دانشگاه صنعتي اميركبير
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

گزارش نوشتاری

عنوان

آشنایی با موتورهای بازی سازی

نگارش
امیرحسین بینش

استاد راهنما
دکتر رضا صفابخش

بهار ۹۸

دانشگاه صنعتي اميركبير
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

گزارش نوشتاری

عنوان

آشنایی با موتورهای بازی سازی

نگارش
امیرحسین بینش

استاد راهنما
دکتر رضا صفابخش

بهار ۹۸

تقدیر و تشکر

زندگی صحنه یکتای هنرمندی ماست، هرکسی نغمه خود خواند و از صحنه رود،

صحنه پیوسته بجاست، خرم آن نغمه که مردم بسپارند به یاد.

از استاد گرانقدر، جناب آقای دکتر رضا صفابخش، که مرا در این راه همراهی نمود کمال تشکر را دارم.

همچنین از حمایت‌های پدر و مادرم که همیشه در این راه کمک دست من بوده‌اند، کمال سپاس را دارم.

چکیده

بازی‌های ویدئویی در بخش مهمی از صنعت کشورهایی مثل فرانسه و کانادا ایفای نقش می‌کند. این پیشرفت ناشی از ۴۰ سال پیشرفت مداوم در این زمینه بوده است. امروزه بیش از نیمی از مردم جهان، وقت خود را صرف این سرگرمی پرطرفدار می‌کنند.

ما پیشرفت صنعت بازی‌های ویدئویی را مدیون ساخته شدن ابزارهای ساخت بازی هستیم که به هرچه سریع و آسان شدن ساخت بازی‌ها کمک کرد. همچنین باعث شد که تمرکز بیشتری روی کیفیت بازی‌ها داشته باشیم.

در این گزارش قصد داریم این ابزار قدرتمند را به طور مفصل‌تر بررسی کنیم.

واژه‌های کلیدی:

موتورهای بازی‌سازی، بازی‌های ویدئویی، طراحی بازی‌های سه بعدی، وظایف موتورهای بازی‌سازی

فهرست مطالب

فصل اول

مقدمه

۱- مقدمه

۱-۱ ساخت بازی‌های ویدئویی

در گذشته برای ساخت بازی‌های ویدئویی، ابتدا ایده‌ی اولیه بصورت نقشه‌ای روی کاغذ پیاده می‌شد. که به این مرحله **ایـده‌پـردازی** می‌گویند. سپس پس از ارائه‌ی توضیحات به برنامه‌نویسان، آن‌ها شروع به ساختن بازی میکردند. این فرایند از دو مرحله‌ی اصلی تشکیل می‌شد؛ **طـراحـی** و **بـرنـامـه‌نـویـسـی** که بصورت جزئی در زیر مشاهده می‌کنید:

- طراحی گرافیکی[4]
- طراحی مراحل[5]
- ساخت صدا و موسیقی
- انکود صدا[6]
- طراحی هوش‌مصنوعی
- نحوه‌ی عکس‌العمل محیط به ورودی‌های کاربر
- نمایش اطلاعات بازی در رابط کاربری

۱-۲ مشکلات راه‌حل‌های قدیمی

تمام این فرایند باید برای ساخت بازی جدید از ابتدا تکرار میشد. بازی جدید باید دوباره از صفر ساخته می‌شد و به‌همین دلیل، بازی‌های ویدئویی ابتدایی نمی‌توانستند از حداکثر قدرت سخت‌افزار استفاده کنند.

این باعث شد که بازی‌سازها به فکر ساخت نرم‌افزاری بیافتند که ساخت بازی را آسان‌تر می‌کند و از دوباره‌کاری جلوگیری می‌کند. در ابتدا این نرم‌افزارها فقط به شکل محیط

[1] Blueprint
[2] Design
[3] Development
[4] Visual Design
[5] Level Design
[6] Sound Encode

یکپارچه توسعه های ابتدایی بودند ولی صنعت بازی به مرور زمان شاهد پیشرفت آن‌ها بود تا اینکه در اوایل دهه‌ی ۹۰ میلادی، نخستین موتور بازی‌سازی معرفی شد.

۱-۳ تاریخچه‌ی صنعت بازی‌سازی

تاریخچه‌ی پیشرفت صنعت بازی در سه فاز خلاصه می‌شود؛ **فاز ابتدایی** ، **دوران طلایی** و **دوران پساطلایی** . دو دوره‌ی اول بیشتر مربوط به پیشرفت سخت‌افزارها[۲] و افزایش مشتریان بازی‌های ویدئویی می‌شود اما دوره‌ی پساطلایی به پیشرفت در ساخت بازی‌ها و ساخت موتورهای بازی سازی مربوط است. نمودار زیر این سه دوره را با بازی پیشرو در آن دوره را نمایش می‌دهد. (نمودار ۱-۱)

نمودار ۱-۱

[1] IDE
[2] Early Stage
[3] Golden Age
[4] Post-Golden Age

\\

فصل دوم

تعریف

۲- تعریف

مـوتـور بـازی یک نرم‌افزار برای ساخت بازی است[1] که با در اختیار قرار دادن ابزارهای لازم به کاربر موجب سهولت ساخت بازی می‌شود. در بعضی از تعاریف موتور بازی را یک فریم ورک[2] تعریف می‌کنند؛ به این معنی که سکویی[3] برای ساخت اپلیکیشن‌های نرم‌افزاری است. در بعضی از طبقه‌بندی-ها موتوربازی در زیرمجموعه‌ی محیط یکپارچه‌ی توسعه قرار دارند که از نظر برخی‌ها دسته‌بندی درستی نیست.

۲-۱ کاربردها

همانطور که در مقدمه ذکر شد، دلایل زیادی باعث شد تا موتورهای بازی ساخته شوند، اما این ابزار به مرور زمان پیشرفت‌هایی داشت که امروزه بدون آن‌ها ساختن بازی‌های باکیفیت بالا[4] میسر نیست. امروزه موتورهای بازی به دلایل مختلفی بطور گسترده استفاده می‌شوند که به بعضی از آن‌ها می‌پردازیم:

۲-۱-۱ ابزارهای آماده‌ی ساخت بازی

موتورهای بازی با داشتن ابزارهای بصری بصورت دو و سه بعدی بصورت بلادرنگ[5] محیط بازی را نمایش می‌دهد. همچنین ابزارهای دیگری مثل مُحیط‌های برنامه‌نویسی بصری، که بصورت ماشین حالت، حالتی که بازی و بازیکن در آن قرار دارد را نمایش می‌دهد، که موجب می‌شود بازی‌ساز بتواند تغییرات ایجاد شده در بازی را ببیند.

۲-۱-۲ استفاده‌ی مجدد از نیم پروژه‌های ساخته شده[6]

موتورهای بازی سازی این قابلیت را به تیم‌های سازنده‌ی بازی می‌دهند که بتوانند بصورت همزمان روی قسمت‌های مختلف بازی کار کنند و آن قسمت را توسعه

[1] Game Engine
[2] Framework
[3] Platform
[4] AAA Games
[5] Real-time
[6] Mini Project

دهند و به‌صورت نیم پروژه هایی دخیره کنند تا در نهایت بصورت قابلیت‌های جدیدی به پروژه‌ی اصلی بازی اضافه شوند.

۲-۱-۳ قابلیت انتقال[۲] بازی روی کنسول های مختلف

امروزه اکثر بازی‌ها بر روی سیستم‌های مختلفی ارائه می‌شوند که به آن‌ها چندسکویی می‌گوییم. پیشرفت سخت‌افزاری باعث شده که شرکت‌هایِ مختلف کنسول بازی اختصاصی خودشان را داشته باشند و به دلیل تفاوت در سخت‌افزار آن‌ها لازم است بازی‌ها برای سخت‌افزارهای مختلف، مجددا توسعه داده شوند، اما موتور-های بازی با قابلیت انتقال، بازی‌سازها را از این کار تکراری آسوده کنند. برای مثال موتور آنریل[۵] قابلیت انتقال بازی روی کامپیوترهای شخصی[۶]، پلی‌استیشن[۷]، ایکس-باکس[۸]، اندروید[۹]، آی او اس[۱۰]، وب و ... را دارد.

۲-۱-۴ تسریع توسعه‌ی بازی

موتورهای بازی با ابزارهای آماده که در اختیار بازی‌سازها قرار می‌دهند، توسعه‌ی بازی‌ها را سرعت می‌بخشند. برای مثال اگر در گذشته نیاز بود تا قابلیتی به بازی اضافه شود باید از اول بازی را برنامه‌نویسی می‌کردند و آن قابلیت را هم به آن اضافه می‌کردند ولی با موتورهای بازی نیازی به دوباره کاری نیست. موتورهای بازی در واقع با کاهش دادن کارهای بیهوده، باعث شده توسعه‌ی بازی با سرعت بیشتری انجام شود.

---

[1] Feature
[2] Port
[3] Game Consoles
[4] Multiplatform
[5] Unreal Engine
[6] PC
[7] Play Station
[8] Xbox
[9] Android
[10] iOS

۲-۱-۵ افزایش کیفیت بازی‌ها

تسریع توسعه باعث می‌شود بازی‌سازها زمان بیشتری برای ساخت بازی داشته باشند و بجای اینکه وقت خود را صرف این کنند که فلان قابلیت را چـگونـه به بازی اضافه کنند، به این سوال پاسخ می‌دهند که چـــه قابلیتی را به بازی اضافه کنند. با محتوا محور بودن بازی ها، کیفیت آن ها نیز افزایش می‌یابد؛ چرا که موتوربازی کارهای سخت را برای بازی‌سازها آسان می‌کند.

۲-۱-۶ ایرادیابی[1] آسان‌تر

زمان اختصاص یافته به ایرادیابی بازی‌های ویدئویی حدود ۳۰ درصد زمان کل توسعه‌ی آن است. موتورهای بازی با ساختار درونی که دارند، بصورت خودکار ایرادیابی بازی‌ها را انجام می‌دهند.

۲-۲ ساختار بازی‌ها

ساختن موتورهای بازی‌سازی باعث شد که بازی‌سازها بجای اینکه با کامپایلرها و زبان‌های برنامه‌نویسی سطح پایین دست‌به‌گریبان شوند، نرم‌افزاری آماده در سیستم‌عامل خود داشته باشند.

ساختار بازی‌ها، مانند سایر نرم‌افزارها لایه‌ای است. کل سیستم بازی از بالا به پایین شامل لایه‌های زیر است: (جدول ۲-۱)

کاربر (بازیکن)
بازی
طراح نقشه
ماد
موتور بازی
سیستم عامل
کامپایلر
زبان ماشین

[1] Debugging
[2] Map Editor
[3] Modification

بازیکن در بالای این ساختار قرار دارد و بصورت مستقیم با بازی و غیرمستقیم با Map Editor ها رابطه دارد. در صورت اینکه سازنده‌ی بازی اجازه بدهد، کاربران و مادساز ها می‌توانند با دست بردن در بازی محتوای آن را به شکل محدودی عوض کنند که به این کار ویرایش بازی می‌گویند. از لایه‌ی بعدی که فقط بازی‌سازها به آن دسترسی دارند، عملیات ساخت بازی انجام می‌شود که به لطف موتورهای بازی این کار – برخلاف قبل – در لایه‌ی بالایی سیستم انجام می‌شود. در لایه‌های زیرین هم کار تبدیل به زبان‌های سطح پایین انجام می‌شود.

[1] Modify

فصل سوم

وظایف

۳- وظایف

۳-۱ رندر[1]

در بازی‌سازی رندر کردن به زبان ساده یعنی بازیکن بصورت لحظه‌ای و بلادرنگ چه چیزی را ببیند و اشیا چگونه در بازی نمایش داده شوند. گرچه واژه‌ی رندر بیشتر در محیط‌های سه بعدی دیده می‌شود ولی محیط‌های دوبعدی هم نیاز به رندر اشیا دارند.

فرض کنید میخواهیم در یک بازی سه‌بعدی، یک شخصیت داشته باشیم. پس از طراحی اولیه شخصیت روی کاغذ که به آن هنر مفهومی[2] می‌گوییم، باید یک فرایند را بگذرانیم تا بتوانیم این شخصیت را در بازی نمایش دهیم. برای مثال پس از دیدن کاغذ شخصیت کاراکتری به نام "لینک" و خواندن توضیحات او میخواهیم او را در بازی داشته باشیم. ابتدا در کاغذهای شطرنجی با دقت، کاراکتر را از سه جهت جـلو، پـهلو و پشت رسم می‌کنند که به آن برگه‌ی مدل شخصیت[3] می‌گویند. سپس با کمک آن فرایند مدلسازی شروع می‌شود که شامل مراحل زیر است که آن‌ها را به مختصر شرح میدهیم:

۳-۱-۱ مدلسازی سه‌بعدی کلی با اضلاع کم[4]

اولین مرحله در فرایند مدلسازی، مدلسازی با اضلاع کم است که به این دلیل انجام می‌شود که کارت‌های گرافیکی توانایی رندر سریع و بلادرنگ مدل‌های با اضلاع زیاد[5] را ندارند؛ خصوصا اینکه در بازی‌ها در یک لحظه چندصد مدل بطور همزمان رندر می‌شود. این مرحله تماما بصورت دستی و با توجه به برگه‌ی مدل شخصیت انجام می‌شود و خروجی آن، شخصیت دلخواه ماست ولی از دید سه‌بعدی دارای نقاط تیز و صیقل نشده‌ی زیادی است. (شکل ۳-۱)

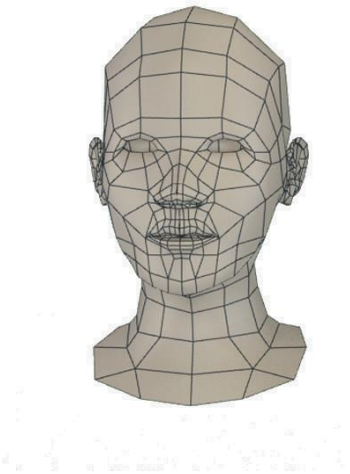
۳-۱-۲ مدلسازی سه‌بعدی جزئی با روش مجسمه‌سازی[6]

در این مرحله مدل مرحله‌ی قبل را صیقل می‌دهیم و تا حد ممکن آن را بصورت جزئی بهبود می‌دهیم. این مرحله با کمک ابزارهای سه‌بعدی "مجسمه‌سازی"

---

[1] Render
[2] Concept art
[3] Character Model Sheet
[4] Low-poly Modeling
[5] High-poly
[6] Sculpting

انجام می‌شود. این ابزارها – همانگونه که از نامشان برمی‌آید – برای پرداختن به جزئیات مدل طراحی شده و خروجی‌اش مدلی کامل و باجزئیات است. (شکل ۲-۳) اما کارت‌های گرافیک قوی هم برای رندر کردن این مدل ها نیاز به زمان زیادی دارند.



شکل ۱-۳ مدل سه‌بعدی شخصیت با اضلاع کم

شکل ۲-۳ مدل سه‌بعدی شخصیت با اضلاع زیاد

۳-۱-۳ پختن مدل جزئی روی مدل با اضلاع کم[1]

دیدیم که در مرحله‌ی قبل مدلی ساختیم که کارت‌های گرافیک قدرت رندر کردن بلاددرنگ آن را ندارند. در این مرحله با استفاده از تکنیک پختن، مدل جزئی را روی مدل با اضلاع کم سوار می‌کنیم به طوری که هم جزئیات قابل‌قبولی داشته باشیم و هم پردازش و رندر آن ها زیاد طول نکشد.

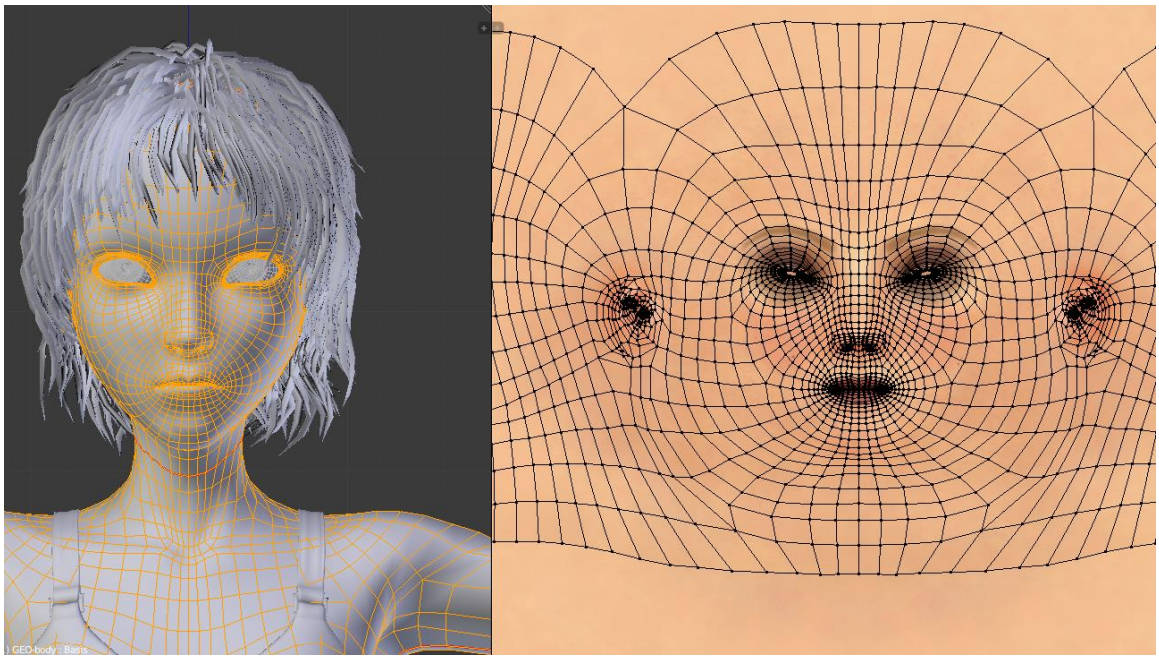
این تکنیک با تبدیل کردن مدل با اضلاع زیاد به یک تصویر نرمال[2] و تصویر ارتفاع[3] ، فقط اطلاعات بازتابش نور و ارتفاع مدل را نگهداری می‌کند. با این کار مدّل موردنظر با اینکه رئوس[4] کمی دارد، طوری نشان می‌دهد که انگار جزئی مدلسازی‌شده در حالی که نمایش این جزئیات فقط با تصویر نرمال و تصویر ارتفاع انجام می‌شوند.

۳-۱-۴ باز کردن مدل سه بعدی روی کاغذ[5]

در این مرحله باید مدل ساخته شده را "برش" دهیم تا بتوانیم آن را روی یک سطح دوبعدی پهن کنیم. یک مکعب را در نظر بگیرید، برای اینکه بتوانیم به آن رنگ و لعاب دهیم باید ابتدا آن را باز کنیم و سپس رنگ را روی سطح صاف پخش کنیم، فرض کنید میخواهیم به آن طرح آجری بدهیم. و نهایتا دوباره مکعب را به شکل اولیه برگردانیم. با این کار تغییر طرح آجری در لبه‌های مکعب منطقی به نظر می‌آیند و انگار طرح ما در وجه‌های دیگر مکعب ادامه دارند و منقطع به نظر نمی‌آیند.

برای شخصیت هم با برش دادن، کاری می‌کنیم که عملیات رنگ و لعاب دادن به او، تسهیل گردد. در شکل ۳-۳ نمونه‌ای از این روش را مشاهده می‌کنید.

[1] Baking
[2] Normal Map
[3] Height Map
[4] Vertex
[5] UV Unwrapping



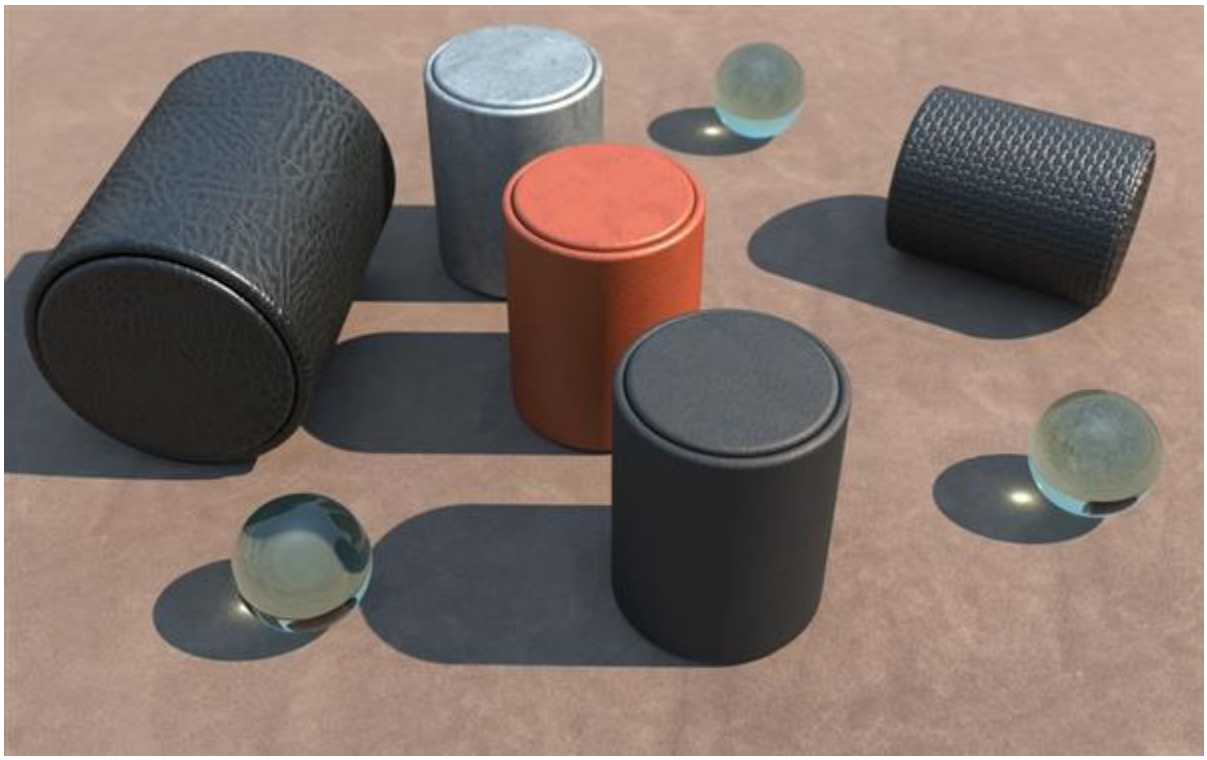
شکل ۳-۳ سمت چپ: مدل سه بعدی – سمت راست: تصویر بازشده‌ی صورت

۳-۱-۵ طراحی بافت[1]

در این مرحله که موازی با مرحله‌ی قبل انجام می‌شود، بافت‌های مورد نیاز برای مدل ساخته می‌شوند. برای مثال فرض کنید شخصیت ما پوستی شبیه به پوست انسان دارد. همانطور که می‌دانید پوست انسان دارای برآمدگی های ریزی است و میزان بازتاب نور از آن با سایر بافت‌ها تفاوت دارد. در این مرحله بافت‌های موردنیاز برای مدل مثل پوست بدن، جنس لباس، جنس مو و ... تعیین می‌شوند که نقش مهمی در نحوه‌ی نمایش مدل در بازی دارد.

در شکل زیر، نمونه ای از بافت‌های پرکاربرد اشیای اطراف را مشاهده می‌کنید.

[1] Material

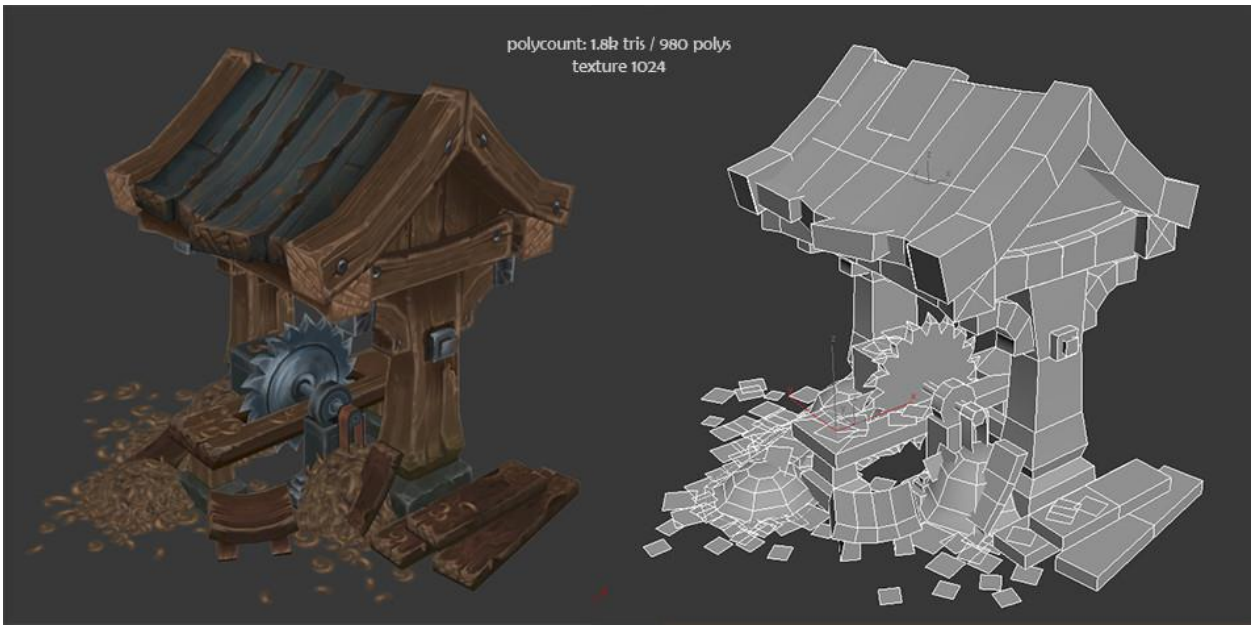


شکل ٤-٣ نمونه‌های پرکاربرد بافت‌های مختلف در
بازی‌های ویدئویی

۳-۱-۶ طراحی تکسچر[1]

در اینجا، تلاش‌هایی که در دو مرحله‌ی قبل کردیم – بازکردن مدل و طراحی بافت – به ثمر می‌نشیند. با طراحی تکسچر تعیین می‌کنیم که پوست طراحی شده در مرحله‌ی قبل چه رنگی باشد و مثلا کجایش سرخ تر باشد. پس از تعیین رنگ لباس و پوست و مو و ... نوبت آن میرسد که با کمک تصویر باز شده، این رنگ‌ها و بافت‌ها را سوار مدل کنیم.

[1] Texture



شکل ۵-۳ مقایسه‌ی قبل و بعد از عملیات
طراحی تکسچر

۳-۱-۷ نورپردازی

این مرحله فقط برای محیط‌ها کاربرد دارد. برعکس صحنه‌های فیلمبرداری و عکس‌برداری که نورپردازی در ابتدای کار انجام می‌شود، در محیط‌های سه‌بعدی این کار در آخرین مرحله انجام می‌شود؛ چون تغییر دادن آن تعیین می‌کند که تصویر نهایی به چه شکل نمایش داده شود.

شکل ٦-٣ تاثیر نورپردازی در رندر نهایی

٣-١-٨ خروجی گرفتن با فرمت دلخواه [1]

در این مرحله مدل کامل را از نرم‌افزار مدلسازی خارج می‌کنیم و آماده‌ی ورود به موتوربازی و بازی می‌شود.

٣-١-٩ رندر

پس از ورود مدل به بازی، با رندر بلادرنگ، مدلی که طراحی کرده بودیم را در بازی به شکل دلخواه می‌بینیم.

[1] Export

۳-۲ فیزیک و برخورد[1]

چیزی که به بازی‌های ویدئویی و پویانمایی ها جان می‌بخشد، حرکت است. در بازی-های ویدئویی حرکت به دو شکل قابل حس است که یکی از آنها فیزیک و برخوردها می‌باشد. در اکثر بازی‌های امروزی دنیای اطراف پویاست و شاید نیمی از این پویایی بخاطر فیزیک محیط باشد. توپی که پرتاب می‌شود، هواپیمایی در که حال سقوط است، تیری که شلیک می-شود و همه‌ی اینها نمونه‌های بارزی از فیزیک در بازی‌ها هستند. ولی فرض کنید روزی از خواب بیدار شوید و ببینید همه‌چیز در حال فرو ریختن است؛ دلیلش فقط یک چیز است: فیزیک.

برای حل این مشکل باید همواره در بازی، تابعی فراخوانی شود که برخورد ها را حس می‌کند. مثلا اینکه خانه درون زمین فرو نرود. این تابع را تشخیص‌دهنده‌ی برخورد[2] می‌نامیم و انجام درست وظیفه‌اش نقش مهمی در تجربه‌ی بازیکن در بازی دارد. فرض کنید در بازی می‌خواهیم با تیر و کمان، چند پرنده شکار کنیم. اینکه تیر چگونه پرتاب می‌شود و پرنده چگونه پرواز – و احتمالا سقوط – می‌کند به فیزیک بازی مربوط است ولی اینکه تیر رها شده به پرنده برخورد کند را تابع تشخیص برخورد شناسایی می‌کند.

در بازی‌های ویدئویی دو نوع فیزیک داریم:

- فیزیک اشیا
- فیزیک بدن

در بازی‌های اخیر، فیزیک کاربردهای بیشتری پیدا کرده‌اند. در سال ۲۰۰۴ که شرکت Valve نسخه‌ی دوم بازی نیمه جان[3] را معرفی کرد، یکی از تمرکزهای اصلی‌اش موتور قدرتمند فیزیکی‌اش بود که کاربردهای فراوانی در بازی داشت. شکستن دیوارها و شیشه‌ها همگی توسط موتور فیزیکی انجام می گرفت. همچنین با معرفی سیستم عروسک کهنه[4] علاوه بر اشیا، شخصیت‌ها هم دارای فیزیک بودند.

[1] Collision

[2] Collision Detection Function

[3] Half life

[4] Ragdoll

نکته‌ی دیگر اینکه موتورهای فیزیکی برخلاف موتورهای رندر به قدرت پردازشی زیادی نیاز دارند، چون تمامی محاسبات ریاضیاتی باید در پردازنده‌ی مرکزی انجام شود و به موتور-بازی بازگردانده شود، خصوصا اینکه در بازی بصورت همزمان اشیای زیادی از فیزیک استفاده می‌کنند.

۳-۳ پویانمایی[1]

در قسمت قبل به این اشاره کردیم که بازی‌های ویدئویی سرشار از حرکت است. حرکت در بازی‌ها به دو شکل فیزیک و یا پویانمایی دیده می‌شود که در این قسمت درباره‌ی پویانمایی بحث می‌کنیم.

پویانمایی قسمتی از حرکت در بازی‌هاست که به‌صورت دستی و توسط انیمیشن‌ساز ها انجام می‌شود. این قسمت شامل حرکاتی است که موتور فیزیکی قادر به ساختن آنها نیست و فیزیک در آن‌ها نقش چندانی ندارد؛ برای مثال راه‌رفتن عملی نیست که بتوان آن را با موتور فیزیکی پیاده‌سازی کرد، پس انیمیشن سازها باید این کار را در موتوربازی انجام دهند.

همانند فیزیک پویانمایی‌ها هم به دو زیردسته تقسیم می‌شوند:

- پویانمایی انسان
- پویانمایی اشیا

که پویانمایی انسان مانند حرکت کردن و حرکات رزمی است و زمان بیشتری می‌برد. پویانمایی اسنان خود به زیردسته هایی مثل پویانمایی چهره و پویانمایی بدن تقسیم می‌شود. پویانمایی اشیا هم در مثال‌هایی از قبیل حرکت دادن چرخ‌های ماشین و باز و بسته شدن درب وسایل نقلیه – که فیزیک به تنهایی قادر به ساختن این حرکات نیست – می‌باشد.

فناوری‌های نوین باعث شده که دیگر پویانمایی بصورت دستی برای شخصیت‌های بازی انجام نگیرد. با استفاده از فناوری موشن کپچر[2] ، پویانمایی‌های صورت و بدن بصورت خودکار در محیط‌های خاص و با سنسورهای خاصی انجام می‌پذیرد. در شکل ۷-۳ نمونه‌ای از این فناوری را می‌بینید.

[1] Animation

[2] Motion Capture

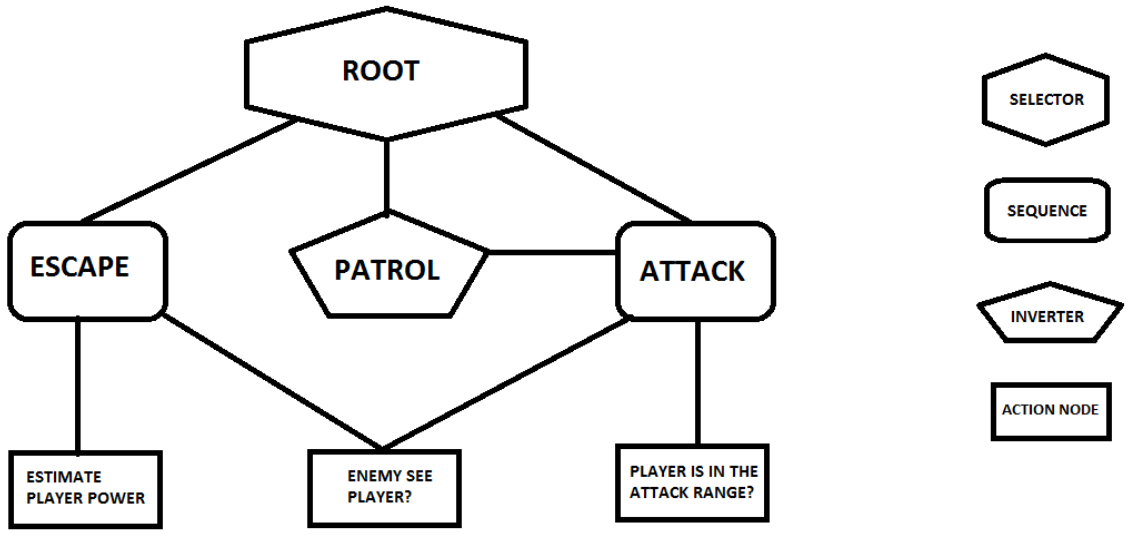
شکل ۷-۳ موشن کپچر حرکات رزمی برای
استفاده‌ی مستقیم در موتوربازی

۳-۴ هوش مصنوعی

در بازی‌های اخیر هوش مصنوعی کاربردهای زیادی پیدا کرده است. هوش مصنوعی در موتورهای بازی که اکثرا در قالب توابعی آماده مورد استفاده قرار می‌گیرند، محیط بازی را پویا و متغیر می‌کند و از بوقوع آمدن مشکلات احتمالی که در کدهای عادی است جلوگیری می‌کند.

یکی از کاربردهای اولیه‌ی هوش مصنوعی در بازی‌های ویدئویی مسیریابی[1] است. در بازی‌های مدرن معمولا محیط بازی پویا است و دائم در حال تغییر است، برای همین اگر شخصیتی[2] که غیر قابل کنترل توسط بازیکن است بخواهد از نقطه‌ی الف به ب برود باید کوتاهترین مسیر بین این دو نقطه را پیدا کند. این جستوجو بصورت آنلاین[3] در بازی انجام می‌گیرد و حلقه‌باز می‌باشد.

کاربرد دیگر هوش‌مصنوعی در بازی‌های مخفی کاری یا احساساتی است. شخصیت های غیرقابل کنترل باید احساس ترس و خشونت را در بازی به بازیکن منتقل کنند تا بازیکن از تجربه‌اش راضی باشد. برای همین به درخت‌های تصمیم پیچیده‌ای در بازی‌ها نیاز است.



نمودار ۱-۳ درخت تصمیم برای یک بازی مخفی کاری

[1] Pathfinding

[2] NPC

[3] Open loop

۳-۵ شبکه

ابزار قدرتمند دیگری که موتوربازی در اختیار توسعه‌دهندگان قرار می‌دهد، ابزار شبکه است. امروزه بازی‌های چندنفره و آنلاین بخش زیادی از بازار بازی‌های ویدئویی را به خود اختصاص داده‌اند و این جهت ابزار شبکه‌ی قدرتمند نقش مهمی در راضی نگه‌داشتن مشتریان ایفا می‌کند.

این ابزار وظایفی از جمله یافتن نزدیک‌ترین سرور برای بازی را در اختیار داردو همینطور پیدا کردن بازی های مناسب برای بازیکنی که در حال جستجو برای بازی‌کردن است از وظایف این ابزار است.

۳-۶ صوت و تصویر

بازی بدون صدا، مثل زنبور بدون عسل می‌ماند. یکی از وظایف مهم موتورهای بازی مدیریت فایل‌های موردنیاز برای بازی است. صدا و تصویر یکی از این فایل ها میباشند. اگر قرار باشد، تمامی این فایل‌ها بدون هیچ حساب قبلی وارد بازی شوند، استفاده‌ی مجدد از آنها و نگهداری آنها به سادگی ممکن نیست.

تصاویر عکس‌هایی هستند که در حین بازی پخش می‌شوند ولی جزو محیط بازی نیستند. همینطور فیلم‌هایی که در بین بازی پخش می‌شود ولی جزوی از محیط بازی نیست از این دسته از فایل‌ها می‌باشد.

صدا ها مانند موسیقی متن بازی که در حین بازی پخش می‌شود. از نمونه‌های دیگر صدا می‌توان به دیالوگ‌های بازی اشاره کرد. کاربرد دیگر صداها در بازی افکت‌های صوتی هستند؛ مانند صدای مشت و گلوله و خوردن توپ به دیوار.

وظیفه‌ی دیگر موتورهای بازی در این باره، رمزنگاری این فایل‌ها است، به طوری که اشخاص خارجی نتوانند به فیلم‌های و موسیقی‌های احتمالا دارای مجوز بازی دسترسی پیدا کنند و موجب مشکلات حقوقی شوند.

---

[1] Matchmaking

[2] Game assets

[3] Cut scene

[4] Sound Effects

۳-۷ رابط کاربری

بازیکن ها علاوه بر محیط بازی نیاز دارند امتیازات و فهرست های دیگری را نیز مشاهده کنند. رابط کاربری در بازی‌های ویدئویی به طور کلی شامل سه زیردسته است:

- نمایشگر سربالا[1] که وضعیت هایی از قبیل سلامتی و امتیاز بازیکن را نمایش می‌دهد.
- فهرست‌های توقف[2] بازی که به بازیکن این انتخاب را می‌دهد که از بین گزینه‌های داده شده یکی را انتخاب کنند.
- نمایشگر دیالوگ‌ها و پیغام‌های دیگری که در بازی لازم است.

شکل ۳-۸ نمایشگر سربالا در یک بازی اکشن که شامل سلامتی و اسلحه‌ی مورد استفاده است

[1] HUD

[2] Pause menu

فصل چهارم

جمع‌بندی

۴- جمع‌بندی

در این بخش به جمع‌بندی و نتیجه‌گیری از مطالب در پیش گفته شده می‌پردازیم.

دیدیم که پیشرفت بازی‌های ویدئویی مدیون قدرت موتورهای بازی‌سازی امروزی است. این موتورها با ابزارهایی که در اختیار بازی‌ساز ها قرار می‌دهند، موجب می‌شوند بازی‌ساز ها روی محتوای بازی تمرکز داشته باشند، تا اینکه روی نحوه‌ی اجرا وقت اضافی صرف کنند؛ چون این کارها رو موتور-بازی با ابزارهایی که دارد به راحتی می‌تواند تسهیل کند.

همچنین بررسی کردیم موتورهای بازی قابلیت انجام چه‌کارهایی را دارند و درحال پیشرفت روزافزون هستند. تا امروز شرکت‌های متفاوت برای کارهای خاص خودشان موتورهای بازی‌سازی اختصاصی زیادی ساخته‌اند که نشان می‌دهد که صنعت بازی‌سازی چقدر به وجود این ابزار وابسته است.

پیشنهاد می‌کنم درباره‌ی انواع موتورهای بازی و کاربرد آنها تحقیق کنید و ببینید بازی‌هایی که خودتان امتحان کرده‌اید با چه موتوی ساخته شده‌اند.

فصل پنجم

منابع

۵- منابع

1. Gregory, J. (2009). Game Engine Architecture. New York: Boca Raton
2. Millington, I.(2007). Game Physics Engine Development: How to Build a Robust Commercial-Grade Physics Engine for Your Game. London : Morgan Kauffman Publishers
3. Millington, I. Funge, J.(2009) Artificial Intelligence for Games. London : Morgan Kauffman Publishers
4. [TheHappieCat]. (2016, September 7). How Game Engines Work [Video File] Retrieved From https://www.youtube.com/watch?v=DKrdLKetBZE
5. [H3Vtux]. (2019, March 10). How do game engines work? [Video File] Retrieved From https://www.youtube.com/watch?v=xrqRuxAkkG8
6. 3D Art Essentials: The Fundamentals of 3D Modeling, Texturing, and Animation
7. Chopline, A. (2012). 3D Art Essentials. London: Taylor and Francis

Amirkabir University of Technology
(Tehran Polytechnic)

Computer Engineering and Information Technology Department

Title of Research
A Simple Approach to Game Engines

By
Amir Hossein Binesh

Supervisor
Dr. Reza Safabakhsh

June 2019