دانشگاه گلستان

دانشکده فنی و مهندسی علی آباد کتول

پروژه درس شیوه ارائه

عنوان:

موتور بازی ساز یونیتی

گردآورنده:

میلاد کرامتلو

استاد پروژه:

دکتر مینا ملک زاده

اردیبهشت ۹۳

سپاسگذاری:

سپاس خدای را که سخنوران، در ستودن او بمانند و شمارندگان، شمردن نعمت های او ندانند و کوشندگان، حق او را گزاردن نتوانند. و سلام و دورد بر محمّد و خاندان پاك او، طاهران معصوم، هم آنان که وجودمان وامدار وجودشان است؛ و نفرین پیوسته بر دشمنان ایشان تا روز رستاخیز...

بدون شک جایگاه و منزلت معلم، اجّل از آن است که در مقام قدردانی از زحمات بی شائبه ی او، با زبان قاصر و دست ناتوان، چیزی بنگاریم.

اما از آنجایی که تجلیل از معلم، سپاس از انسانی است که هدف و غایت آفرینش را تامین می کند و سلامت امانت هایی را که به دستش سپرده اند، تضمین؛ بر حسب وظیفه و از باب " من لم یشکر المنعم من المخلوقین لم یشکر الله عزّ و جلّ" :ازپدر و مادر عزیزم...این دو معلم بزرگوارم... که همواره بر کوتاهی و درشتی من، قلم عفو کشیده و کریمانه از کنار غفلت هایم گذشته اند و در تمام عرصه های زندگی یار و یاوری بی چشم داشت برای من بوده اند؛از استاد با کمالات و شایسته؛ جناب آقای دکتر .... که در کمال سعه صدر، با حسن خلق و فروتنی، از هیچ کمکی در این عرصه بر من دریغ ننمودند و زحمت راهنمایی این رساله را بر عهده گرفتند؛از استاد صبور و با تقوا ، جناب آقای دکتر .... ، مدیریت محترم کرسی گروه، که زحمت مشاوره این رساله را در حالی متقبل شدند که بدون مساعدت ایشان، این پروژه به نتیجه مطلوب نمی رسید؛و از استاد فرزانه و دلسوز؛ جناب آقای دکتر ... که زحمت داوری این رساله را متقبل شدند؛ کمال تشکر و قدردانی را دارم.

باشد که این خردترین، بخشی از زحمات آنان را سپاس گوید .

چکیده:
موتور های بازی ساز از نظر دسته بندی و نحوه اجرا بسیار گسترده اند ولی یک سری اجزا در موتور های بازی ساز مشترک هستند مثل سیستم انیمیشن و سیستم فیزیک. در این مقاله به مباحث ابتدایی بازی سازی، موتور های بازی و به طور ویژه به موتور بازی یونیتی پرداخته شده است. از آنجایی که کار با موتور بازی نیازمند این است که بدانیم که این نرم افزار در کجای فرآیند بازی سازی جای دارد لذا ما نگاهی جزئی به مراحل ساخت بازی انداختیم و سپس برای درک بهتر از مراحل ساخت یک بازی ما وارد بحث سیستم های موجود در یک بازی شدیم.

کلمات کلیدی: موتور بازی ساز، ساخت بازی، سیستم های بازی، یونیتی

فهرست مطالب

فهرست تصاویر

صفحه

فهرست جداول صفحه

۱

فصل اول

مقدمه

۱.۱ مقدمه

تأثیر گذاری بالای بازی های ویدئویی این ابزار را به یک ابزار با استعداد بسیار بالا برای آموزش، آگهی، ترویج مذاهب، سو استفاده سیاسی و... تبدیل کرده‌است. امروزه صنعت تولید بازی‌های رایانه‌ای به یک عرصه سود آور تبدیل شده‌است و شرکت‌های بزرگ تولید کننده این گونه بازی‌ها هر سال سود هنگفتی به دست می‌آوردند، به گونه‌ای که در سال ۲۰۰۴ میزان سود خالص سالانه به دست آمده از صنعت بازی‌های رایانه‌ای به مرز ۱۰ میلیارد دلار رسید و سود حاصل سالانه به دست آمده از هالیوود (نه و نیم میلیارد دلار) را پشت سر گذاشت.

۲

فصل دوم

مبانی بازی سازی

۲.۱ بازی سازی چیست؟

بازی سازی به معنای ساختن بازیهای رایانه‌ای است که ساخت بازیهای رایانه‌ای با نرم‌افزارهایی انجام می‌شود که به این نوع از نرم‌افزارها انجین می‌گویند. انجین چیست؟ انجین‌ها نرم‌افزارهای ساخت بازی هستند. نرم‌افزارهای ساخت بازی جزو پیچیده ترین نرم‌افزارها بعد ازسیستم عامل‌ها هستند، زیرا نرم‌افزارهای ساخت بازی از اجزای زیادی تشکیل شده‌اند، از قسمت‌های مختلف می‌توان به ادیتور،کامپایلر و امکانات جانبی مانند پارتیکل اشاره کرد.

۲.۲ مراحل طراحی یک بازی ویدئویی

به طور کلی طراحی یک بازی، پروسه ای از طراحی محتوا و قوانین بازی در مرحله پیش ساخت و طراحی بازی نامه، محیط، storyline و کاراکتر ها در طی مرحله ساخت است که با توجه به نوع و مخاطبان بازی مراحل توسعه متفاوتی دارد. طرح توسعه بازی زیر توسط جیمز اف دانیگان طراح بازی های جنگی ارائه شده است:

- ایده توسعه
- جمع آوری منابع
- یکپارچه سازی
- تهیه یک سند کامل از بازی
- تهیه یک پیش نویس اولیه از قوانین
- ساخت بازی
- تست ناآگاه
- ویرایش
- ساخت محصول
- بازخورد

۲.۳ طراحی های مختلف در یک بازی ویدئویی

یک بازی ویدئویی از قسمت های مختلفی تشکیل شده است و به طور کلی گروه های تشکیل دهنده یک تیم بازی ساز برخاسته از بخش های مختلف موجود در بازی سازی است، موارد زیر طراحی های مختلف در یک بازی ویدئویی را نشان می دهد:

- طراحی جهان
- طراحی سیستم
- طراحی محتوی
- نوشتن بازی
- طراحی مرحله
- طراحی رابط کاربری
- طراحی صدا

۲.٤ هوش مصنوعی بازی

در بازی های رایانه ای ، از هوش مصنوعی برای تولید شبیه سازی هوش شخصیت‌های غیر قابل بازی استفاده می‌شود. تکنیک‌هایی که به صورت معمول استفاده می‌شود، از روش‌های موجود در رشته هوش مصنوعی بهره می‌گیرند. با این وجود، اصطلاح هوش مصنوعی بازی، اغلب، به مجموعه‌ای وسیع از الگوریتم‌هایی که شامل تکنیک‌های تئوری کنترل، رباتیک، گرافیک رایانه‌ای وعلوم رایانه می‌باشد، مربوط است.

از آنجایی که هوش مصنوعی در بازی‌های رایانه‌ای بر روی ظاهر هوش و گیم پلی تمرکز دارد، روشی که از آن استفاده می‌کند با روش‌های معمول هوش مصنوعی سنتی تفاوت زیادی دارد. در اینجا روش‌های دور زدن باگ‌ها (Workaround) و تقلب در بازی (Cheats) مورد پذیرش است. مثلاً در اکثر موارد، توانایی‌های کامپیوتر باید پایین آورده شود تا بازیکن‌های انسان بتوانند از پس حریف ماشین خود برآیند. این مثال مخصوصا در بازی‌های تیراندازی اول شخص برجسته‌است. اگر نشانه‌گیری شخصیت‌های غیر قابل بازی فوق‌العاده باشد، بازی کردن با چنین ماشینی فراتر از حد توانایی یک انسان می‌شود.

۲.۵ ساختار یک تیم بازی ساز

یک تیم بازی ساز متشکل از تخصص های مختلف است:

- مدیر ساخت
- مدیر فنی
- کارگردان هنری
- گروه داستان نویسی
- انیماتور
- ایده پرداز
- طراح دوبعدی
- مدل ساز
- گروه صدا
- برنامه نویس شیدر
- برنامه نویس هوش مصنوعی
- برنامه نویس بازی نامه
- هنرمند تکسچر ساز
- گروه تبلیغات و بازاریابی

۲.٦ دسته بندی بازی ها

بازی های ویدئویی از نظر مخاطب و سبک بازی به دسته های مختلفی تقسیم می شوند:

۱ *Action*

۱.۱ *Ball and paddle*

۱.۲ *Beat 'em up and hack and slash*

۱.۳ *Fighting game*

۱.٤ *Maze game*

۱.٥ *Pinball game*

٥ *Simulation*

٥.١ *Construction and management simulation*

٥.٢ *Life simulation*

٥.٣ *Vehicle simulation*

٦ *Strategy*

٦.١ ٤*X game*

٦.٢ *Artillery game*

٦.٣ *Real-time strategy (RTS)*

٦.٤ *Real-time tactics*

٦.٥ *Tower defense*

٦.٦ *Turn-based strategy*

٦.٧ *Turn-based tactics*

٦.٨ *Wargame*

٧ *Sports*

٧.١ *Racing*

٧.٢ *Sports game*

٧.٣ *Competitive*

٨ *Other notable genres*

٨.١ *Casual game*

٨.٢ *Music game*

٨.٣ *Party game*

٨.٤ *Programming game*

٨.٥ *Puzzle game*

٨.٦ *Trivia game*

٨.٧ *Board game / Card game*

۳

فصل سوم

موتور بازی ساز

۳.۱ موتور بازی چیست؟

موتور بازی مجموعه ای از ابزارهای توسعه دیداری علاوه بر مؤلفه‌های نرم‌افزاری با قابلیت استفاده مجدد را ارائه می دهد. این ابزارها معمولاً در یک محیط توسعه یکپارچه ارائه می شوند تا توسعه بازیها را با یک رویکرد مبتنی بر داده ساده تر و سریع تر انجام دهند. موتورهای بازی را گاهی اوقات" میان افزار بازی" نیز می نامند زیرا از نقطه نظر تجاری این اصطلاح، آنها یک سکوی نرم‌افزاری منعطف و قابل استفاده مجدد را ارائه می کنند که تمام کاربردهای موردنیاز را فراهم می آورند تا درحالیکه هزینه ها، پیچیدگی‌ها و زمان ارائه با بازار - که همگی این عوامل در صنعت رقابتی بازی‌های کامپیوتری حیاتی می باشند - کم می کند، توسعه و تولید بازی‌ها را امکان پذیر سازد.

۳.۲ دسته بندی موتور های بازی ساز

۳.۲.۱ موتور های بازی ساز عمومی

انجین‌های عمومی انجین‌هایی هستند که تمام سبک‌ها با این انجین‌ها قابل پیاده سازی است و مخصوص سبک به خصوصی نیستند مانند:

- یونیتی
- گیم میکر

۳.۲.۲ موتور های بازی ساز خصوصی

انجین‌هایی هستند که مخصوص به سبک خاصی طراحی شده‌اند و در سبک بخصوص خود از انجین‌های عمومی کاملتر هستند مانند:

- Frosbit
- IceEngine

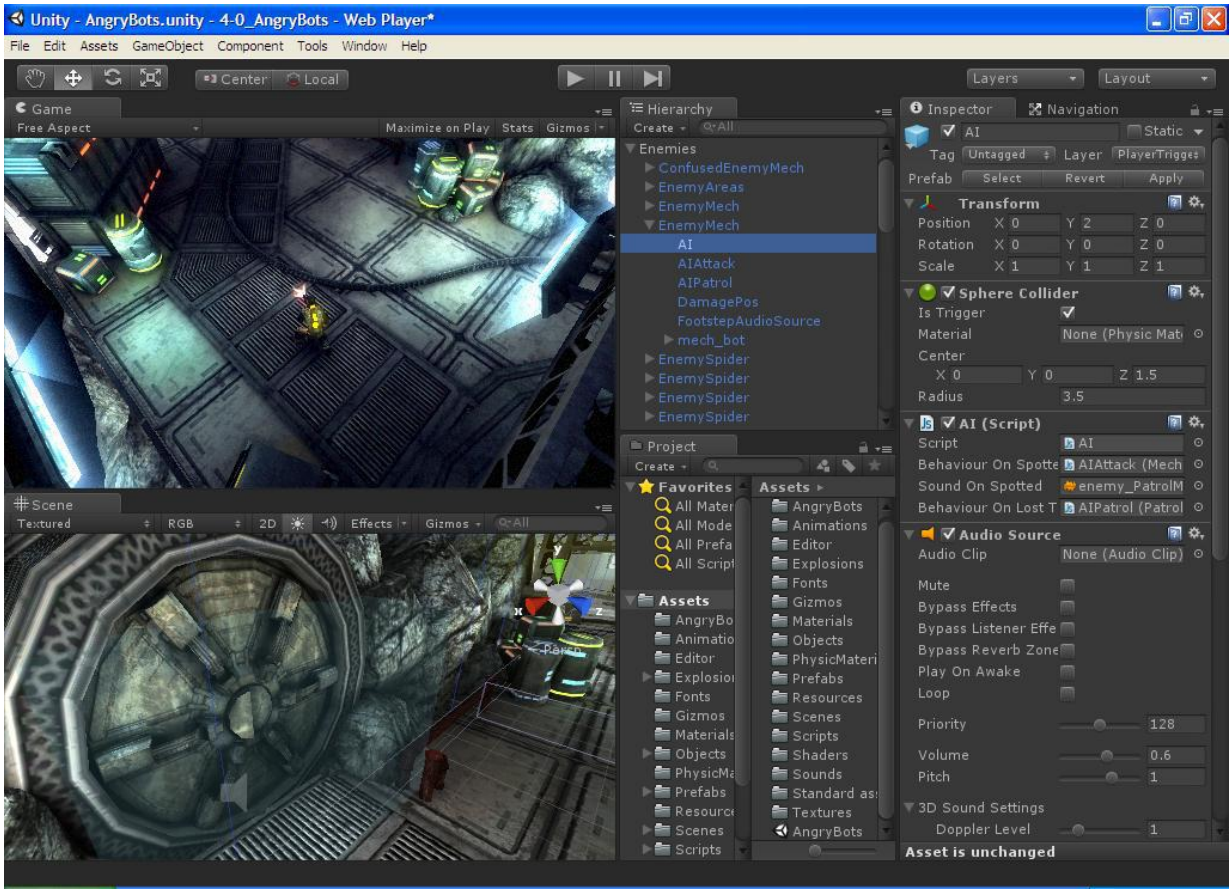
٤

فصل چهارم

مبانی یونیتی

٤.١ معرفی موتور بازی ساز یونیتی

یونیتی جزو چهار موتور بازی ساز بزرگ می باشد که البته روز به روز در حال پیشرفت است، طی چند سال اخیر بسیاری از بازی سازان به سمت این موتور گرایش پیدا کرده اند، حتی در داخل کشور نیز بسیاری از شرکت های بازی سازی این موتور را به عنوان موتور بازی ساز خود انتخاب کرده اند و در حال ساخت بازی با این موتور بازی ساز هستند. ساخت بازی با این موتور بازی ساز بسیار ساده است چرا که بیشتر کار در محیط ویژوال انجام می‌شود، یعنی حدود ۸۰ درصد کار بدون کدنویسی انجام می‌شود و تنها حدود ۲۰ درصد کار کدنویسی است، که این موضوع باعث جذب بسیاری از کاربران علاقه مند به بازی سازی شده است که علم زیادی در برنامه نویسی ندارند.



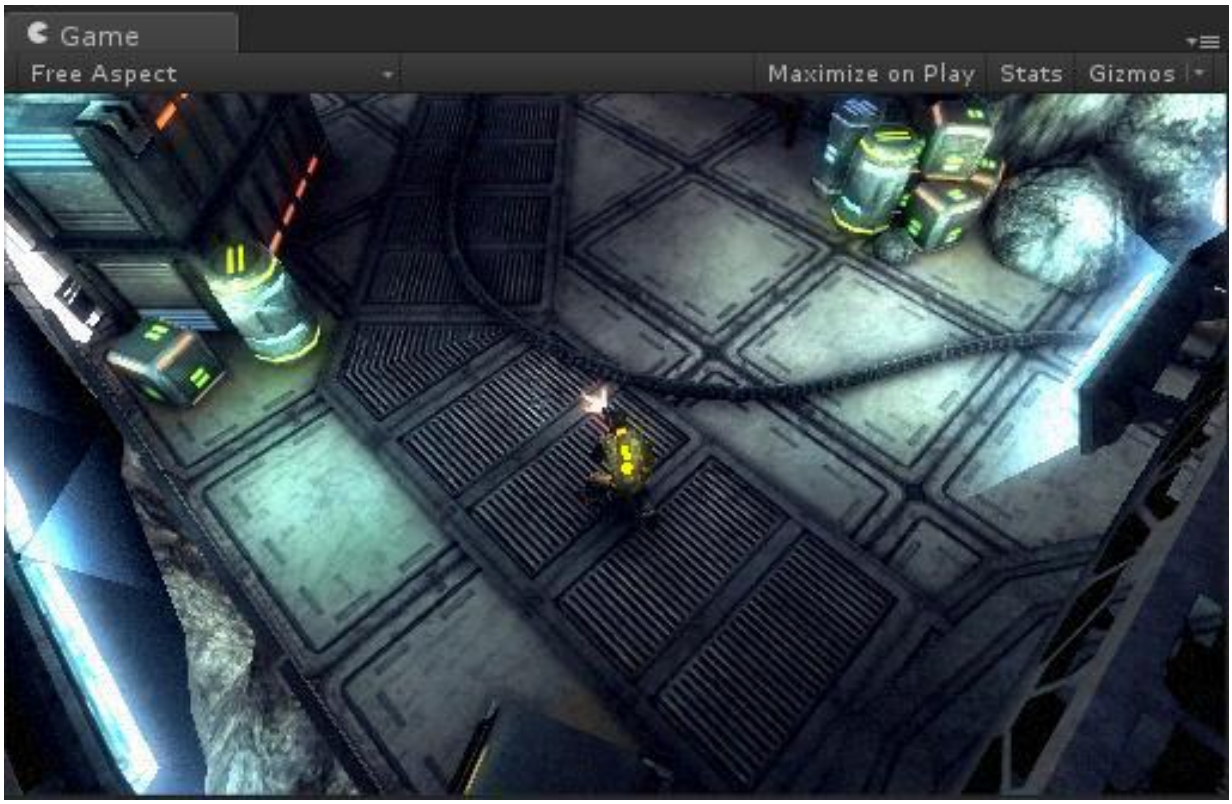
شکل ٤.١ نمایی از موتور بازی یونیتی

٤.٢ رابط کاربری

میتوان از مهمترین قسمت های رابط کاربری یونیتی به موارد زیر اشاره کرد:

٤.٢.١ تب Game

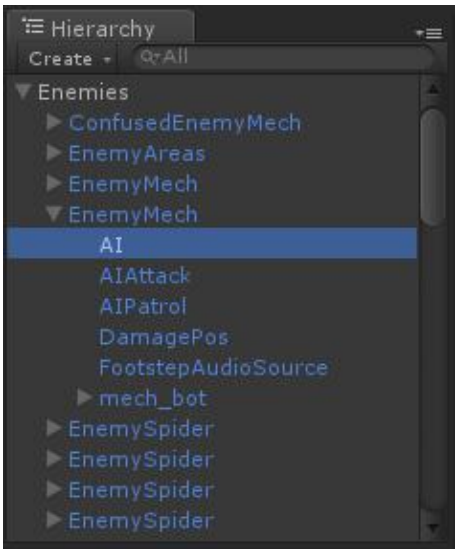
تب Game نمایی از خروجی بازی که توسط یک دوربین گرفته می شود است. این تب شامل دکمه های شروع، توقف و مکث بازی است.



شکل ٤.٢ نمایی از تب Game

٤.٢.٢ تب Hierarchy

در این تب تمام دارایی های موجود در صحنه (*Scene*) نمایش داده می شوند در تب *Hierarchy* گزینه *Create* برای ایجاد اشیاء جدید وجود دارد که به محض ایجاد یک شی آن را در صحنه *(Scene)* اضافه می کند. این تب قابلیت جستجو یک شی در صحنه را نیز داراست. اگر شما یک شی را از صحنه خود پاک کنید متقابلاً در تب *Hierarchy* نیز آن شی پاک می شود.



شکل ٤.٣ نمایی از تب Hierarchy

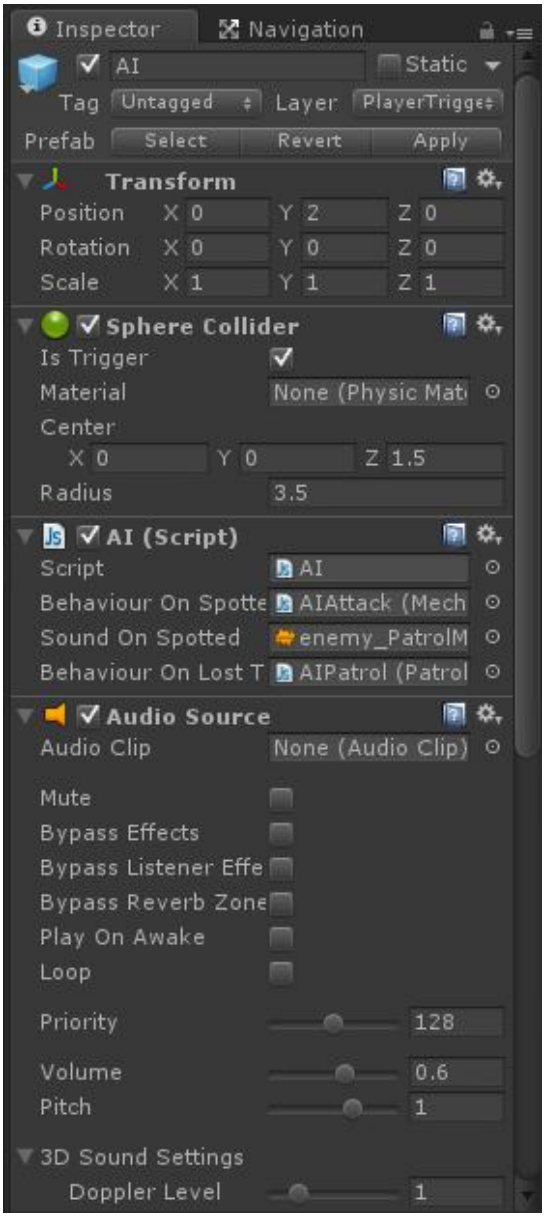
٤.٢.٣ تب Project

این تب تمام دارایی های موجود در کل پروژه را نمایش می دهد این تب نیز به دلیل احتمالی بودن وجود دارایی های زیاد، دارای قابلیت جستجو یک شی است.

شکل ٤.٤ نمایی از تب Project

٤.٢.٤ تب Inspector

این تب خصوصیات یک شی را نمایش می دهد، به عنوان مثال به اطلاعات ثابت موجود در این تب می توان به اطلاعات مربوط به محل قرارگیری شی، چرخش و اندازه آن اشاره کرد. هر کامپوننتی که به شی اضافه شود به تب نیز اضافه می شود.



شکل ٤.٥ نمایی از تب Inspector

٤.٢.٥ تب Scene

این تب نمایی از صحنه را از دید بازی ساز نمایش می دهد. شما در این تب می توانید محیط ، دشمنان، دوربین ها،بازیکن و تمام اشیاء مورد نظر خود را بچینید. نحوه قرار گیری اشیاء در بازی یکی از مهمترین بخش های یک بازی است، پس خیلی مهم است که بتوانید با این تب سریع کار کنید از این رو یونیتی کلید های میانبری برای سرعت بخشیدن به کار با این تب فراهم کرده است.

شکل ٤.٦ نمایی از تب Scene

٤.٣ جریان کار در یونیتی

برای کار در یونیتی باید ابتدا با یک سری اصطلاحات کلیدی برای درک بهتر جریان کار در این موتور بازی ساز آشنا شد. در زیر به مهمترین این اصطلاحات به صورت خلاصه اشاره می شود:

٤.٣.١ Asset

*Asset*ها منابع بازی هستند تمام چیزهایی که در بازی وجو دارند assest هستند (مثل کد، تکسچر، صدا، مدل سه بعدی و...). Asset ها در تب Project نمایش داده می شوند، هر *Asset* ویژگی هایی دارد که با انتخاب آن در تب Inspector نمایش داده می شوند، برای تغییر ویژگی ها نیز می توان از این تب استفاده کرد. برای مدیریت راحت تر Asset ها باید آنها را پوشه بندی کرد، برای وارد کردن یک Asset می توانیم در پنجره Project کلیک راست کرده و گزینه Import New Assets را انتخاب کنیم تا شی مورد نظر وارد یونیتی شود. برخی از *Asset*ها باید در خود یونیتی ساخته شوند مانند متریال، کدها،

شیدرهاو...برای ساخت این نوع *Asset* ها در پنجره *Project* کلیک راست کرده و سپس از گزینه *Create*، *Asset* مورد نظر را انتخاب می کنیم.

٤.٣.٢ Package

به مجموعه ای از Asset ها که در یک فایل قرار دارد *Package* می گویند. یکی از کاربرد های Package استفاده از Assetهای موجود در آن در چندین پروژه است، به عنوان مثال تمام کدها، اشیاء و سایر *Asset*های مورد نیاز برای یک ماشین را در یک Package قرار می دهیم حال اگر در پروژه های بعدی اگر نیازی به استفاده از ماشین داشتیم می توانیم از این Package آماده شده استفاده کرد. معمولاً یونیتی چندین *Package* آماده دارد که می توان از آنها استفاده کرد،از Packageهای آماده یونیتی به موارد زیر می توان اشاره کرد:

- Character Controller
- Glass Refreaction
- Image Effect
- Light Cookies
- Light Flares
- Particles
- Physics Materials
- Scripts
- SkyBoxs
- Terrain Assets
- Toon Shading

٤.٣.٣ Game Object

Game Object مهمترین چیز در یونیتی است، *Game Object* به خودی خود هیچ عملی انجام نمی دهد و با اضافه شدن *Component* ها ویژگی خاصی پیدا می کنند. هر چیزی که در بازی وجود دارد یک *Game Object* است. در یونیتی *Game Object* هایی آماده ساخته شده است که میتوان آن ها را از منوی *Game Object* و *Create Other* ایجاد کرد قابل ذکر است که *Game Object* آماده خود نیز متشکل از یک *Empty game object* بعلاوه یکسری *Component* است . ما بسته به نیاز بازی خود این *Game Object* ها را میسازیم. *Game Object* های یونیتی در لیست زیر آمده است:

- Partical system
- Camera
- GUI text
- Gui texture
- ٣d text
- Directional light
- Point light

- Spotlight
- Area light
- Cube
- Sphere
- Capsule
- Cylinder
- Plane
- Quad
- Sprite
- Cloth
- Terrain
- Ragdoll
- Tree
- Wind zone

٤.٣.٤ Component

Component ها ویژگی هایی هستند که ما به یک *Game Object* میدهیم تا ویژگی خاصی در بازی داشته باشد برای متال به آن *Collider* می دهیم تا در بازی دارای فیزیک باشد یا به او *Audio Source* میدهیم تا در بازی از خود صدایی پخش کند و یا خیلی کارهای دیگر در فصل بعد با بسیاری از این *Component* ها آشنا می شویم. برخی از کامپوننت های اصلی یونیتی:

- Mesh
- Effects
- Physics
- Physics ٢d
- Navagation
- Audio
- Renderring
- Miscellaneous
- Scripts
- Image effects

٤.٣.٥ Prefab

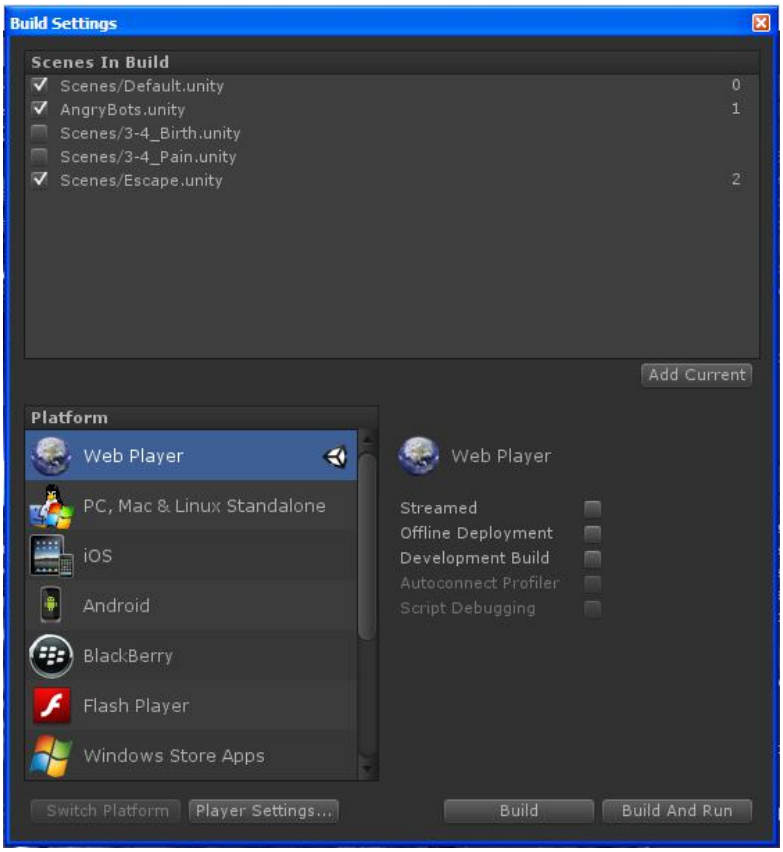
هر Game Object که در صحنه از بازی ساخته میشود مختص همان صحنه است و نمیتوان در صحنه های دیگر بازی از آن استفاده کرد پس ما باید آن را درون یک Prefab قرار دهیم تا هم آن را در صحنه های دیگر استفاده کنیم و یا اگر تغییری در Prefab ایجاد کنیم تمامی نمونه هایی که از این Prefab در بازی قرار گرفته است نیز تغییر می کند. مفهوم Prefab تغریبا شبیه به کلاس در برنامه نویسی است به طوری که ما یک Prefab اصلی ایجاد میکنیم و نمونه هایی از آن را در بازی قرار می دیهم حال با تغییر Prefab اصلی تمامی نمونه ها نیز تغییر می کند.

٤.٤ پلتفرم های خروجی

یونیتی یک موتور بازی ساز چند پلتفرمه است، یعنی می تواند برای بسیاری از پلتفرم های موجود بازی را ایجاد کند. امروزه تنها حدود ۵ درصد کاربران بازی‌ها را بر روی کامپیوترهای شخصی اجرا می کنند و سهم بسیاری به کنسول های بازی و موبایل و وب می رسد. یونیتی برای پلتفرم های زیر خروجی ایجاد می کند:

- مایکروسافت ویندوز
- مک اواس
- وی
- ایکس‌باکس ۳۶۰
- پلی‌استیشن ۳
- آی‌اواس
- اندروید
- وب

و به زودی برای پلتفرم های BlackBerry ۱۰, پلی‌استیشن ۴ ,وی یو ,ویندوز ۸ ۸ Windows Phone ,خروجی خواهد داد.



شکل ٤.٧ نمایی از پنجره تنظیمات خروجی یونیتی

از جمله بازی های ساخته شده برای وب توسط یونیتی می توان به بازی های زیر اشاره کرد:

- Battlestar Galactica
- Fusion Fall
- BeGone
- Marvel Superhero Squad Online
- Paper Moon

به زودی یونیتی خروجی به صورت فلش با پسوند فایل اس دبلیو اف را نیز به خروجی های خود اضافه خواهد کرد که این باعث می شود تا هر پلتفرمی که پخش کننده فایل های فلش را دارد بازی های خروجی گرفته شده از یونیتی را اجرا کند و این انقلابی در بازی های سه بعدی خواهد بود.

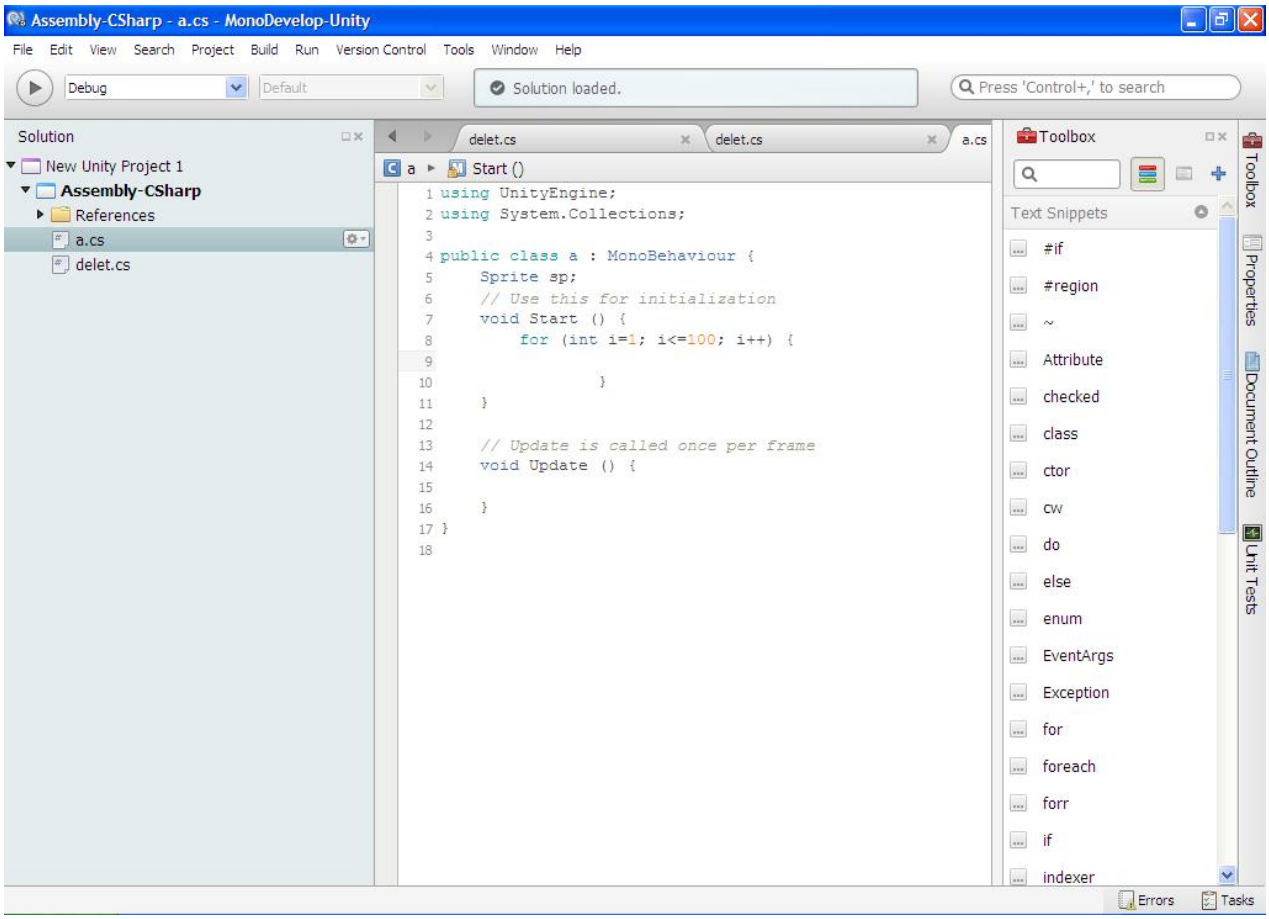
٤.٥ زبان های برنامه نویسی پشتیبانی

یونیتی از جمله موتور های بازی ساز است که می تواند از زبان های برنامه نویسی معروف پشتیانی کند، زبان‌های برنامه‌نویسی سی‌شارپ(*C#.net*) ، جاوااسکریپت و زبان برنامه‌نویسی بو که یک زبان از خانواده زبان برنامه‌نویسی پایتون می باشد در یونیتی قابل استفاده هستند. زبان سی‌شارپ (*C#.net*) یکی از قدرتمندترین زبان های برنامه‌نویسی است که قابلیت شی گرایی دارد و به واسطه این قابلیت، قابلیت هایی همچون ارث بری را نیز دارا می باشد، این قابلیت ها در پروژه های بزرگ برنامه‌نویسی بسیار سودمند هستند.

٤.٦ کامپایلر مونو

این موتور بازی ساز از نرم افزار مونو که یک نرم افزار متن باز ویرایشگر زبان های برنامه نویسی است به عنوان ویرایشگر زبان برنامه نویسی استفاده می کند، این نرم افزار بسیار قدرتمند است و کاملا با موتور بازی ساز یونیتی هماهنگ شده است، مهمترین قابلیت های این نرم افزار در لیست زیر معرفی شده است:

- پیشنهاد دهنده خودکار کلمات
- سیستم دیباگینگ پیشرفته
- پشتیبانی کامل از زبان سی شارپ



شکل ٤.٨ نمایی از کامپایلر مونو

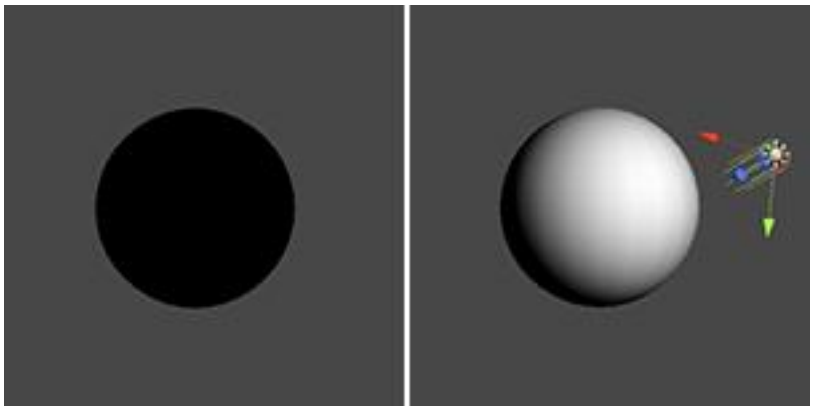
۵

فصل پنجم

بخش های مختلف یونیتی

۵.۱ سیستم نور

نور یکی از اساسی ترین بخش های هر صحنه بازی است. زمانی که مش ها و تکسچر ها معنی اشکال هستند و در صحنه دیده می شوند، همان زمان نورها معنی رنگ و حالت سه بعدی محیط شما هستند. ساختن آنها کنارهم مستلزم داشتن تمرین و تلاش کمی است ولی نتیجه کار می تواند بسیار شگفت انگیز باشد.



شکل ۵.۱ مقایسه دو کره از نظر نورپردازی

۵.۱.۱ منابع نور

نورها می توانند برای منبع شبیه سازی خورشید ، چراغ قوه، آتش تفنگ و یا انفجار مورد استفاده قرار گیرند. برای ساختن هر یک از این شبیه سازی ها از نور متفاوتی استفاده می شود. یونیتی شامل چهار منبع نور است:

منبع نور	توضیح	مثال استفاده در بازی
Direct Lights	اگر این نور ها در فاصله بسیار دور قرار گیرند بازهم بر روی تمامی اجسام بازی اثر می گذارند	خورشید
Point Lights	از محلی به طور مساوی رو تمامی جهت ها نور می تابد	لامپ
Spot Lights	از یک نقطه به یک جهت نور می تابد و فقط اشیائی را که در مسیر نور مخروطی شکل آن قرار گیرند روشن می کند	چراق جلو ماشین
Area Light	به تمامی جهت هایی که یک طرف مستطیل صفحه قرار دارند نور می تابد (مورد استفاده در ***lightmap***)	مورد استفاده در ***light map***

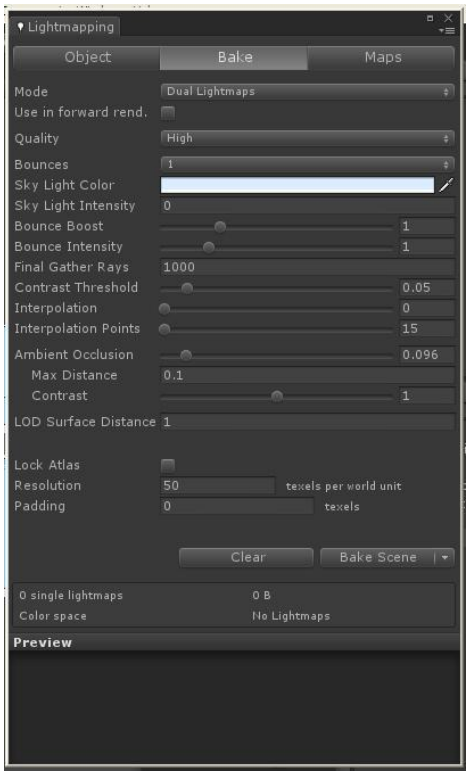
جدول ۵.۱ انواع نور

۵.۱.۲ سایه

نور ها می توانند سایه را تولید کنند، فقط نسخه حرفه ای یونیتی از سایه پشتیبانی می کند، سایه را می توان به وسیله خصوصیت ***shadow type*** در پنجره ***inspector*** تنظیم نمود. سایه دو حال دارد، حال اول سایه سخت یا ***Hard Shadow*** است و حالت دوم حالت سایه نرم یا ***Soft Shadow*** است.

۵.۱.۳ Lightmap

یکی از ویژگی های منحصر به فرد یونیتی *lightmap* است، کار این گزینه در اصل *bake* کردن اثرات نور و سایه بر روی هر شکل و یا اثرات افکت *Ambient Occlusion* بر روی اشیا می باشد، میتوان گفت بدون وجود این گزینه احتمالا بازی ها با این سرعت اجرا نمی شدند. همانطور که می دانیم نورپردازی و مخصوصا سایه زنی جزء اعمالی هستند که نیاز به پردازش های زیادی دارند و تصور این که در هر فریم از بازی این محاسبات انجام شود و بازی با افت فریم مواجه نشود بسیار سخت است. به همین منظور این گزینه برای جلوگیری از این مشکل طراحی شده است. کار این گزینه این است که یک بار این اثرات را برای اجسام ثابت محاسبه و در هر فریم به جای محاسبه مجدد آن، فقط آن را نمایش دهد. در زیر تصوریری از تنظیمات این گزینه را می توانید مشاهده کنید. لازم به ذکر است که پردازش هر بار از این گزینه می تواند برای محیط های شلوغ تا چندین روز به طول بینجامد پس احاطه کامل به گزینه ها میتواند به گرفتن خروجی بهتر در بازه زمانی کمتر کمک کند.



شکل ۵.۲ نمایی از پنجره LightMap

۵.۲ رندرینگ و افکت های تصویری

یونیتی از میان افزار نیم‌سایه برای رندر کردن تصاویر استفاده می کند، نیم‌سایه یک میان افزار برای کنترل رندر است که برای بهبود عملیات رندر از تکنولوژی Occlusion Culling استفاده می کند و با این کار بازدهی بی نظیری در رندر ایجاد می کند، این میان افزار هم اکنون با پلتفرم های ایکس‌باکس ۳۶۰، پلی‌استیشن ۳ وPC سازگاری دارد Occlusion Culling .یکی از بهترین و معروفترین تکنولوژی های برای بالابردن فریم ریت بازی هاست، در صورتی که بخواهیم به ساده ترین شکل ممکن Occlusion Culling را تعریف کنیم باید بگوییم هر چیزی که در دید دوربین است رندر میشود و تمام اجسامی که در خارج از دید آن هستند مورد پردازش قرار نمیگیرند، همین تعریف ساده نشان میدهد که این تکولوژی باعث بازدهی بسیار بالایی خواهد شد. همچنین یونیتی به طور پیش فرض بسیاری از افکت های مورد نیاز دوربین در بازی را به همراه دارد که استفاده از این افکت ها باعث بالا رفتن کیفیت تصویر در بازی می‌شود، دوتا از مهمترین آنها عبارتند از:

Screen Space Ambient Occlusion (SSAO) image effect: این اسکریپت زمانی که بر روی دوربین استفاده شود باعث ایجاد سایه ناشی از نزدیکی دو جسم به یکدیگر به صورت realtime می شود که در زیبا تر جلوه دادن بازی بسیار مؤثر است.

Depth of Field Image Effect: این اسکریپت باعث شبیه سازی لنز دوربین می شود به طوری که منطقه ای از دید دوربین واضح و بقیه مات می باشد، این افکت باعث طبیعی تر شدن دید دوربین می‌شود.

در زیر لیستی از افکت ها آمده است:

- Blur image effect
- Bloom and Flares Image Effect
- Color Correction Curves image effect
- Color Correction image effect
- Contrast Enhance image effect
- Contrast Stretch image effect
- Crease image effect
- Depth of Field Image Effect
- Luminance Edge Blur image effect
- Edge Detection image effect
- Edge Detect Normals image effect
- Fisheye image effect
- Glow image effect
- Grayscale image effect
- Motion Blur image effect
- Noise image effect
- Sepia Tone image effect
- Screen Space Ambient Occlusion (SSAO) image effect
- Sun Shafts image effect
- Twirl image effect
- Vignetting Image Effect
- Vortex image effect

۵.۳ سیستم صدا

۵.۳.۱ صدا در یونیتی

یکی از بخش های بسیار مهم در ساخت بازی صداگذاری می باشد، شرکت های بزگ بازی سازی مبالغ زیادی را صرف ساخت موسیقی بازی می کنند، پس این بخش را باید در بازی‌ها بسیار جدی گرفت .موتور بازی ساز یونیتی از فرمت های *ogg*، *wav*، *aif*، *mp* ۳ پشتیبانی می کند اما برای اینکه صدای بازی بر روی تمامی پلتفرم ها اجرا شود بهتر است از فایل های *mp* ۳ در صدا گذاری بازی استفاده شود *Audioclip*. بخش اصلی صدا گذاری در یونیتی است که از *audiosource* ها استفاده می کند، به این نحو که ابتدا یک *audiosource* برای یک شی در بازی ایجاد می کنیم و بعد با استفاده از *audioclip* آن را کنترل می کنیم. هنگام وارد کردن صدا ها به بازی میتوان نوع صدا را براساس دوبعدی یا سه بعدی بودن انتخاب کرد. صداهای سه بعدی صداهایی هستند که محل پخش آنها در شنیده شدنشان تاثیر دارد مانند صدای شلیک که هرچه فاصله دورتر باشد صدای ضعیف تری شنیده می شود اما صداهای دوبعدی صداهایی هستند که میزان شنیده شدنشان ربطی به محل پخش آنها ندارد و همیشه ثابت است مانند آهنگ پس زمینه بازی.

۵.۳.۲ Audio Listener

این گزینه در واقع گوش کارکتر در بازی است و نقطه ای است که صداها بر اساس آن نقطه شنیده می شود در یونیتی دو نوع صدای سه بعدی و دوبعدی را میتوان تعریف کرد، درصداهای دو بعدی فاصله مهم نیست و صدا به صورت یکنواخت شنیده می شود ولی در صداهای سه بعدی هرچه محل پخش صدا به Audio listener نزدیک تر باشد به همان نسبت صدای بلند تری شنیده می شود. این گزینه تنظیماتی ندارد و به صورت کامپوننت به یک شی اضافه می شود معمولا این گزینه را به دوربین اصلی بازی اختصاص می دهند.

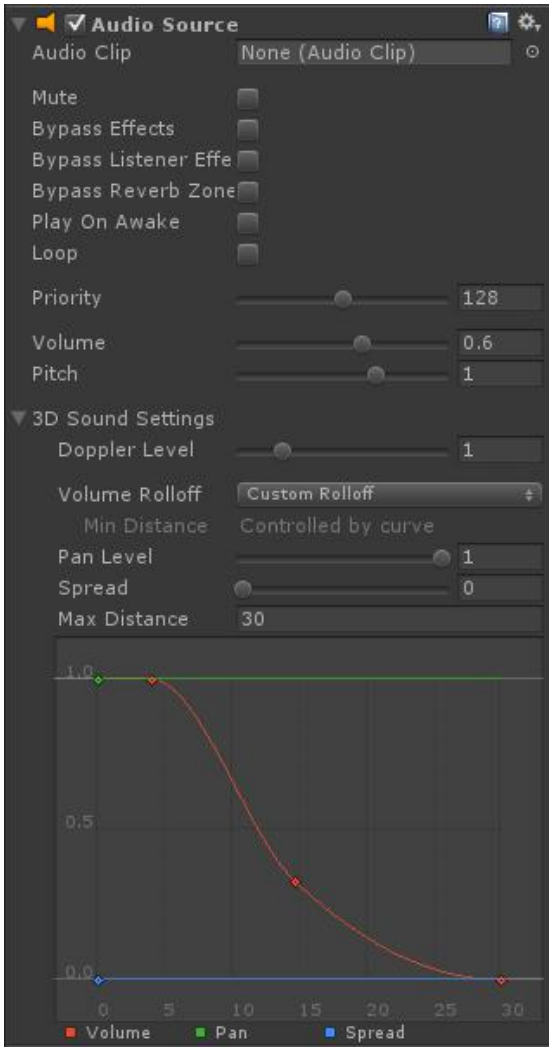
۵.۳.۳ Audio Reverb Zone

این گزینه برای هر منطقه از بازی مشخص میکند که نوع خروجی صدا چگونه باشد مثلا فرض کنید شما در منطقه ای کوهستانی شلیک می کنید صدای خروجی با زمانی که شما درون یک اتاق شلیک می کنید تفاوت دارد.در یونیتی بعضی از مناطق مثل جنگل یا کوهستان یا درون اتاق طراحی شده است اما شما با کم و زیاد کردن گزینه ها میتوانید افکت جدید نیز ایجاد کنید

۵.۳.٤ Audio Source

یک شی قابلیت پخش صدا را دارد به شرط اینکه کامپوننت *Audio Source* به آن شی اضافه شود، این کامپوننت دارای گزینه هایی است که با تنظیم آن ها می توان به نتیجه دلخواه رسید . برای نمونه می توان صدایی پیش فرض به آن داد و یا از طریق کد صدایی به آن شی داد تا آن را پخش کند. لازم به ذکر هست که هر شی در آن واحد فقط قابلیت پخش یک صدا را دارد. از گزینه های موجود در کامپوننت *Audio Source* میتوان به *main distans* و *max distans* اشاره کرد که کمترین و بیشترین فاصله ای که صدا

شنیده می شود را مشخص می کند. فرض کنید یک صدای انفجار صد در صد بازه شنیده شدنش بسیار بیشتر از صدای راه رفتن است.



شکل ۵.۳ نمایی از تنضیمات Audio Source

۵.٤ سیستم انیمیشن

انمیشن ها در یونیتی همراه فایل *FBX* وارد میشود برای اجرای انمیشن در یونیتی باید کامپوننت *Animation* در شی وجود داشته باشد انواع اجرا انمیشن در یونیتی:

- *Loop*
- *Once*
- *Ping pong*
- *Clamp forever*

۵.٤.۱ تب انمیشن

در این تب میتوانید به سخات انمیشن در خود موتور یونیتی این تب به قدرت نرم افزارهای ساخت انمیشن به شما امکانات نمیدهد اما میتواند در بسیاری از مواقع کار را برای شما راحت کند. همچنین شما میتوانید انمیشن های ساخته شده در دیگر نرم افزار ها را نیز در این پنجره ویرایش کنید.

شکل ۵.٤ نمایی از پنجره انیمیشن

۵.٤.۲ مکانیم

با افزودن مکانیم انیمیشن ، یونیتی متحول شده است. این سیستم جدید انیمیشن به هنرمندان امکان تولید درختهای ترکیبی، ماشینهای وضعیت و کنترلر را مستقیماً داخل خود یونیتی میدهد. مکانیم روشی ساده برای تولید انیمیشن از کاراکترهای انسان نما و همینطور تغییر هدف از یک کاراکتر به کاراگتر دیگر را میدهد.

در نسخه ی جدید انیمیشن نسبت به نسخه های قبلی بسیار بهتر و راحت تر ساخته میشود و میتوان با راحتی بیشتری کلیپهای انیمیشن را تنطیم و مرور کرد . قدرت مکانیم در مدیریت تعاملات پیچیده با استفاده از ابزار برنامه نویسی ویژوال میباشد چون این ابزار ویژوال چیزی است که اغلب هنرمندان با آن راحت تر کار می کنند.

۵.۵ سیستم فیزیک

یونیتی دارای موتور فیزیکی NVIDIA PhysX است. این موتور اجازه ایجاد رفتار های فیزیکی بی نظیر را می دهد و همچنین خصوصیات مفید دیگری نیز دارد. برای داشتن رفتار فیزیک قانع کننده بسیار دقیق و سریع نسبت به برخورد ها واکنش نشان دهد و از سایر نیرو ها نیز تاثیر پذیرد. به وسیله کنترل فیزیک توسط کد ها شما می توانید به یک شی تحرک ببخشد(مثل یک وسیله نقلیه، یک ماشین ویا حتی حرکت جزئی از لباس).

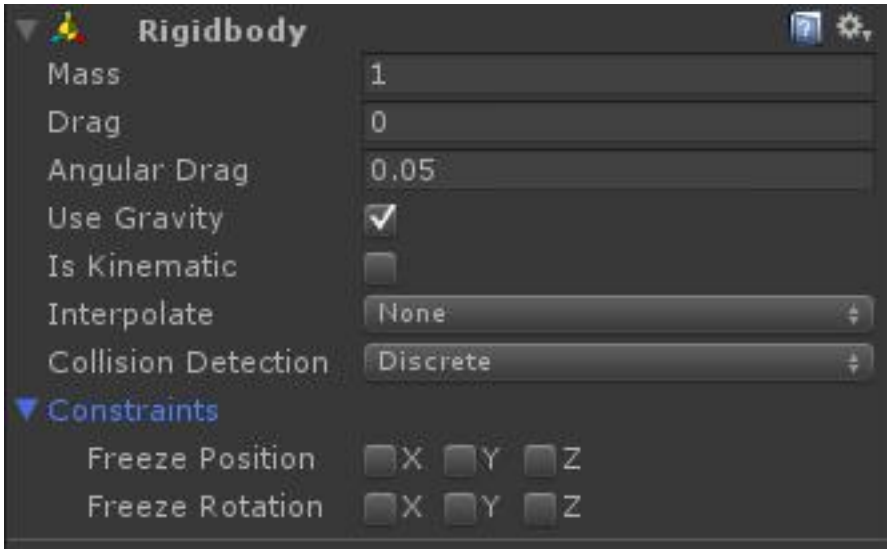
Collider ۵.۵.۱

Collider ها مهمترین بخش یک فیزیک هستند که مشخص کننده ناحیه برخورد فیزیکی جسم را مشخص می کنند اگر شیی دارای *Collider* نباشد یعنی جسم نیست و این به این معنی است که اشیائ دارای *collider* در هنگام برخورد باآن شی از خود هیچ رفتاری بروز نمی دهند و از آن عبور می کنند. در جدول زیر انواع *Collider* به همراه توضیح آن آمده است:

- Box Collider
- Capsule Collider
- Shpere Collider
- Mesh Collider
- Wheel Collider

Rigidbody ۵.۵.۲

Rigidbody شی شما قادر می سازد تا تحت کنترل فیزیک عمل کند. *Rigidbody* می تواند نیرو و گشتاور را به منظور حرکت دادن شی در یک فضای طبیعی دریافت کند. همه اشیاء باید شامل *Rigidbody* باشند تا از نیروی جاذبه تاثیر بپذیرند.



شکل ۵.۵ نمایی از تنضیمات Rigid Body

۵.۶ دوربین

دوربین ها ابزاری هستند که تصاویر را می گیرند و برای کاربر نمایش می دهند. شما می توانید تعداد نامحدودی دوربین در صحنه داشته باشید آنها می توانند در هر آرایشی و هر مکانی روی صحنه باشند و یا فقط بخش خاصی از صفحه نمایش را نشان دهند، آنها می توانند به طور دستی تنظیم شوند و یا کد نویسی

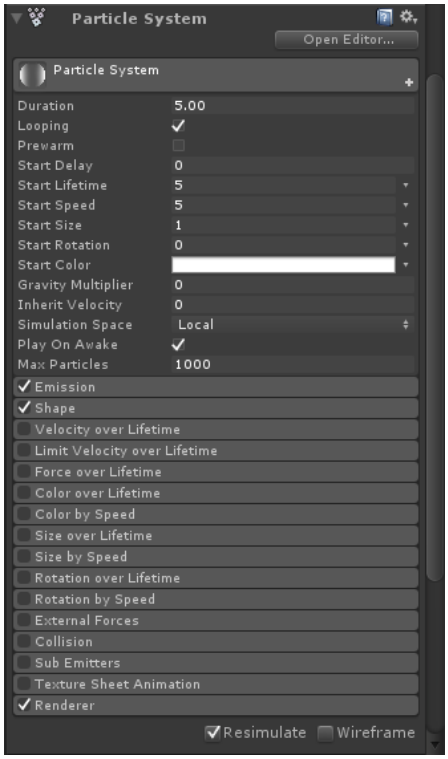
شوند به عنوان مثال برای بازی های پازل شما ممکن است دوربین را برای تمام پازل ثابت در نظر بپیرید. برای اول شخص تیرانداز(*FPS*) شما باید دوربین را به عنوان فرزند کاراکتر در نظر بگیرید و آن را بر روی چشم های کاراکتر قرار دهید برای بازی رانندگی و مسابقه ای شما احتمالاً می خواهید که دوربین وسیله بازیکن را دنبال کند.

۵.۷ سیستم ذرات

اصولا کار سیستم ذرات ایجاد جلوه سه بعدی از تصاویر دو بعدی است، که در یونیتی از سه بخش زیر تشکیل شده است.

- Particle Emitter
- Particle Animator
- Particle Renderer

این قابلیت برای ایجاد آتش، دود، آبشار یا آبنما و از این قبیل جلوه های سه بعدی استفاده می شود، که این کار توسط به حرکت در آوردن تصاویر دو بعدی در زوایا و مسیرهای گوناگون انجام می شود. همچنین می توان برای ذرات قابلیت برخورد را نیز تعریف کرد، یعنی می توان برخورد ذرات با دیگر اجسام در محیط بازی را کنترل کرد که تمامی قابلیت های سیستم ذرات در یونیتی به وسیله اسکریپت نویسی قابل کنترل است.



شکل ۵.۶ نمایی از پنجره Partical System

۵.۸ شیدر

برنامه ای است که به وسیله *gpu* اجرا می شود.
ورژنهای مورد نیاز *Open GL* برای کار با شیدرها:
۱ .شیدرهای :*ARB* و۱.۵ *OpenGL*
۲ .شیدرهای *GLSL* ویرایش ۱.۰ ۱.۵ *OpenGL* : به بالا همراه با الحاقی
۳ .شیدرهای *GLSL* ویرایش ۱.۱ ویرایش استاندارد و مورد تایید بورد : (*ARB* حداقل *OpenGL*۲)
٤.شیدرهای *GLSL* ویرایش ۱.۳ به بالا ۳.*x* *OpenGL* : به بالا

تقریبا تمام نیازها را ویرایش ۱.۱ جوابگو است به جزء شیدرهای Geomtry که آنها هم تحت الحاقی کار می کنند. اما در ویرایش ۱.۳ آنها جزء هسته OpenGL به شمار می روند و کاملاً پشتیبانب می شوند.
اصطلاحا به برنامه ای که عملیات گرافیکی رو کنترل میکنه و تصمیم میگیره که گرافیک نهایی تولید شده رو تصویر به چه شکل باشه شیدر گفته میشه. اصولا وقتی از شیدرها استفاده میشه که کانال گرافیکی ثابت پردازنده گرافیکی جواب گوی تمامی نیازهای ما نباشه. در این موارد برنامه نویسان گرافیک یه برنامه می‌نویسن که به جال کانال پردازش قبلی میشینه و اعمال گرافیکی رو کنترل میکنه. به این برنامه ها که به وسیله *gpu* اجرا میشن اصطلاحا شیدر گفته می شود. هر برنامه شیدر از دو و یا سه قسمت که به ترتیب عبارتند از vertex shader، pixel shader و یا geometry shader تشکیل شده است.

۵.۸.۱ شیدرهای vertex

شیدرهای vertex برنامه هایی هستند که رئوس ارسال شده به سمت پردازنده گرافیکی رو پردازش می کنند و در نهایت حداقل یک موقعیت راس رو تولید می کنند. معمولترین کاری که این شیدرها انجام می دهند جا به جا کردن موقعیت یک راس با توجه به بافت و یا موارد دیگر و محاسبه مختصات بافتها است.

۵.۸.۲ شیدرهای پیکسل

شیدرهای پیکسل و یا فرگمنت در OpenGL مسئول پردازش تک تک پیکسها هستند به شکلی که بعد از پایان پردازش رنگ یک پیکسل رو مشخص می کنند. این نوع شیدها بسیار پر کاربدر هستند و افکتهای خیلی زیادی رو می توان با آنها به سادگی پیاده سازی کرد.

۵.۸.۳ شیدرهای geometry

شیدرهای geometry که بعد از شیدر مدل چهار وارد کارتهای گرافیکی شدند (تنها سری ۸ هزار به بالای geforce از اونها پشتیبانی می کند). این شیدرها بر اساس مدلهای هندسی سه بعدی همانند مثلث و یا مربع عمل میکنند که با کمک آنها می شود حجم سازی رو خیلی خوب پیاده کرد. و در سایه های نرم ، مه، تکنیک relife و ... بسیار موثر عمل می کنند.

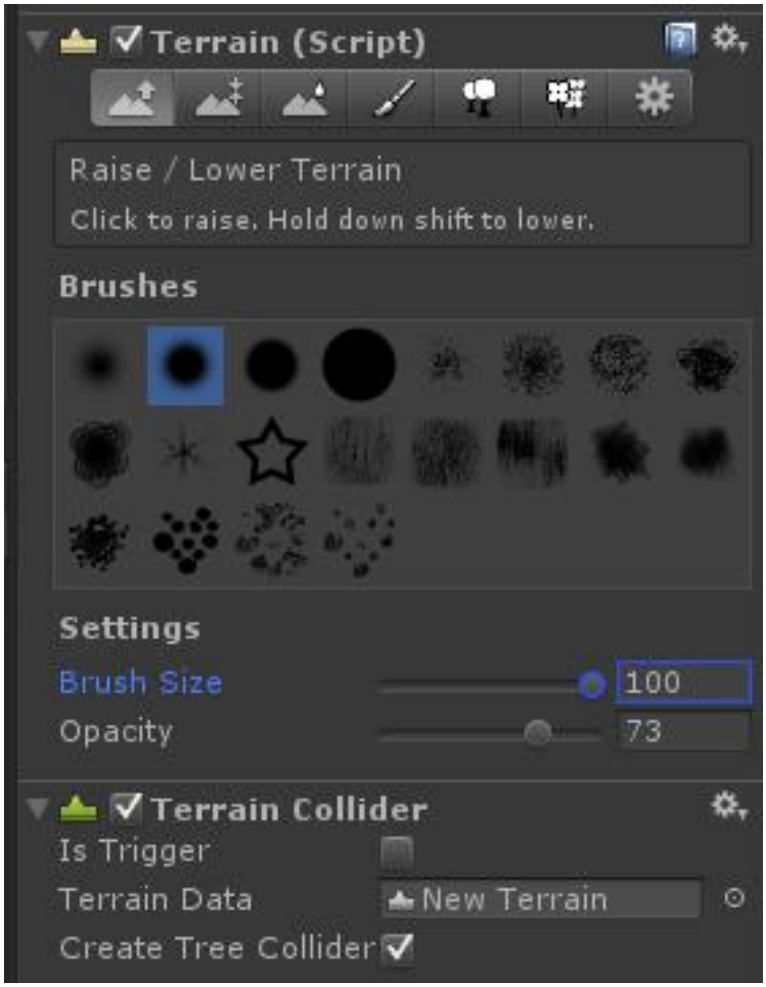
۵.۸.٤ شیدرهای آماده یونیتی

چون زبان شیدر نویسی بسیار دشوار است یونیتی شیدر های پرکاربرد را خود آماده و درون نرم افزار قرار داده است تا آن دسته از کسانی که با زبان دشوار شیدر نویسی آشنایی ندارند هم بتوانند با این موتور بازی بسازند در زیر به برخی از آن ها اشاره می کنیم:

- Bumped diffuse
- Bumped specular
- Decal
- Diffuse
- Diffuse detail
- Parallax diffuse
- Parallax specular
- Specular
- Vertexlit
- Fx
- Gui
- Mobile
- Nature
    - Terrain
    - TreeCreator Bark
    - TreeCreator Leaves
- Particales
    - Additive
    - Alpha Blended
    - Multiply
    - Vertexlit Blended
- Reflective
- Renderfx
    - SkyBox
    - Skybox Cubed
- Selt-illumin
- Sprites
- Transparent
- Unlit
    - Texture
    - Transparent
    - Transparent Cutout
- Legacy shaders

۵.۹ عوارض زمین

یکی دیگر از قابلیت های یونیتی ایجاد عوارض زمین به صورت ویژوالی می باشد، این قابلیت کاربر را قادر می سازد تا سریع و ساده زمین و عوارض آن مانند پستی و بلندی را ایجاد کند، همچنین کاربر می تواند با استفاده از قلم، بافت گذاری و ایجاد آبجکت های گوناگون مانند درختان بر روی زمین را به راحتی انجام داد.



شکل ۵.۷ نمایی از تنضیمات Terrain

۵.۱۰ کامپوننت های دوبعدی یونیتی

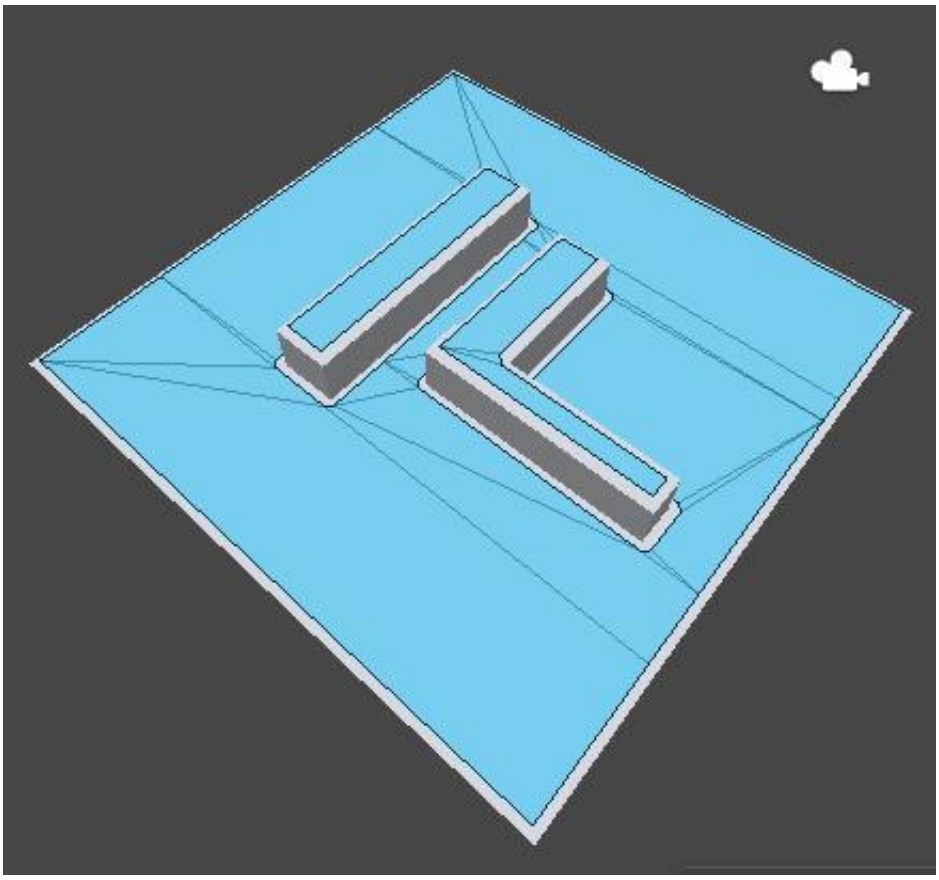
کامپوننت دوبعدی یونیتی یکی از آخرین و البته بهترین کامپوننت هایی است که به یونیتی ٤ اضافه شده است. این کامپوننت ساخت بازی های دوبعدی را بسیار راحت کرده است. در زیر به قابلیت های کامپوننت دوبعدی اشاره خواهیم کرد:

عنوان	توضیح
Sprite Renderer	این اجازه را به شما می دهد که Sprite را در دو نمای دو بعدی و سه بعدی نمایش دهید
Rigidbody 2D	قابلیت های جاذبه سه بعدی را داراست با این تفاوت که در دوبعدی شی فقط می تواند در صفحه xy حرکت کند و حول یک محور عمودی بچرخد.
Circle Collider 2D	همان طور که از نام آن پیداست یک دایره حول شی می کشد که نشان دهنده مرز Collider آن است
Box Collider 2D	یک مستطیل با توجه به طول و عرض داده شده در مختصات محلی Sprite رسم می کند
Polygon Collider 2D	با توجه به نقاط داده شده یک چند ضلعی رسم می کند سپس شما می توانید با دقت بسیار بالا آن نقاط را با شی تنظیم کنید
Edge Collider 2D	با توجه به نقاط داده شده یک لبه (Edge) رسم می کند سپس شما می توانید با دقت بسیار بالا آن را تنظیم کنید
Spring Joint 2D	این اجازه را به شما می دهد دو Sprite متصل به هم چیزی شبیه به یک فنر داشته باشید، فنر نیرویی را در امتداد محور بین دو شی وارد می کند که سعی می کنند یکدیگر را بگیرند
Distance Joint 2D	این اجازه را به شما می دهد دو Sprite که به هم متصل شده اند از یک فاصله ثابت توسط فیزیک جاذبه کنترل شوند، Distance Joint بر عکس Spring Joint است، Disnance Joint محکم است و اجازه نمی دهد که دو شی از هم دور شوند
Hingle Joint 2D	این اجازه را به شما می دهد دو Sprite که به هم متصل شده اند از نقطه ای در فضا توسط فیزیک جاذبه کنترل شوند که شی بتواند دور آن نقطه بچرخد

جدول ۵.۲ کامپوننت های دوبعدی یونیتی

۵.۱۱ مسیر یابی(Path Finding)

بی شک یک بازی خوب بدون داشتن مسیر یابی درست و صحیح نمی توان تصور کرد برای مثال بازی با گرافیک عالی و صدای گزاری زیبا و .. را تصویر کنید اما دشمنان بازی برای پیدا کردن شما به درو دیوار می خورند آیا این بازی را میتوان حتی باز ی نام گذاشت، یونیتی در نسخه های اولیه گزینه ای بدین منظور تعبیه نشده بود اما در نسخه های جدید خود با گزینه $Navigation$ این امکان را به راحتی فراهم کرده اما باید توجه کرد که این مسیر یابی به صورت ثابت و $static$ است . روند کار این گزینه بدین شکل است که شما ابتدا مسیر هایی که قابل حرکت است را $Bake$ میکنید سپس کامپوننت $NavMesgAgent$ را با کاراکتر مورد نظر اظافه میکنید حال در زمان کد نویسی با دادن نقطه هدف کاراکتر از نزدیک ترین راه به سمت نقطه مورد نظر شما میرو در تصویر زیر نمایی $bake$ شده از محیطی را می توانید ببینید.



شکل ۵.۸ نمایی از خروجی مسیریابی

۵.۱۲ منابع آموزشی

یکی از برتری های موتور بازی ساز یونیتی داشتن فایل راهنما بسیار قدرتمند است، این باعث می شود تا برنامه نویسی با یونیتی بسیار سریع و راحت انجام شود، این فایل راهنما به همراه یونیتی پس از نصب به صورت آفلاین در دسترس برنامه نویس قرار می گیرد و تقریبا تمامی کلاس ها و متدهای مورد استفاده یونیتی را به همراه مثال کاربردی به سه سی شارپ، جاواسکریپت و بو (*BOO*) را دارد. همچنین در سایت یونیتی قسمت هایی برای پشتیبانی برنامه نویسی در نظر گرفته شده است که در صورت داشتن هرگونه سوال می توان به آنها مراجعه کرد. چندین شرکت ساخت فیلم آموزشی برای موتور بازی ساز نیز ساعت ها فیلم آموزشی ساخته اند، همچنین کاربران بسیاری در زمینه آموزش یونیتی فیلم ساخته و یا مقاله نوشته اند که تمامی آنها در اینترنت قابل دسترسی است.

٤.١٣ قیمت

موتور بازی ساز یونیتی با توجه به قابلیت های بسیاری که دارد دارای قیمت مناسب و ارزانی است، که این خود یکی از دلایل روی آوردن بازی سازیان به این موتور بازی ساز شده است. این موتور بازی ساز در دو نسخه رایگان و غیر رایگان یا حرفه ای عرضه می شود، قیمت نسخه حرفه ای حدود ۱۵۰۰ دلار می باشد که البته برای بار اول باید این مبلغ را پرداخت کرد و برای نسخه های جدید تر نیازی به پرداخت دوباره نیست بلکه تخفیف بسیاری برای نسخه های بعدی به مشتری داده می شود. همچنین می توان به صورت ماهیانه با پرداخت ۷۵ دلار لایسنس این موتور را خریداری کرد.

اصطلاحات مهم بازی سازی

ایده (Concept)

ایده اصلی که بازی بر اساس آن ساخته شده است. در طراحی کاراکتر ها نیز Concept به ایده اولیه طراحی گفته می شود.

سند ایده ($Concept\ Document$)

دو یا سه صفحه از سند که یک بازی را در شرایط کوتاه و ساده شرح می دهد و به عنوان یک گام فروش برای به دست آوردن یک سرمایه استفاده می شود.

مرور بازی (Game OverView)

بخشی از سند بازی که مخاطبان مورد هدف، پلتفرم های مورد هدف، ویژگی های کلیدی، خلاصه داستان، محیط و ارزیابی ریسک را شرح می دهد.

بازی نامه Emergent (Emergent Game Play)

یک بازی نامه که به خودی خود و توسط برنامه نویسی هوش مصنوعی ایجاد می شود و کاراکترهای بدون بازیکن (NPC)، دشمنان و هیولاها را در یک بازی کنترل می کند. بسته به حرکت ها، انتخاب ها و موقعیت بازیکن، NPC ها می توانند به طور شگفت انگیزی در حالت های مختلف واکنش نشان دهند. به عبارتی ساده تر یک بازی نامه که توسط طراح برنامه ریزی نشده است.

کژوال ($Casual$):

بازی ویدئویی که در مدت زمان کوتاهی ساخته شود، تجربه حادثه ای واینکه مردم بازی می کنند تا نسبت به بازی کتک کاری آرامتر شوند یا به این بازی ها اغلب بازی های کژوال گفته می شود. بازی پرنده های خشمگین به عنوان یک نمونه بزرگ از بازی های کژوال شناخته می شود. بازی های ویدئویی کژوال هزینه بسیار پایینی را برای ساخت و البته هزینه پایینی برای فروش نسبت به بازی های درجه یک دارند.

تست ناآگاهانه (*Blind Testing*)

یکی از روش های تست بازی که طراح بازی درگیر آن نشده باشد سپس گیمرها فقط مثل اینکه بازی را خریده اند و از کارتون درآورده اند بازی می کنند. برای بازی های رو میزی این آخرین مرحله قوانین است.

بازی نامه (GamePlay)

بازی نامه قلب هر بازی است. چه اتفاقیی خواهد افتاد؟ بازیکن چه کاری باید انجام دهد؟ چه چیزی جالب یا لذت بخش است(یا هر دو)؟

مکانیزم یا مکانیک (*Mechansim/Mechanic*)

قوانین بازی به طور کلی متدهایی رو توضیح می دهند که بازی رو به جلو حرکت میکند و این متد های مکانیک های بازی هستند. برای مثال، قلتیدن دو تاس و حرکت شما که نتیجه جمه تاس روی صفحه بازی است این یک مکانیک بازی است. حرکت یک جزء روی یک صفحه مربعی ۸ در ۸ بر طبق توانایی حرکت یک جزء یک مکانیک در بازی شطرنج است. در بازی های ویدئویی مکانیک ها نتایج در رقابتی است که در آن بازیکن برای پیروز شدن به مبارزه می طلبد (مثل حرکت بک دسته بازی یا فشردن یک دکمه).

قوانین (*Rules*)

تمام بازی های داری قوانین هستند. در بازی های ویدئوی قوانین سرتاسرمکانیک های بازی به عنوان بیان شده اند به عنوان اجبار شده توسط نرم افزار. در بازی های رومیزی به طور واقع قوانین نوشته شده اند برای آنکه بازیکن ها باید بفهمند. یک دلیل که چرا بازی های ویدئویی اینقدر محبوب شده اند این است که هیچ چی برای خواندن قوانین ندارند.

پلتفرمر (Platformer)

دسته ای از بازی های ویدئویی است که در آن فعالیت اصلی پرش، دویدن و جستن است، از پلتفرمی به پلتفرم دیگر پریدن(معمولا بازی های پلتفرمر در بین آسمان و زمین انجام می شوند).

طراحی مرحله (*Level Design*)

خیلی از بازی های ویدئویی دارای چند مرحله، طبقه، ماموریت و یا داستان فرعی بازی هستند. این ها معمولا توسط یکی از بخش های طراحی به نام طراح مرحله طراحی می شوند. این افراد کاری با هنر های بصری ندارند اگر چه بعضی از طراحان مرحله ممکن است ایده بازی را نیز فراهم کنند.

این خیلی عادی است در بازی های ویدئویی که طراح مرحله کدنویسی برنامه های کوچک را نیز برعهده بگیرد تا کمک کند مرحله را شخصی ترکند سپس او گرافیک را خلق می کند .

ژانر (Genre)

ژانرهای بازی های ویدیویی برای دسته بندی بازی ها بر اساس تعامل گیم پلی آنهاست نه براساس تفاوت های بصری یا روایتی بازی ها. یک ژانر تعریفی است از مجموعه چالشهای گیم پلی. بر خلاف دیگر آثار هنری مانند فیلم ها یا کتاب ها، بازی ها از روی تنظیمات و جهان محتوایی خود طبقه بندی نمیشوند. برای مثال، یک بازی اکشن که چه در دنیای فانتزی و چه در فضای بین سیاره ای اتفاق بیافتد، هنوز هم اکشن است. در مطالعات بازی ها هنوز تعریف رسمی برای ژانرهای بازی وجود ندارد، و فقط برخی از آنها بیش از دیگران استفاده شده اند. مانند هر طبقه بندی دیگری ژانرهای بازی ها هم به ثابت های مشخصی نیاز دارند.

بازی آرکید (Arcade Game)

یک بازی با حرکت های سریع، جایی که هماهنگی چشم و دست لازمه ی مهارت اصلی برای انجام آن بازی می باشد. به عنوان مثال: بازی های *Shank, Rayman, Mario, Sonic*

منشعب کردن رویدادها(Branching Events)

یک روش ایجاد بازی نامه غیر خطی با استفاده از دادن گزینه های متفاوت در نقاط کلیدی بازی به بازیکن و اجازه دادن برای دنبال کردن مسیر های مختلف می باشند.

نتیجه گیری:

با توجه به امکانات یونیتی و مقایسه آن با موتور های دیگر یونیتی دارای ویژگی هایی است که به نظر نویسنده می تواند انتخاب بسیار مناسبی برای تمامی بازی سازهای مستقل جهان و الخصوص ایرانی باشد در زیر برخی از این ویژگی ها که منجر به این نظر شد:

سرعت پیشرفت بسیار زیاد نسبت به موتورهای دیگر

دردسترس بودن آموزش های این موتور

پشتیبانی بسیار قوی شرکت سازنده مانند سایت و راهنمای برنامه، سایت *unityanswer* و یا سایت *wikiunity* و ....

قیمت بسیار مناسب

خروجی های بسیار متنوع(تقریبا از تمامی پلتفرم ها پشتیبانی می کند)

منابع

١) Rollings, Andrew; Adams, Ernest (2003). Andrew Rollings and Ernest Adams on Game Design. New Riders Games.

٢) Gregory, Jason ,Game Engine Architecture, ISBN 978-1-4398-6526-2

٣) Gahan,Andrew,3ds Max Modeling For Game

٤) http://www.takingame.com/blog/2014/02/24/

٥) Milington, Ian, Artificial Intelligence For Game

٦) Watkings, Adam, Creating Game With Unity And Maya, ISBN 978-0-240-81881-8

٧) Finnegan, Thomas, Unity Android Game Development, ISBN 978-1-84969-201-4

٨) McDermott, Wes, Creating 3D Game Art for the iPhone with Unity

٩) Van Donegen, joost , Using raycasting in GPU shaders to enhance real-time graphics

١٠) Fosner, Ron, Real-Time Shader Programming, ISBN 1558608532

١١) www.En.wikipedia.com

١٢) "What is a Game Engine?". GameCareerGuide.com. Retrieved ٢٠١٣-١١-٢٤.

١٣) Bramwell, Tom (2007-08-09). "id Tech 5 Interview • Page 1 • Interviews •". Eurogamer.net. Retrieved 2013-11-24.

١٤) James Arvo (editor). *Graphics Gems II*. San Diego, CA: Academic Press,1991.

١٥) James J. Buckley and Esfanfiar Eslami. *An Introduction to Fuzzy Logic and Fuzzy Sets*.Springer-Verlag, Berlin/New York, 2002.

١٦) Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, San Francisco,CA, 2005