند . در Terrain به صورت پیش فرض تنظیمات Cast Shadows (سایه) فعال است :

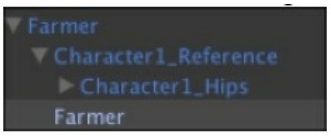


درخت ها نیز سایه دارند . این سایه ها از نور سمتی (directional) که در صحنه داریم ، ناشی می شوند . به راحتی می توانید در نمای Inspector این نور را غیرفعال کنید :

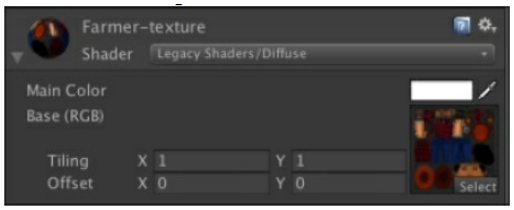


اما اگر بخواهیم دو کشاورز و حیوانات مزرعه نیز سایه داشته باشند ، چطور ؟ با انجام دو مرحله می توانیم به هر شئ بازی ، سایه را اضافه کنیم :

۱. به شکل زیر ، در نمای Hierarchy ، گزینه Farmer | Character1_Reference | Farmer را انتخاب کنید :



۲. در قسمت Farmer-texture نمای Inspector ، گزینه Shader | Legacy Shaders | Diffuse را انتخاب کنید :



حالا می بینید که Pa Poo ، کشاورز مسن ما ، سایه دارد . انیمیشن هایش را تماشا کنید ! می بینید که سایه متحرک است :

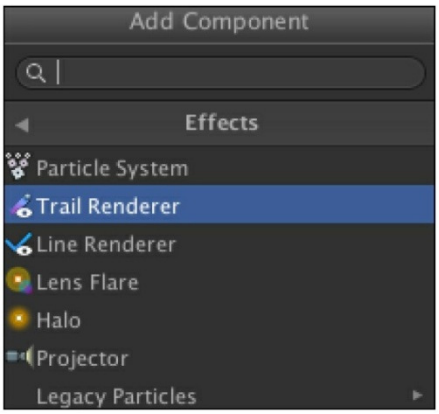
دنباله

در یونیتی می توانیم افکت تصویری ایجاد کنیم که در پشت سر یک شئ در زمانیکه در حال حرکت است ، یک دنباله ایجاد شود . مثل یک کشتی که در دنبالش موج به جای می گذارد ، یک دنباله نشان می دهد که شئ کجا بوده است ، در چه جهتی در حال حرکت است و چه سرعتی دارد . ما در بازی خود ، به این خصوصیت نیازی نداریم ، اما شما می توانید از آن در بازی های دیگرتان به سادگی استفاده کنید . پس ما آن را همراه با کارکتر Colt نشان می دهیم . برای انجام آن ، مراحل زیر را دنبال کنید :

۱. در نمای Hierarchy ، گزینه ThirdPersonController را انتخاب کنید .

۲. در نمای Inspector ، دکمه Add Component را کلیک کنید .

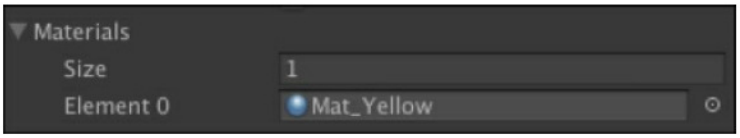
۳. به شکل زیر ، Effects | Trail Renderer را انتخاب کنید :



۴. در نمای Inspector ، گزینه Cast Shadows را on قرار دهید . این کار باعث می شود ، دنباله رندر شده دارای سایه باشد . سایه ها بسته به اندازه دنباله و مکان نسبی آنها نسبت به منبع نور هستند .

۵. دقت کنید که گزینه Receive Shadows انتخاب شده باشد .

۶. متریال Mat_Yellow را به فیلد Materials | Element 0 درگ کنید تا متریال زرد نورانی به دنباله اعمال شود :



۷. حالا با اجرای بازی این افکت تصویری را ببینید .

می بینید که Colt در پشت سرش یک دنباله زرد رنگ به جا می گذارد . در تصویر زیر ، قسمتی از دنباله که قوس دار شده ، نشان می دهد که Colt پریده است :

چند تنظیمات دیگر مربوط به Trail Renderer وجود دارد که باید دقت کنید :

Time : در اینجا اندازه دنباله را مشخص می کنید . مقادیر به ثانیه هستند .

Start width : عرض دنباله در نقطه شروع .

End width : عرض دنباله در نقطه پایان .

Autodestruct : بعد از اینکه شئ به مدت زمانی که در قسمت Time مشخص کرده بودید به حالت استراحت رفت ، به صورت اتوماتیک نابود شود .

اشاره

اگر گزینه Autodestruct مربوط به Trail Renderer را به کارگیری کنید ، شئ شما بعد از گذشتن تعداد ثانیه هائی که در قسمت Time مشخص کرده اید ، نابود خواهد شد . در خصوص کاراکتر Colt ، خواهید دید که بعد از اینکه بیکار می شود ، به صورت اتوماتیک نابود می شود . این خصوصیت برای چیزهائی مثل جرقه ها و شهاب سنگ ها عالی است اما برای کاراکتر اصلی ما مناسب نیست .

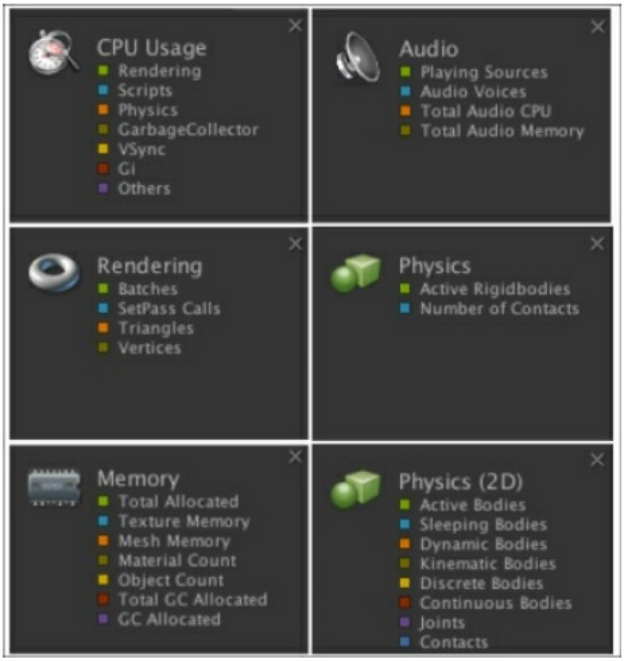
برای غیر فعال کردن این افکت ، در نمای Inspector ، تیک گزینه Trail Renderer را بردارید :

تنظیمات رندرینگ

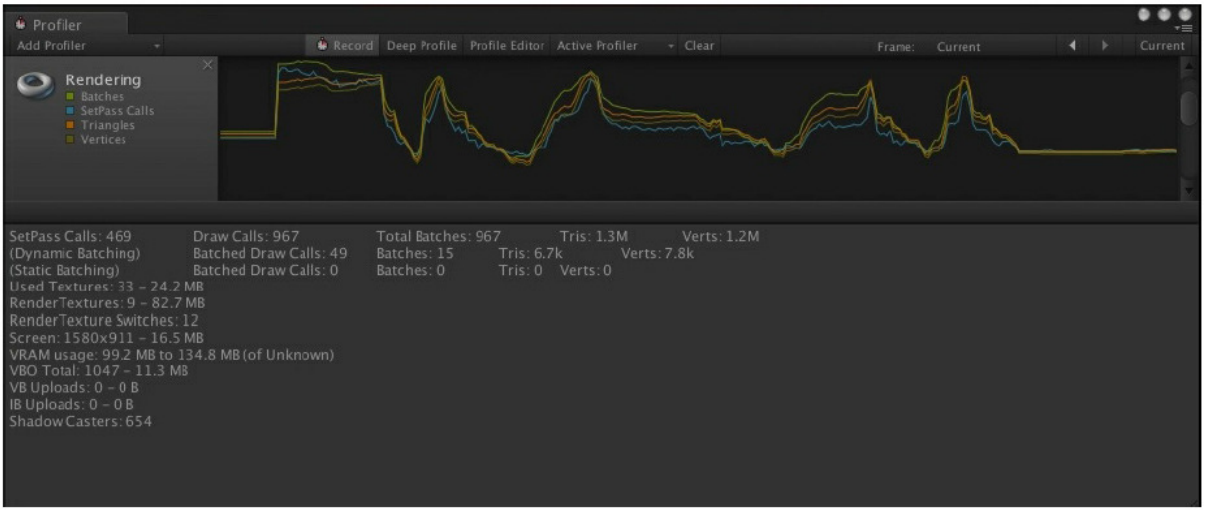
همانطور که می دانید ، رندرینگ (rendering) ، عمل تولید یک تصویر سه بعدی از تصاویر ، رنگ ها و سایه هاست . بسته به بازی شما ، این می تواند کار زیادی را بر دوش موتور یونیتی تحمیل کند . با انتخاب منو Profiler | Window ، می توان روی آن متمرکز شد . با انتخاب این منو ، پنجره Profiler باز می شود .

پنجره Profiler

پنجره Profiler دارای قسمت های CPU usage ، Rendering ، Memory ، Audio ، Physics و Physics 2D است . بازدید این قسمت ها به ما در تعیین نحوه عملکرد بازی و جاهائی که پتانسیل بهینه شدن را دارند ، کمک می کند . ما روی قسمت Rendering متمرکز می شویم :



هنگامی که بازی در حالت play باشد و شروع به گردش در دنیای داخل بازی کنیم ، به شکل زیر قسمت Rendering پنجره Profiler شروع به جمع آوری اطلاعات می کند :



علاوه بر نمودار رنگ بندی شده که در قسمت بالای پنجره مشاهده می کنید ، اطلاعات مخصوص رندرینگ نیز در قسمت پائین نشان داده می شود .

بهینه سازی رندرینگ

چند چیز وجود دارد که می توان انجام داد تا پردازش بیش از حد مربوط به رندرینگ در زمان اجرای بازی را کاهش داد . چند مشکل که معمولا در زمان اجرای بازی ها وجود دارد عبارت است از :

اندازه فایل های بازی : اگر بازی را به صورت دیجیتالی عرضه می کنید یا آن را در فروشگاه های برنامه های موبایل (app store) منتشر می کنید که دارای محدودیت اندازه فایل هستند ، اندازه فایل های بازی مشکل ساز می شود .

تاخیر (lag) : مشکل تاخیر در اجرای بازی زمانی مطرح می شود که فشار بیش از حد بر روی امکانات پردازش سخت افزاری وارد می شود یا با توجه به محدودیت شبکه کاربر ، گرفتار محدویت پهنای باند می شوید .

محدودیت های سخت افزاری : دستگاه های موبایل امکانات پردازشی خوبی دارند اما هنوز در مقایسه با کنسول های بازی و کامپیوترهای شخصی ضعیف هستند . مثلا هنگام ساختن بازی برای دستگا های موبایل ، "شفافیت آلفا" و "تعداد فراخوان های ترسیم" موضوعات مهمی هستند که باید رسیدگی شوند .

محدودیت های مهم

چند تکنیک کمکی برای کاهش زمان و پردازش بیش از حد رندرینگ :

۱. دوری یا محدود کردن استفاده از مه در بازی

۲. محدود کردن استفاده از افکت های ذره ای متراکم

۳. محدود کردن سایه ها

۴. استفاده از light maps به جای نورپردازی دینامیک

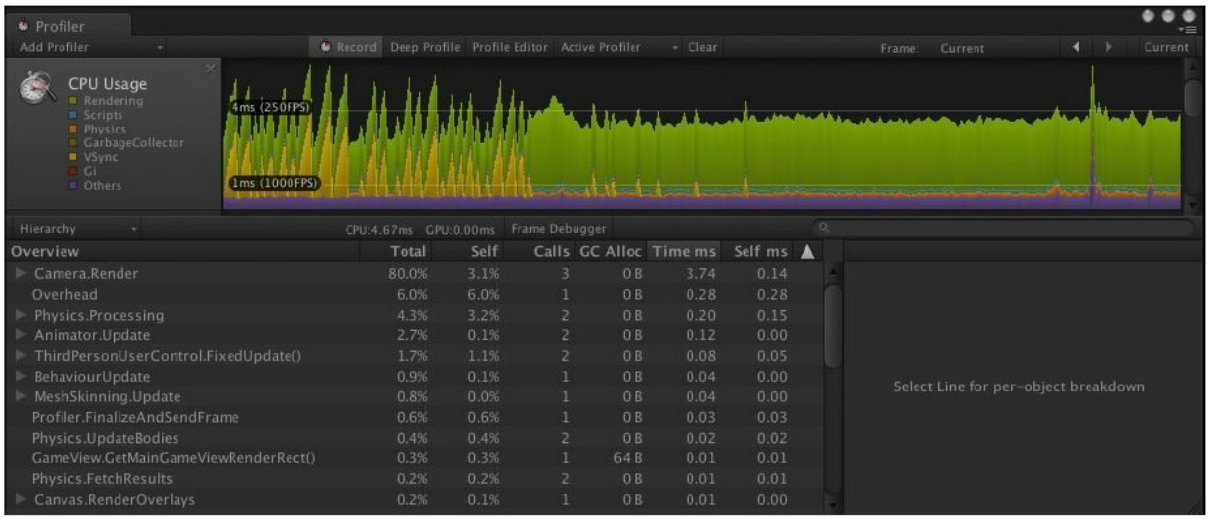
تصاویر آماده

می توانیم از تصاویر آماده و بهینه شده برای شبیه سازی سایه ها و عمق استفاده کنیم . با استفاده از تصاویری مثل آنچه در زیر برای کاراکتر Colt استفاده شده است ، می توان خصوصیات ویژه را در داخل کار گرافیکی انجام داد . این کار نیاز به سایه ها و افکت های نورپردازی را کاهش می دهد :

نرم افزارهائی مثل فوتوشاپ می توانند نورپردازی را به اشیاء و مدل های سه بعدی اعمال کنند . یک تکنیک این است که افکت های نورپردازی را در یک لایه جداگانه ایجاد کرد و سپس از گرافیک نهائی ، همراه با همه لایه هایش ، خروجی گرفت .

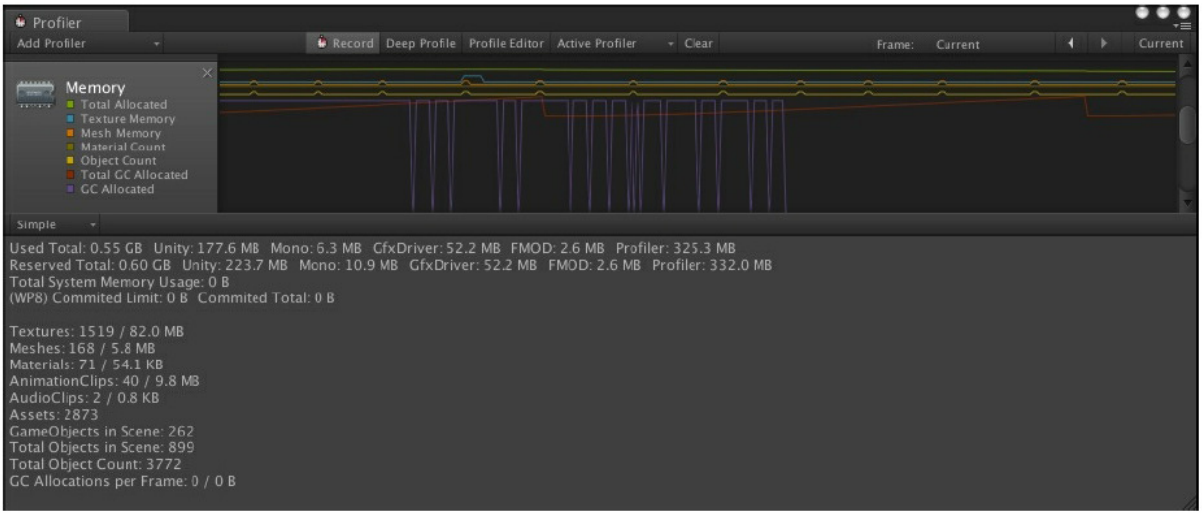
بهینه سازی اسکریپت ها

ما می خواهیم اسکریپت هایمان در زمینه استفاده از حافظه و پردازش CPU بهینه باشند . شیوه خوبی که برای بهینه سازی اسکریپت ها وجود دارد ، اجرای بازی همراه با پنجره Profiler و انتخاب قسمت CPU Usage است :



با این پنجره ، می توانید پردازش هائی که بیشتر توان CPU را گرفته اند ، ببینید . قسمت Overview ، جزئیاتی از مقدار پردازش CPU که به هر شئ و پردازش ، اختصاص یافته را نشان می دهد .

قسمت دیگر پنجره Profiler ، گزینه Memory است که اطلاعات آن به شما مقدار حافظه تخصیص یافته و مقدار حافظه سیستم مورد استفاده را می گوید و تعداد Texture ، Mesh ، Material ، AnimationClip ، AudioClip ، Asset و GameObject را نشان می دهد :



هنگامی که فهمیدیم مشکل در کجاست یا کجا پتانسیل ایجاد مشکل را دارد ، می توان چند کار برای بهینه کردن اسکریپت ها انجام داد :

۱. با استفاده از `Destroy(theGameObject)` ، اشیائی را که دیگر نیازی به آنها نیست ، تخریب کنید . البته اگر قصد استفاده مجدد از شئ را داشته باشید ، دوبار تخریب کردن آن پردازش اضافه ای را می خواهد که گاهی اوقات باعث ایجاد lag (تاخیر) در بازی می شود .

۲. اگر شئ ای داریم که باید وجود داشته باشد اما فعلا مقداری ندارد ؛ با استفاده از `theGameObject = null` می توان مقدارش را با `null` نصب کرد .

البته تکنیک های پیشرفته تری برای بهینه سازی اسکریپت های یونیتی وجود دارد که مطالعه آنها را بر عهده خودتان می گذاریم !!!

خلاصه

در این فصل اهمیت استفاده از جلوه های صوتی تصویری را دیدیم و امکانات یونیتی برای مدیریت صداها را بررسی کردیم . ما از این دانش استفاده کرده و دو افکت صوتی را به بازی اضافه کردیم : صدای "pigs crying" برای بچه گراز و صدای "points" که زمانیکه کاراکتر بازی آیتمی را برمی داشت ، پخش می شد . همچنین به جلوه های ویژه گرافیکی شامل گوی نورانی ، سایه و دنباله نیز نگاهی داشتیم . همچنین به نحوه بهینه سازی بازی با استفاده از تنظیمات رندرینگ پرداختیم . در ضمن پنجره Profiler یونیتی را به شما معرفی کرده و نحوه خواندن اطلاعات مربوط به رندرینگ ، میزان مصرف CPU و حافظه را نشان دادیم . در خصوص بهینه سازی رندرینگ و اسکریپت ها نیز آموزش دیدید .

در فصل آخر ، به شیوه های بهبود بازی Little Farmer Colt ، گردش کار یونیتی ، نحوه مقیاس بندی و مدیریت پروژه می پردازیم .

فصل ۸ : بهبود بازی و مدیریت پروژه

حالا که بازی ما یعنی Little Farmer Colt کاملا آماده شده ، به شیوه های بهبود این بازی می پردازیم . در بخش دوم این فصل ، به مسائل مربوط به مدیریت پروژه مثل گردش کار و مقایس بندی پروژه ها می پردازیم . در بخش سوم ، موضوعاتی مثل مستقل بودن از پلتفرم ، پلاگین ها و مالکیت محتوا را بررسی می کنیم .

در بخش پایانی فصل ، به مباحث پیشرفته ای مثل سیستم های ذره ای ، سیستم کنترل موجودی و سیستم های گفتگو محور ، می پردازیم .

بعد از مطالعه این فصل :

با راه های بهبود بازی Little Farmer Colt آشنا می شوید .

با گردش کار پروژه های یونیتی آشنا می شوید .

با نحوه مقیاس بندی پروژه ها آشنا می شوید .

با مسائل مربوط به مستقل بودن از پلتفرم آشنا می شوید .

با اهمیت مالکیت محتوا ، آشنا می شوید .

با چند مبحث پیشرفته آشنا خواهید شد .

بهبود بازی

می توانیم به بازیمان افتخار کنیم اما بعد از مدتی که بازیکن ، بازی ما را اجرا کند ، دیگر چیزی برای به هیجان آمدن بازیکن وجود نخواهد داشت . در اینجا پیشنهاداتی برای بهتر شدن بازی و لذت بخش کردن آن برای بازیکنان ، مطرح می کنیم .

صدا

در مجموع ، ما دو افکت صوتی به بازی اضافه کردیم : صدای نعره بچه گراز و صدائی که در زمانیکه Colt ، آب یا غله به دست می آورد ، پخش می شود .

چند پیشنهاد برای اضافه کردن صداهای بیشتر به بازی :

موسیقی پس زمینه : یک موسیقی پس زمینه زیبا ، می تواند بازی را جذاب تر کند .

افکت های صوتی حیوانات : هر حیوان می تواند سه صدا داشته باشد : حالت بیکاری ، حالت غذاخوردن و حالت آب خوردن . افکت صوتی "pigs crying" که ما به بازی اضافه کردیم مخصوص حالت بیکاری بچه گراز و گراز بالغ است . اضافه کردن افکت های صوتی بیشتر ، باعث واقعی تر شدن بازی شده و آن را سرگرم کننده می کند .

کشاورز مسن : در حال حاضر ، کشاورز مسن چندین انیمیشن دارد . یکی از این انیمیشن ها "صحبت کردن" است . می توان برای این انیمیشن ، صدائی را اضافه کرد .

آب : بازی ما دارای رودخانه بزرگی است . Colt می تواند به داخل آب برود و آب جمع آوری کند . این به ما فرصت اضافه کردن دو افکت صوتی را می دهد : صدای ریزش آب درهنگامی که Colt به داخل آب می رود و صدای برداشتن آب توسط Colt.

Colt : می توان چند افکت صوتی را به کاراکتر اصلی بازی اضافه کرد . هنگامی که Colt می پرد ، یک افکت صوتی می تواند پخش شود یا هنگامی که پاهایش بعد از پرش به زمین می رسد ، یک صدای دیگر پخش شود . یا مثلا بازیکن پس از چند ثانیه که دوید ، صدای نفس نفس زدنش به گوش برسد .

افکت های تصویری

غیر از نورپردازی و سایه ها ، بازی ما افکت تصویری دیگری ندارد . چند ایده برای اضافه کردن این افکت ها به شکل زیر است :

آب : هنگامی که Colt به داخل آب می رود ، اگر ریزش آب دیده شود ، عالی خواهد بود . همچنین حباب های هوا درزمان رفتن به داخل آب نیز افکت زیبائی خواهد بود .

آئینه : ما بازی را به صورت اول شخص ساخته ایم . این یعنی بازیکن هرگز کاراکتر Colt را نمی بیند . برای دیده شدن Colt ، می توان یک آئینه به دیوار انبار اضافه کرد . برای اضافه کردن آئینه ، مراحل زیر را دنبال کنید :

۱. با انتخاب منو GameObject | 3D Object | Cube یک مکعب ایجاد کنید .

۲. با استفاده از ابزارهای تغییر شکل ، مکعب را طوری تغییر دهید که مسطح شده و برای نصب روی دیوار انبار به عنوان آئینه مناسب شود .

۳. مکعب را روی دیوار انبار ببرید .

۴. درحالیکه در نمای Hierarchy ، مکعب انتخاب شده باشد ، در نمای Inspector دکمه Add Component را کلیک کنید . سپس گزینه Reflection Probe | Rendering را انتخاب کنید .

۵. در نمای Inspector ، گزینه Type | Realtime را انتخاب کنید تا در طول بازی ، بازتاب به صورت دینامیک انجام شود .

۶. در نمای Inspector ، گزینه Type | Refresh Mode | Every frame را انتخاب کنید . این کار باعث می شود که در طول بازی ، آینه دائما به روز شود .

۷. مکعب را به MirrorCube تغییر نام دهید .

۸. MirrorCube را از نمای Hierarchy به نمای Project درگ کنید .

۹. در نمای Inspector ، تغییرات زیر را روی Mesh Renderer اعمال کنید :

Reflection Probes را به Blend Probes تغییر دهید .

MirrorCube را از نمای Project به Anchor Override درگ کنید :

می توانید با اصلاح تنظیمات Mesh Renderer و Reflection Probe آئینه را آنطوری که می خواهید ، تغییر دهید .

عملکرد

می توانید هر عملکردی را که می خواهید به این بازی اضافه کنید . برای این کار نیاز به منابع و اسکریپت نویسی دارید . پیشنهادهای زیر می تواند برای بازی مناسب باشد :

آب : در حال حاضر وقتی Colt به داخل آب می رود ، هیچ اتفاقی نمی افتد . ما قبلا افکت های صوتی تصویری را در این خصوص شرح دادیم . در زمینه عملکرد ، اگر بازیکن بتواند در عمق دریاچه راه برود و سپس بالا بیاید ، عالی می شود . برای این کار ، بازیکن باید مقدار مشخصی اکسیژن داشته باشد که توسط یک نمایشگر روی صفحه نشان داده شود . این باعث می شود بازیکن بالاجبار به سطح زمین برگردد . اگر بازیکن به سرعت به سطح زمین برنگردد ، می میرد .

غله : می توانید اسکریپتی به بازی اضافه کنید که هر خوشه غله فقط یکبار بتواند برداشته شود . هربار که بازیکن عمل "برداشتن" را انجام می دهد ، خوشه غله برداشته شده ،باید ناپدید شود (تخریب شئ بازی)

کشاورز مسن : یک گزینه خوب برای این کاراکتر ، پیاده سازی یک سیستم گفتگو با کارکتر اصلی است تا بتوانند با هم تعامل داشته باشند . حتی می توانید بازی را طوری اصلاح کنید که بازیکن بتواند هر دو کشاورز مسن و Colt را کنترل کند . همچنین می توانید بازی را طوری ویرایش کنید که دو بازیکن بتوانند در یک زمان از طریق شبکه یا سرور وب ، بازی کنند . بازیکن اول باید Colt را کنترل کند و دیگری ، کشاورز مسن را .

مراحل

بازی ما فقط یک مرحله دارد که هیچ وقت تمام نمی شود . اگر بازی چندین مرحله داشته باشد ، جذاب تر می شود . هر مرحله باید یک ماموریت داشته باشد . اما این مراحل اضافه باید شامل چه چیزهائی باشند ؟ چند پیشنهاد ببینید :

هر مرحله جدید باید یک حیوان جدید مثل گاو ، اسب یا بز اضافه کند .

در هر مرحله جدید باید چیز جدیدی را جمع آوری کرد مثل شیر ، علوفه و ...

برای سخت تر شدن بازی ، تعداد هر یک از انواع حیوانات افزایش پیدا کند .

در هر مرحله ، اندازه مزرعه باید افزایش پیدا کند .

مدیریت پروژه

هنگام ساختن بازی ها در یونیتی ، باید برای مدیریت پروژه طرحی داشته باشید . اصطلاح "تیم بزرگتر ، نیاز بیشتر" در خصوص یونیتی نیز حکم می کند . همانطور که در بازی Little Farmer Colt دیدید ، تعداد منابع مورد نیاز حتی در یک پروژه کوچک نیز می تواند به سرعت افزایش پیدا کند . اگر شما یک سازنده مستقل هستید ، نیازی به نگرانی در خصوص مدیریت پروژه ندارید . اما اگر دو یا چند نفر در تیم شما حضور دارند ، توصیه می کنم گردش کار پروژه تان را مشخص کنید .

گردش کار یونیتی

هیچ نوع گردش کار کاملی برای کارکردن با یونیتی وجود ندارد . هر تیمی کارش را به صورت های مختلفی جلو می برد . بنابراین به جای پیشنهاد یک گردش کار مشخص ، چند بخش مهم هر گردش کار را ذکر می کنیم و متدهائی را برای انجام آنها پیشنهاد می کنیم .

گرافیک و انیمیشن

معمولا اعضائی از تیم که بر روی کارهای هنری و انیمیشن کار می کنند از آنهائی که اسکریپت می نویسند ، جدا و متمایز هستند . تیم هنری باید بتواند کارش را همراه با اسکریپت نویسان پیش ببرد . مثلا می توانیم بدون داشتم آیتم های گرافیکی اصلی ، سیستم کنترل موجودی های بازی (مثل مقدار مهمات اسلحه و جان بازیکن) را اسکریپت نویسی کنیم . می توان به صورت موقت از مکعب ها یا کره ها به جای گرافیک های اصلی استفاده کنیم تا اینکه تیم هنری ، مدهایشان را آماده کنند . با این شیوه ، هر دو فرایند به صورت هم زمان پیش می رود .

اسکریپت ها

اگر بیش از یک نفر روی اسکریپت نویسی کار می کند ، باید برای دسترسی آسان به اسکریپت ها ، آنها را در یک مکان مرکزی نگهداری کنید . ابزارهای مدیریت پروژه مثل JIRA و BaseCamp می توانند کمک کننده باشند .

منابع دیگر

برای هر عضو تیم مهم است که محل ذخیره منابع و قرادادِ نام گذاری آنها را بداند . استفاده از یک قرارداد نام گذاری استاندارد برای منابع ، یکی از شیوه های خوب سازماندهی است .

مقیاس بندی پروژه ها

یونیتی معروف به این است که یک موتور بازی عالی برای سازندگان مستقل و پروژه های کوچک بازی است . این تا حدی نادرست است . از یونیتی می توان برای پروژه هائی در مقیاس بزرگ نیز استفاده کرد . در این نوع پروژه ها به چند موضوع باید توجه کنید :

مدیریت فایل ها

مدیریت منابع

کنترل مراحل

ساختن راهنما

پشتیبان گیری از داده ها

کنترل ورژن

مفاهیم توسعه

در این بخش درخصوص مستقل بودن از پلتفرم ، پلاگین ها و مالکیت محتوا بحث می کنیم .

رعایت مستقل بودن از پلتفرم

می توانید بازی های multiplayer را بسازید و کاربرانی با دستگاه های اندرویدی یا iOS داشته باشید که با هم یا در مقابل هم بازی می کنند . برای انجام این کار ، می توانید از سرویسی مثل Google Play استفاده کنید .

موضوع دیگری که باید رسیدگی شود ، اندازه صفحه نمایش گوشی های موبایل است . Apple تعداد محدودی از دستگاه هائی دارد که از iOS استفاده می کند اما اندروید بر روی تعداد زیادی از اندازه های صفحه نمایش و رزولیشن های مختلف اجرا می شود . لازم نیست برای هر اندازه صفحه نمایش ، یک ورژن ویژه از بازی را ایجاد کنید .

همیشه بازیتان را روی دستگاه های مختلف ، سیستم عامل هائی با ورژن های متفاوت ، اندازه های صفحه نمایش مختلف و رزولیشن های متفاوت ؛ تست کنید . اگر سخت افزار اصلی را برای تست بازی در اختیار ندارید از شبیه سازهای نرم افزاری (emulator) استفاده کنید .

پلاگین ها

ما با استفاده از MonoDevelop ، اسکریپت هایمان را نوشتیم و بازی را اسکریپت نویسی کردیم . در خارج از یونیتی نیز می توانید نوشتن کدها را انجام دهید . اگر پلاگینی نوشته اید یا حق استفاده از یک

پلاگین را دارید ، در هرجائی از پروژه تان می توانید آن را قرار دهید . منطقی است که در نمای Project ، یک پوشه Assets | Plugins ایجاد کنید .

مالکیت محتوا

اشتباهی که اغلب سازندگان مستقل انجام می دهند ، استفاده از منابعی است که از طریق اینترنت به دست آورده اند . چند جا مثل Unity Asset Store (فروشگاه منابع یونیتی) در دسترس است . مهم است که مطمئن شوید ، مجوز لازم برای استفاده از محتوای دانلود شده در بازیتان را دارید .

اگر روی یک پروژه تجاری کار می کنید ، باید مطمئن شوید که منابع شما "حق امتیاز رایگان استفاده تجاری" را دارند . اگر مطمئن نیستید ، با تهیه کننده منبع تماس بگیرید .

اشاره

فقط به دلیل اینکه برای دانلود یک منبع ، مبلغی را پرداخت کرده اید ، دلیل نمی شود که مجوز استفاده از آن برای اهداف تجاری را دارید .

مباحث پیشرفته

در این قسمت به بررسی مختصر سیستم های ذره ای ، سیستم کنترل موجودی و سیستم های گفتگومحور می پردازیم .

سیستم های ذره ای

یونیتی توانائی شگفت انگیزی برای رندر ذرات موجود در بازی دارد . ما از ذرات برای نمایش گرافیک های متحرک مثل مه ، دود ، آتش ، جرقه ، گردو غبار و ... استفاده می کنیم . ذرات را می توان به صورت چندین تصویر خیلی کوچک در حال حرکت ، تصور کرد . آتش بازی را تصور کنید که متشکل از صدها نور کوچک در حال حرکت است . با استفاده از سیستم ذره ای یونیتی (particle system) می توانیم این نوع شبیه سازی ها را انجام دهیم .

برای پیاده سازی یک سیستم ذره ای در یونیتی ، منو GameObject | Particle System را انتخاب کنید . این باعث می شود در نمای Hierarchy ، یک شئ Particle System مستقر گردد . بررسی Particle System در نمای Inspector ، تعداد زیادی از تنظیمات را نشان می دهد که می توانند نحوه عملکرد سیستم ذره ای و رندر آن را تغییر دهند .

سیستم های کنترل موجودی

سیستم کنترل موجودی (Inventory control systems) یکی از بخش های کلیدی هر بازی است . یونیتی این را به عنوان یک کامپونت فراهم نکرده است بنابراین می توانید از یک پلاگین مجزا برای آن استفاده کنید یا اینکه خودتان آن را بنویسید . ۶ مرحله ساده برای ایجاد این نوع سیستم وجود دارد :

۱. جدولی از انواع آیتم های بازیتان فراهم کنید و خصوصیات هر آیتم را نیز مشخص کنید .

۲. آیتم ها را دسته بندی کنید .

۳. GUI را برای تعامل کاربر با سیستم کنترل موجودی ، طراحی کنید .

۴. گرافیک های مورد نیاز را آماده کنید طوری که همه آیتم ها داخل GUI شما جای شوند .

۵. سیستم کنترل موجودی را اسکریپت نویسی کنید .

۶. سیستمتان را تست کنید .

سیستم های گفتگو

هنگامی که بازی شما کاراکترهای شبه انسان NPC دارد ، سیستم گفتگو (Dialog systems) یکی از جذابیت های بازی خواهد بود . چیزی که برای این سیستم ، استاندارد است ، بودن متن ها روی صفحه همراه با پاسخ های ممکن است . احتمالا این خصوصیات را در بازی های زیادی دیده اید و با آن آشنا هستید . هنگام ایجاد این سیستم ها ، چند نکته را توجه کنید :

۱. قبل از اسکریپت نویسی ، همه گفتگوها را به صورت نمودار روی کاغذ ترسیم کنید .

۲. همیشه به بازیکن گزینه ای برای پایان دادن به گفتگو بدهید .

۳. کاربر باید همه پاسخ ها را یکجا و در یک نگاه ، ببیند .

خلاصه

در این فصل به شیوه های بهبود بازی Little Farmer Colt پرداختیم . پیشنهادات ما در خصوص بهبود افکت های صوتی تصویری ، عملکرد و مراحل بازی ارائه شد . همچنین در خصوص مسائل مربوط به گردش کار پروژه های یونیتی ، مقیاس بندی پروژه ها و مدیریت پروژه های تیمی و چندنفره توضیحاتی ارائه شد .

در زمینه موضوعات مستقل بودن از پلتفرم بازی ، استفاده از پلاگین های مجزا و مالکیت محتوا نیز نکاتی ارائه شد . در انتها نیز به چند مبحث پیشرفته شامل سیستم های ذره ای ، سیستم کنترل موجودی و سیستم گفتگو پرداختیم .

امیدورام از ساختن بازی ها با استفاده از یونیتی ، لذت ببرید !