

لَا إِلَهَ إِلَّا اللَّهُ



دانشگاه صنعتی اصفهان
مهندسی برق و کامپیوتر

بررسی سیستم گراف پویانمایی در موتور بازی سازی آنریل و پیااده سازی یک سیستم پویانمایی با استفاده از OpenGL

پایان نامه کارشناسی مهندسی کامپیوتر

نامی نذیری

استاد راهنما
دکتر مازیار پالهنک

تیرماه ۱۴۰۱



دانشگاه صنعتی اصفهان
مهندسی برق و کامپیوتر

پایان نامه کارشناسی رشته مهندسی کامپیوتر آقای نامی نذیری
تحت عنوان

بررسی سیستم گراف پویانمایی در موتور بازی سازی آنریل و
پیاده سازی یک سیستم پویانمایی با استفاده از OpenGL

در تاریخ ۱۴۰۱/۰۴/۲۹ توسط کمیته تخصصی زیر مورد بررسی و تصویب نهایی قرار گرفت:

۱- استاد راهنمای پایان نامه دکتر مازیار پالهنک

۲- استاد داور دکتر زینب زالی

کلیه حقوق مالکیت مادی و معنوی مربوط به این پایان نامه متعلق به دانشگاه صنعتی اصفهان و پدیدآورندگان است. این حقوق توسط دانشگاه صنعتی اصفهان و بر اساس خط مشی مالکیت فکری این دانشگاه، ارزش گذاری و سهم بندی خواهد شد. هر گونه بهره برداری از محتوا، نتایج یا اقدام برای تجاری سازی دستاوردهای این پایان نامه تنها با مجوز کتبی دانشگاه صنعتی اصفهان امکان پذیر است.

فهرست مطالب

عنوان	صفحه
فهرست مطالب.....	شش
چکیده.....	۱
فصل اول: مقدمه	۲
فصل دوم: مرور	۵
۱-۲ تاریخچه‌ی پویانمایی سنتی.....	۶
۱-۱-۲ فریم‌های کلیدی و درمیان.....	۶
۲-۱-۲ چشم‌انداز چندمنظوره.....	۷
۳-۱-۲ لایه‌های مختلف.....	۷
۲-۲ پویانمایی کامپیوتری.....	۸
۱-۲-۲ فریم‌های کلیدی و درمیان.....	۹
۲-۲-۲ روبه.....	۹
۳-۲-۲ مبتنی بر فیزیک.....	۹
۴-۲-۲ ضبط حرکت ^۱	۹
۳-۲ موتور بازی‌سازی.....	۱۰
۴-۲ موتور بازی‌سازی آنریل.....	۱۰
۵-۲ زبان برنامه‌نویسی در آنریل.....	۱۱
۶-۲ گرافیک کامپیوتری.....	۱۱
۱-۶-۲ OpenGL.....	۱۲
۲-۶-۲ سایه‌زنی فانگ.....	۱۲
۷-۲ پویانمایی اسکلتی.....	۱۴
۱-۷-۲ شبکه‌ی ^۲ چندضلعی.....	۱۴
۲-۷-۲ مدل.....	۱۶
۳-۷-۲ زیرمش ^۳	۱۶
۴-۷-۲ ماده ^۴	۱۷
۵-۷-۲ بافت ^۵	۱۷
۶-۷-۲ اسکلت.....	۱۸
۷-۷-۲ Skinning.....	۱۹
۸-۷-۲ ژست شخصیت.....	۱۹
۹-۷-۲ کلیپ‌های پویانمایی.....	۲۰

^۱Motion Capture

^۲Mesh

^۳Sub-Mesh

^۴Material

^۵Texture

۲۰	۱۰-۷-۲ ترکیب کلیپ‌های پویانمایی
۲۲	فصل سوم: سیستم گراف پویانمایی در موتور بازی سازی آنریل
۲۳	۱-۳ بازیگران، پیاده‌ها و شخصیت‌ها
۲۳	۲-۳ اجزاء
۲۴	۳-۳ شخصیت‌ها
۲۴	۴-۳ مولفه‌ی مش اسکلتی
۲۵	۵-۳ طرح پویانمایی
۲۵	۶-۳ گراف رویداد
۲۵	۷-۳ گراف پویانمایی
۲۷	۸-۳ گره‌های گراف پویانمایی
۲۷	۹-۳ ساختار گره‌های گراف پویانمایی
۲۸	۱۰-۳ جریان اجرا در گراف پویانمایی
۲۸	۱۱-۳ دنباله‌ی پویانمایی
۲۹	۱۲-۳ فضای ترکیب
۳۰	۱۳-۳ گره‌های ترکیب
۳۰	۱-۱۳-۳ گره‌ی ترکیب استاندارد
۳۱	۲-۱۳-۳ گره‌ی ترکیب بر اساس یک مقدار
۳۱	۳-۱۳-۳ گره‌ی ترکیب لایه‌ای برای هر مفصل
۳۳	۱۴-۳ گره‌های کنترل اسکلت
۳۳	۱۵-۳ گره‌های تبدیل فضا
۳۳	۱۶-۳ گره‌ی ماشین حالت
۳۴	۱۷-۳ نتیجه‌گیری
۳۵	فصل چهارم: پیاده سازی
۳۵	۱-۴ کتابخانه‌های کمکی
۳۵	GLFW ۱-۱-۴
۳۶	GLAD ۲-۱-۴
۳۶	GLM ۳-۱-۴
۳۶	Assimp ۴-۱-۴
۳۶	stb ۵-۱-۴
۳۷	۲-۴ پیاده‌سازی سیستم پویانمایی
۳۷	۱-۲-۴ نمایش مدل گرافیکی
۳۷	۲-۲-۴ قرارگیری مدل سه‌بعدی در کارت گرافیک
۳۸	۳-۲-۴ سایه‌زنی فانگ
۴۱	۴-۲-۴ اسکلت شخصیت
۴۲	۵-۲-۴ اتصال اسکلت و مدل سه‌بعدی
۴۳	۶-۲-۴ کلیپ پویانمایی
۴۳	۷-۲-۴ پخش‌کننده‌ی کلیپ‌های پویانمایی
۴۴	۸-۲-۴ الگوریتم پخش‌کننده‌ی کلیپ‌های پویانمایی

۴۵.....	۹-۲-۴ ماشین حالت پویانمایی
۴۶.....	۱۰-۲-۴ به روزرسانی ماشین حالت پویانمایی.....
۴۷.....	۳-۴ نتیجه گیری
۴۸	فصل پنجم: نتیجه گیری
۵۰.....	فهرست شکل ها.....
۵۲	مراجع

چکیده

پویانمایی کامپیوتری فرایندی است که برای تولید تصاویر متحرک دیجیتالی استفاده می‌شود. پویانمایی کامپیوتری مدرن معمولاً از گرافیک کامپیوتری سه‌بعدی برای ایجاد یک تصویر سه‌بعدی استفاده می‌کند. در اکثر سیستم‌های پویانمایی کامپیوتری سه‌بعدی یک پویاساز نمایش ساده از آناتومی یک شخصیت ایجاد می‌کند که مشابه یک اسکلت یا آدمک است. در شخصیت‌های انسان و حیوانات اکثر قسمت‌های این مدل اسکلتی با استخوان‌های واقعی مطابقت دارد.

گام اول این پروژه بررسی سیستم گراف پویانمایی در موتور بازی‌سازی آنریل است. گراف پویانمایی برای محاسبه‌ی وضعیت نهایی یک مش اسکلتی در فریم فعلی استفاده می‌شود. به صورت کلی این گراف برای نمونه‌گیری، ترکیب و دستکاری ژست‌ها استفاده می‌شود و این ژست به مش‌های اسکلتی توسط طرح پویانمایی اعمال می‌شود. در این گام به بررسی این گراف و الگوریتم‌های به کار گرفته شده در آن خواهیم پرداخت.

در گام دوم نیز به پیاده‌سازی سیستم پویانمایی اسکلتی از پایه، با توجه به روش‌های بدست آمده پرداخته می‌شود. این مرحله سه هدف را دنبال می‌کند. هدف اول نمایش اسکلتون در یک محیط سه‌بعدی و نورپردازی آن، که با استفاده از OpenGL به وجود آمده، است. در این مرحله باید با استفاده از زبان ++C برنامه‌ای بنویسیم که در نهایت بتواند یک کلیپ پویانمایی به وجود آمده به وسیله‌ی فریم‌های کلیدی را نمایش دهد. هدف دوم اضافه کردن یک مش به اسکلتون با استفاده از روش‌های پوسته سازی است. در هدف نهایی نیز روش ترکیب کلیپ‌های پویانمایی را پیاده‌سازی می‌کنیم تا بتوانیم کلیپ‌های مختلف را با یکدیگر ترکیب کنیم.

کلمات کلیدی

پویانمایی کامپیوتری، موتور بازی‌سازی، موتور آنریل، گراف پویانمایی، گرافیک سه‌بعدی کامپیوتری

فصل اول

مقدمه

پویانمایی هنر جان بخشیدن به اجسام بدون جان است. والت دیزنی درباره‌ی پویانمایی می‌گوید: ”پویانمایی می‌تواند هر آنچه را که ذهن انسان تصور می‌کند، توضیح دهد“

وقتی می‌گوییم جسمی را پویا کردیم، یعنی به آن جان بخشیدیم. زمانی که یک فیلم پویانمایی شده را در تلوزیون یا سینما می‌بینید، شخصیت‌های درون آن فیلم در حالت حرکت هستند. این حرکت معمولاً صاف و به هم پیوسته است. نوارهای حاوی فیلم متشکل از دنباله‌ای از تصاویر هستند که به عنوان ”فریم“ شناخته می‌شوند و درواقع با پخش شدن این فریم‌ها به صورت متوالی، توهم ایجاد حرکت به مخاطب منتقل می‌شود.

پویانمایی از گذشته تا امروز تغییرات فراوانی را دیده است. در پویانمایی سنتی، تصاویر به وسیله‌ی دست روی صفحات سلولوئیدی شفاف ترسیم یا نقاشی شده سپس از آن‌ها عکس گرفته و روی فیلم نمایش داده می‌شدند. امروزه اکثر پویانمایی‌ها با تصاویر کامپیوتری^۱ ساخته می‌شوند [۱].

علاوه بر این، دامنه‌ی استفاده از این پویانمایی نیز دستخوش بسیاری تغییرات شده است. در گذشته پویانمایی را می‌توانستیم در فیلم‌های پویانمایی شده یا کارتون‌ها مشاهده کنیم. اما اکنون با پیشرفت فناوری، پویانمایی نقش بسیار اساسی‌ای در بازی‌های کامپیوتری پیدا کرده است. هدف بازی‌های کامپیوتری، به خصوص بازی‌های کامپیوتری داستان

^۱CGI

محور، غوطه‌ور کردن بازیکن در داستان است. همانطور که اشاره شد پویانمایی هنر جان بخشیدن به اجسام است و به وسیله‌ی آن است که می‌توانیم احساسات و اعمال شخصیت بازی را به بازیکن منتقل کنیم.

بازی‌های کامپیوتری به صورت معمول توسط موتورهای بازی‌سازی ساخته می‌شوند. اگر بخواهیم تعریفی برای موتور بازی‌سازی آوریم می‌توان گفت آن‌ها پلتفرم‌هایی هستند که ساخت بازی‌های کامپیوتری را آسان‌تر می‌کنند. موتورهای بازی‌سازی متشکل از مولفه‌های مختلفی هستند که قابلیت‌های لازم برای ساخت بازی را فراهم می‌کنند. از رایج‌ترین مولفه‌های موتور بازی می‌توان به مولفه‌ی صدا، مولفه‌ی رندر، مولفه‌ی هوش مصنوعی و مولفه‌ی پویانمایی اشاره کرد [۲].

هدف اصلی این پروژه آشنایی با روش‌های استفاده شده در محیط‌های گرافیکی مانند موتورهای بازی‌سازی با تاکید بیشتر بر روی سیستم‌های پویانمایی به کار رفته در این محیط‌ها است.

به همین جهت این پروژه به دو صورت این هدف را دنبال می‌کند. جهت آشناشدن با یک موتور بازی‌سازی و نحوه‌ی پیاده‌سازی سیستم پویانمایی آن، موتور بازی‌سازی آنریل انتخاب شده است. آنریل یکی از معروف‌ترین موتورهای بازی‌سازی در جهان است که اولین نسل آن توسط تیم سوینی، بنیانگذار اپیک گیمز^۱، توسعه یافت. آخرین نسخه‌ی این موتور به اسم موتور بازی‌سازی آنریل ۵ در سال ۲۰۲۰ معرفی و در سال ۲۰۲۲ انتشار یافت. سیستم پویانمایی این موتور بسیار وسیع است. به همین دلیل بخش کوچکی از این سیستم که گراف پویانمایی نام دارد، انتخاب شده و به بررسی ساختار و نحوه‌ی استفاده از این گراف می‌پردازیم.

پس از بدست آوردن تجربه‌ی اولیه از گراف پویانمایی برای آشنایی کامل تر با محیط گرافیکی و همچنین سیستم پویانمایی، به پیاده‌سازی یک سیستم پویانمایی با استفاده از OpenGL پرداختیم. OpenGL یک واسط برنامه نویسی کاربردی^۲ است که با فراهم کردن توابع مختلف به توسعه‌دهندگان امکان دستکاری گرافیک و تصاویر را می‌دهد. با استفاده از این API می‌توان آشنایی خوبی در مورد گرافیک کامپیوتری و به صورت کلی محیط‌های گرافیکی بدست آورد. برای محیط سه‌بعدی پیاده‌سازی شده از روش سایه‌زنی فانگ برای نورپردازی محیط استفاده شده است. این روش یکی از معروف‌ترین روش‌های نورپردازی در محیط‌های سه‌بعدی بلادرنگ به‌خصوص بازی‌های کامپیوتری است. علاوه بر تولید صحنه‌ی سه‌بعدی، برای بدست آوردن آشنایی کامل با سیستم‌های پویانمایی که در بازی‌ها استفاده می‌شوند، به پیاده‌سازی یک نمونه از آن پرداختیم. در این پیاده‌سازی سیستم پویانمایی به چند بخش کلی تقسیم شده است که هر کدام هدف‌های مختلفی را دنبال می‌کند. برای اینکه یک سیستم پویانمایی داشته باشیم در ابتدا به یک شخصیتی نیاز داریم تا کلیپ‌های پویانمایی بر روی آن اجرا شود. شخصیت‌ها در این پیاده‌سازی توسط کتابخانه‌ی Assimp در ساختمان داده‌های مناسب ذخیره می‌شوند. هر شخصیت در این پیاده‌سازی به دو قسمت کلی مش و اسکلت

^۱Epic Games

^۲API

تقسیم می‌شود. یکی از وظایف مهم این پیاده‌سازی، اتصال این دو قسمت به یکدیگر است. این اتصال به صورت کلی به اسم Skinning نام دارد. مرحله‌ی بعدی پیاده‌سازی به پخش کلیپ‌های پویانمایی بر روی این شخصیت می‌پردازد. درنهایت از ماشین حالت متناهی برای برای ترکیب کلیپ‌های پویانمایی متفاوت با یکدیگر استفاده شده است. خروجی این پروژه یک تحقیق در مورد سیستم گراف پویانمایی آنریل به همراه یک نرم‌افزار گرافیکی سیستم پویانمایی است.

در فصل‌های آتی به بررسی این موارد گفته‌شده پرداخته می‌شود. ابتدا در فصل دوم یک مروری بر تاریخچه‌ی پویانمایی می‌شود. سپس توضیحاتی درباره‌ی موتور بازی‌سازی و موتور بازی‌سازی آنریل داده می‌شود و درنهایت توضیحاتی کلی درباره‌ی روش‌های نورپردازی محیط و پویانمایی اسکلتونی که به وفور در موتورهای بازی‌سازی استفاده می‌شود، داده می‌شود.

در فصل سوم به بررسی موتور بازی‌سازی آنریل با تاکید بر روی گراف پویانمایی می‌پردازیم و نحوه‌ی استفاده از آن را بررسی می‌کنیم.

در نهایت در فصل چهارم توضیحاتی درباره‌ی نحوه‌ی پیاده‌سازی سیستم پویانمایی به همراه توضیحات سیستم‌های موجود در این پیاده‌سازی می‌پردازیم.

درنهایت در فصل "نتیجه‌گیری"، یک نتیجه‌گیری کلی از خروجی‌های این پروژه ارائه کرده و به بررسی مشکلاتی که می‌توانند در پیاده‌سازی برطرف شوند به همراه پیشنهاداتی برای ادامه‌ی این پروژه پرداخته می‌شود.

فصل دوم

مرور

از گذشته تا امروز نحوه‌ی تولید و همچنین استفاده از کلیپ‌های پویانمایی تغییرات زیادی کرده است. در گذشته اکثر پویانمایی‌ها با استفاده از دست و ترسیم بر روی صفحات سلولوئیدی تولید می‌شدند. در صورتیکه که امروزه، اکثر کلیپ‌های پویانمایی توسط تصاویر گرافیکی کامپیوتری ساخته می‌شوند. اگرچه با پیشرفت فناوری، روش‌های جدیدی برای تولید پویانمایی به وجود آمده، همچنان نیز از اکثر روش‌هایی که در پویانمایی سنتی استفاده می‌شد، در پویانمایی کامپیوتری نیز استفاده می‌شود.

امروزه نه تنها از پویانمایی برای تولید فیلم و کارتون استفاده می‌شود، بلکه آن‌ها نقش تاثیرگذاری در بازی‌های کامپیوتری پیدا کرده‌اند. برای تولید بازی‌های کامپیوتری از موتورهای بازی‌سازی استفاده می‌شود. موتورهای بازی‌سازی دارای مولفه‌های متفاوتی هستند که هر کدام وظیفه‌ی مخصوص به خودشان را دارند. مولفه‌ای که وظیفه‌ی پخش کلیپ‌های پویانمایی را دارد، مولفه‌ی پویانمایی گویند. بدیهی است اگر بخواهیم از سیستم پویانمایی استفاده کنیم نیازمند یک محیط گرافیکی هستیم. محیط‌های گرافیکی به صورت معمول از کارت گرافیک استفاده می‌کنند. برای دسترسی به توابع کارت گرافیکی می‌توان از یک واسط برنامه‌نویسی کاربردی مانند DirectX یا OpenGL استفاده کرد. علاوه بر این، برای مشاهده‌ی اشیاء موجود در محیط سه‌بعدی نیاز است که محیط را نورپردازی کنیم. روش‌های متنوعی برای نورپردازی محیط وجود دارند. یکی از این روش‌ها سایه‌زنی فانگ است که در این پروژه نیز استفاده شده

است.

در اکثر سیستم‌های پویانمایی بازی‌های کامپیوتری از پویانمایی اسکلتی برای پخش و ترکیب کلیپ‌های پویانمایی استفاده می‌شود.

پویانمایی اسکلتی تکنیکی در پویانمایی کامپیوتری است که به وسیله‌ی آن، شخصیت‌های درون بازی متحرک می‌شوند. این سیستم به دو بخش کلی تقسیم می‌شود. یک بخش، یک مش یا پوسته است که برای به نمایش کشاندن شخصیت در محیط سه‌بعدی استفاده می‌شود و بخش دوم یک اسکلت است. این اسکلت مجموعه سلسله مراتبی از قطعات به هم پیوسته است که به هر قطعه یک مفصل گویند. در این تکنیک، اسکلت شخصیت درون بازی متحرک شده و با روش‌های موجود، پوسته یا مش آن شخصیت، اسکلت را دنبال می‌کند.

در این فصل، ابتدا مروری بر پویانمایی سنتی و روش‌های تولید آن می‌کنیم. سپس نگاهی به پیشرفت صنعت تولید پویانمایی کامپیوتری می‌اندازیم و روش‌های فعلی تولید پویانمایی را خواهیم گفت. پس از آن توضیحی در رابطه با موتورهای بازی‌سازی و به ویژه موتور آنریل خواهیم داد.

در قسمت انتهایی توضیحی در مورد سایه‌زنی فانگ که یکی از روش‌های نورپردازی محیط سه‌بعدی است خواهیم داد و علاوه بر آن در انتها توضیحات جامعی را در مورد پویانمایی اسکلتی ارائه خواهیم کرد.

۲-۱ تاریخچه‌ی پویانمایی سنتی

پویانمایی سنتی که با اسم‌های مختلفی مانند "پویانمایی مرسوم"، "پویانمایی سل‌ای" و "پویانمایی بادست" شناخته می‌شود، روشی غالب برای تولید فیلم‌های پویانمایی شده در حدود قرن ۲۰ میلادی بود. در این روش، به صورت کلی پویانمایی به وسیله‌ی نقاشی با دست به وجود می‌آید. درواقع هر فریم از فیلم، یک عکس از نقاشی است و برای به وجود آوردن توهم حرکت، هر نقاشی اندکی با نقاشی قبلی خود تفاوت دارد.

برای تولید پویانمایی سنتی، از روش‌های مختلفی استفاده می‌شد. در اینجا به بررسی سه روش استفاده شده در پویانمایی سنتی می‌پردازیم.

۲-۱-۱ فریم‌های کلیدی و درمیان

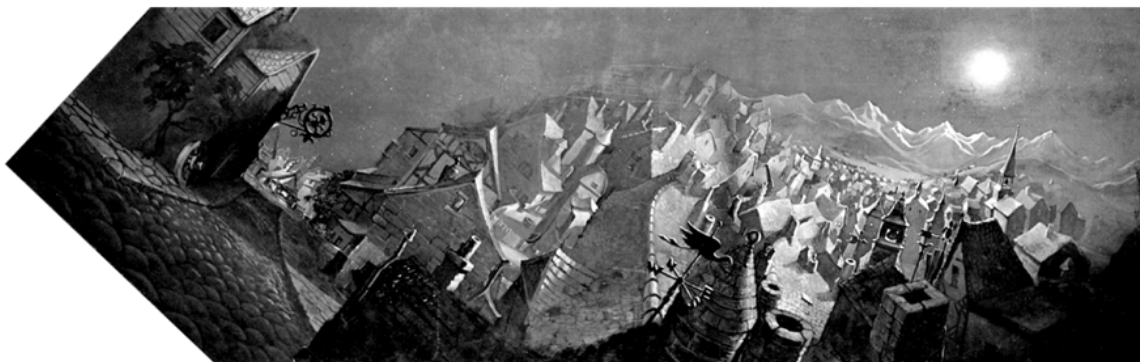
از آنجایی که تولید پویایی با دست و کشیدن نقاشی کار بسیار طولانی‌ای بود، برای اینکه وقت پویاسازهای ارشد ذخیره شود، این پویاسازها فریم‌های اصلی یک حرکت را بر روی کاغذ ترسیم می‌کردند و فریم‌های میانی را پویاسازهای جوان پر می‌کردند [۳]. این روش را می‌توان در شکل ۲-۱ مشاهده کرد. قسمت بالای شکل، فریم‌های کلیدی را نشان می‌دهد که توسط پویاسازهای ارشد کشیده می‌شوند و قسمت پایینی توسط پویاسازهای جوان تر پر می‌شوند.



شکل ۱-۲ - فریم های کلیدی و درمیان [۴]

۲-۱-۲ چشم انداز چندمنظوره

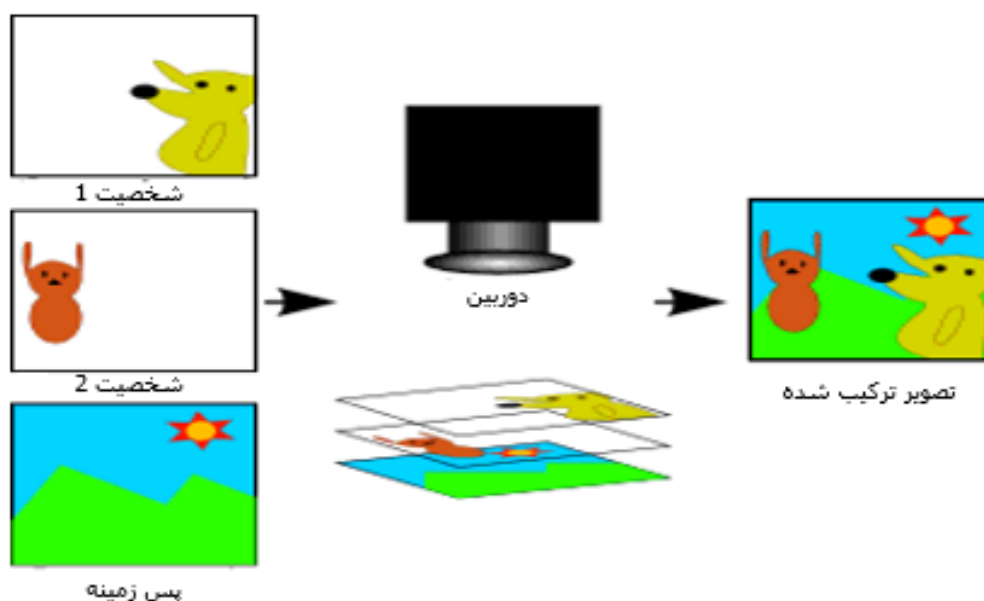
استفاده از چشم انداز چندمنظوره روش دیگری بود که در پویانمایی سنتی استفاده می شد. همانطور که از شکل ۲-۲ مشخص است، برای نمایش یک محیط از یک چشم انداز استفاده می شد. این چشم انداز می توانست نشان دهنده ی قرارگیری محیط در فواصل مختلف باشد. در این صورت، زمانی که دوربین در صحنه حرکت می کرد این توهّم را در مخاطب ایجاد می کرد که گویی در محیط در حال حرکت هستیم [۳].



شکل ۲-۲ - چشم انداز چندمنظوره [۵]

۳-۱-۲ لایه های مختلف

با استفاده از این روش، پویاسازها یک صحنه را به چند قسمت مختلف تقسیم می کردند. به عنوان مثال لایه های مختلفی برای هر شخصیت درون صحنه استفاده می شد. علاوه بر این یک لایه نیز برای تصویر پس زمینه استفاده می شد. از آنجایی که این لایه ها یک صفحه ی شفاف هستند، بنابراین می توان لایه ها را بر روی هم انباشته کرد و با تصویر برداری از بالا، تمام صحنه را تصویربرداری کرد [۳]. این روش در شکل ۳-۲ آورده شده است.



شکل ۲-۳- لایه‌های مختلف [۶]

۲-۲ پویانمایی کامپیوتری

اگر بخواهیم نگاهی به تاریخچه‌ی پویانمایی کامپیوتری بیاندازیم، مشاهده می‌کنیم که در حدود دهه‌ی ۱۹۸۰ میلادی شرکت دیزنی به عنوان یکی از اولین شرکت‌های جهان، شروع به دیجیتالی کردن خط لوله‌ی تولید پویانمایی سنتی خود کرد. در این دیجیتال‌سازی بسیاری از روش‌ها و ایده‌های استفاده شده در پویانمایی سنتی، به کار گرفته شد. اولین مقالات این حوزه توسط آقای جان لستر از کارمندان پیکسار به عنوان "اصول پویانمایی سنتی به کار رفته در پویانمایی کامپیوتری سه بعدی" ارائه شد. در این مقاله اصول اولیه پویانمایی سنتی دوبعدی ترسیم شده با دست و کاربرد آن‌ها در پویانمایی کامپیوتری سه بعدی شرح داده شده است [۷].

پویانمایی کامپیوتری تنها محدود به دنیای سینما و فیلم‌های پویانمایی نمی‌شوند و به دنیای بازی‌های کامپیوتری نیز ورود پیدا کرده‌اند. بازی‌های کامپیوتری سعی می‌کنند دیوار میان تماشاگران و فیلم را بشکنند و با تعاملی بودن و دادن آزادی عمل به بازیکن، سعی می‌کنند داستان را به گونه‌ای تعریف کنند که گویی بازیکن یکی از شخصیت‌های اصلی داستان است. پویانمایی در بازی‌های کامپیوتری اهمیت بسیار بالایی دارد زیرا همانطور که گفته شد باعث جان بخشیدن به شخصیت‌ها می‌شود که اهمیت بسیار بالایی برای جلب توجه بازیکنان در هنگام داستان‌سرایی دارد.

با پیشرفت فناوری، همراه با استفاده از روش‌های گذشته، روش‌های جدیدتری برای تولید پویانمایی توسعه یافته‌است که در ادامه به چند مورد از آن‌ها می‌پردازیم.

۱-۲-۲ فریم‌های کلیدی و درمیان

همانطور که اشاره شد در پویانمایی کامپیوتری از روش‌های موجود در پویانمایی سنتی استفاده شده است. در اینجا نیز فریم‌های کلیدی یک حرکت، توسط پویاسازها به وجود می‌آیند ولی فریم‌های میانی به جای اینکه توسط پویاسازها به وجود آیند، توسط کامپیوتر با استفاده از روش‌های درونیابی به وجود می‌آیند [۳].

۲-۲-۲ رویه

در این روش، حرکت بر اساس یک الگوریتم بیان می‌شود. درواقع کلیپ‌های پویانمایی در این نوع پویانمایی، توابعی با تعداد کمی از متغیرها هستند. به عنوان مثال یک تابعی را در نظر بگیرید که با گرفتن ورودی ثانیه، دقیقه و ساعت، یک شیء ساعت را خروجی دهد که عقربه‌هایش در جای مناسب با توجه به ورودی‌ها قرار گرفته باشد. حال می‌توان با تغییر ورودی‌ها حرکت ساعت را شبیه‌سازی کنیم [۳].

۳-۲-۲ مبتنی بر فیزیک

پویانمایی مبتنی بر فیزیک پلی میان دنیای پویانمایی با دنیای واقعی است. در این روش با نسبت دادن ویژگی‌های فیزیک به اشیاء سه‌بعدی و سپس حل کردن فرمول‌های فیزیک مانند فرمول حرکت یا فرمول‌های نیوتن، فیزیک شبیه سازی می‌شود. پویانمایی‌های مبتنی بر فیزیک شخصیت را قادر می‌سازد تا حرکت‌های خود را به صورت پویا با محیط تنظیم کند [۳].

۴-۲-۲ ضبط حرکت^۱

به فرآیند ثبت و دیجیتالی کردن حرکت یک شیء یا شخص، ضبط حرکت گویند. ضبط حرکت توسط دوربین‌های مادون قرمز که تعدادی زیادی از آن‌ها در صحنه‌ی ضبط قرار دارند، صورت می‌گیرد. این دوربین‌ها به صورت شبکه به یکدیگر متصل هستند و پس از کالیبره شدن، آماده‌ی استفاده هستند. این دوربین‌ها با استفاده از نشانگرهای سفیدی که بر روی لباس بازیگران ضبط حرکت قرار دارد، داده‌های مورد نیازشان را دریافت می‌کنند. قابل ذکر است این نشانگرها بازتابنده‌ی مادون قرمز هستند که توسط دوربین‌ها دریافت می‌شود. در نهایت پویاسازها به پاکسازی و پردازش این داده‌ها پرداخته تا آن را برای استفاده‌ی شخصیت‌های سه بعدی آماده کنند [۳].

^۱ Motion Capture

۳-۲ موتور بازی سازی

موتورهای بازی سازی پلتفرم‌هایی هستند که ساخت بازی‌های کامپیوتری را آسان‌تر می‌کنند. آن‌ها به شما این امکان را می‌دهند تا عناصر بازی مانند پویانمایی، تعامل با کاربر یا تشخیص برخورد میان اشیاء را در یک واحد ادغام و ترکیب کنید [۲]. زمانی که از اصطلاح موتور بازی سازی استفاده می‌کنیم منظورمان نرم‌افزارهای قابل توسعه‌ای هستند که می‌توانند پایه و اساس بسیاری از بازی‌های مختلف باشند [۸]. موتورهای بازی سازی متشکل از اجزای مختلفی هستند که قابلیت‌های لازم برای ساخت بازی را فراهم می‌کنند. رایج‌ترین اجزای موتور بازی عبارتند از: [۲]

- مولفه‌ی صدا: نقش اصلی این مولفه تولید جلوه‌های صوتی در بازی است.
- موتور رندر: وظیفه اصلی این مولفه تبدیل داده‌های ورودی به پیکسل‌ها، برای به تصویر کشاندن بر روی صفحه است.
- مولفه هوش مصنوعی: این مولفه مسئولیت ارائه‌ی تکنیک‌هایی برای تعریف قوانین رفتار شخصیت‌هایی را دارد که توسط بازیکنان کنترل نمی‌شوند.
- مولفه پویانمایی: نقش اصلی این مولفه اجرای کلیپ‌های پویانمایی مختلف مانند حرکت است.
- مولفه شبکه: وظیفه اصلی این مولفه قادر ساختن بازی همزمان بازیکنان با یکدیگر، از طریق استفاده از دستگاه‌های متصل به اینترنت است.
- مولفه منطق یا مکانیک بازی: این مولفه قوانین حاکم بر دنیای مجازی، ویژگی‌های شخصیت‌های بازیکنان، هوش مصنوعی و اشیاء موجود در دنیای مجازی و همچنین وظایف و اهداف بازیکنان را تعریف می‌کند.
- ابزارهای نرم‌افزاری: وظیفه اصلی این ابزارها افزایش راندمان و سرعت تولید بازی با موتور بازی سازی است. آن‌ها توانایی اضافه کردن بسیاری از عناصر مختلف را به بازی‌ها، از پویانمایی و جلوه‌های صوتی گرفته تا الگوریتم‌های هوش مصنوعی، را فراهم می‌کنند.
- یکی از مهم‌ترین مولفه‌های موجود در هر موتور بازی، مولفه‌ی پویانمایی آن است. در این پروژه به بررسی سیستم گراف پویانمایی که وظیفه‌ی پخش کلیپ‌های پویانمایی شخصیت‌های سه‌بعدی را در موتور بازی آنریل دارد می‌پردازیم.

۴-۲ موتور بازی سازی آنریل

اولین نسل موتور بازی سازی آنریل توسط تیم سوینی، بنیانگذار اپیک گیمز^۱، توسعه یافت. سوینی در سال ۱۹۹۵ شروع به نوشتن این موتور برای تولید بازی تیراندازی اول شخصی به اسم غیرواقعی^۲ کرد.

^۱Epic Games

^۲Unreal

نسخه‌ی دوم موتور بازی‌سازی آنریل در سال ۲۰۰۲ منتشر شد.

نسخه سوم نیز در سال ۲۰۰۴ پس از حدود ۱۸ ماه توسعه، منتشر شد. در این نسخه، معماری پایه‌ای موجود در نسخه‌ی اول مانند طراحی شی گرا، اسکریپت‌نویسی مبتنی بر داده و رویکرد نسبتاً ماژولار نسبت به زیرسیستم‌ها وجود داشت. اما برخلاف نسخه دوم که از یک خط لوله با عملکرد ثابت^۱ استفاده می‌کرد، این نسخه به صورتی طراحی شده بود تا بتوان قسمت‌های سایه‌زنی سخت‌افزاری^۲ را برنامه‌نویسی کرد.

موتور بازی‌سازی آنریل ۴ در سال ۲۰۱۴ در کنفرانس توسعه‌دهندگان بازی^۳ منتشر شد. این نسخه با طرح کسب‌وکار اشتراکی برای توسعه‌دهندگان در دسترس قرار گرفت. این اشتراک به صورت ماهانه، با پرداخت ۱۹ دلار آمریکا به توسعه‌دهندگان این اجازه را می‌داد تا به نسخه‌ی کامل موتور، از جمله کد منبع C++ آن دسترسی پیدا کنند. البته در سال ۲۰۱۵ اپیک گیمز موتور بازی‌سازی آنریل را به صورت رایگان برای همگان منتشر ساخت.

آخرین نسخه آنریل به اسم موتور بازی‌سازی آنریل ۵ در سال ۲۰۲۰ معرفی شد. این نسخه از تمام سیستم‌های موجود از جمله کنسول‌های نسل بعدی پلی‌استیشن ۵^۴ و ایکس‌باکس سری X/S^۵ پشتیبانی می‌کند. کار بر روی این موتور حدود دو سال قبل از معرفی آن شروع شده بود. در سال ۲۰۲۱ نسخه‌ای از آن به صورت دسترسی اولیه منتشر شد. به طور رسمی در سال ۲۰۲۲ نسخه‌ی کامل این موتور برای توسعه‌دهندگان انتشار یافت [۹].

۲-۵ زبان برنامه‌نویسی در آنریل

موتور بازی‌سازی آنریل از زبان برنامه‌نویسی C++ به همراه اسکریپت بصری به نام Blueprint استفاده می‌کند. Blueprint یک سیستم برنامه‌نویسی کامل گیمپلی مبتنی بر مفهوم استفاده از رابط‌های مبتنی بر گره برای ایجاد عناصر گیمپلی از درون ویرایشگر است. این سیستم بسیار منعطف و قدرتمند است زیرا این توانایی را در اختیار طراحان قرار می‌دهد تا از طیف گسترده‌ای از مفاهیم و ابزارها که عموماً فقط در دسترس برنامه‌نویسان هستند استفاده کنند [۱۰].

۲-۶ گرافیک کامپیوتری

گرافیک کامپیوتری زیرشاخه‌ای از علوم کامپیوتر است که روش‌های ترکیب دیجیتال و دستکاری محتوای بصری را مطالعه می‌کند. اغلب اوقات این اصطلاح به مطالعه‌ی گرافیک کامپیوتری سه‌بعدی اشاره دارد [۱۱]. گرافیک کامپیوتری را می‌توان در موارد مختلفی مانند طراحی رابط کاربری، رندر اشیاء هندسی، پویانمایی و بسیاری موارد دیگر

^۱fixed-function pipeline

^۲shader hardware

^۳GDC

^۴PlayStation 5

^۵Xbox Series X/S

استفاده کرد. ابزارهای مختلفی برای پیاده‌سازی گرافیک کامپیوتری استفاده می‌شوند. یکی از این ابزارها OpenGL است. در این پروژه برای ایجاد محیط گرافیکی از OpenGL استفاده شده است. برای اینکه بتوان شخصیت‌ها و محیط سه‌بعدی را به وضوح مشاهده کرد از یک الگوریتم نورپردازی به نام سایه‌زنی فانگ استفاده شده است. در این بخش توضیح کوتاهی درباره‌ی OpenGL و این روش سایه‌زنی آورده شده است.

۱-۶-۲ OpenGL

OpenGL یک واسط برنامه نویسی کاربردی^۱ است که با فراهم کردن توابع مختلف به توسعه‌دهندگان امکان دستکاری گرافیک و تصاویر را می‌دهد. OpenGL یک کتابخانه‌ی رندرینگ است. یک "شیء" به خودی خود در OpenGL مفهومی ندارد و به صورت مجموعه‌ای از مثلث‌ها و حالات مختلف در نظر گرفته می‌شود. بنابراین وظیفه‌ی ما است که بدانیم چه شیء‌ای در کدام قسمت صفحه رندر شده است. این کتابخانه تنها وظیفه‌اش، کشیدن تصاویری که است که می‌خواهیم به تصویر کشیده شوند. در این صورت اگر می‌خواهیم تصویری را به‌روزرسانی کنیم و یا به عنوان مثال شیء‌ای را متحرک کنیم باید به OpenGL درخواست دهیم که صحنه را دوباره برای ما رندر کند [۱۲]. به صورت کلی OpenGL را می‌توان یک ماشین حالت بزرگ در نظر گرفت. هر حالت شامل مجموعه‌ای از متغیرها است که نحوه‌ی عملکرد OpenGL را مشخص می‌کند. به مجموعه‌ی این حالت‌ها OpenGL context نیز می‌گویند. در واقع context را می‌توان یک شیء در نظر گرفت که کل OpenGL را دربر می‌گیرد. عموماً تمامی تغییرات، روی context فعلی اعمال می‌شود و سپس رندر می‌شود [۱۲] [۱۳].

۲-۶-۲ سایه‌زنی فانگ

نورپردازی در دنیای واقعی بسیار پیچیده است و به عوامل بسیار زیادی بستگی دارد. با توجه به قدرت محدود پردازش، برای ما چنین امکانی وجود ندارد که رفتار آن را به صورت کامل تخمین بزنیم. بنابراین برای نورپردازی محیط‌های سه‌بعدی از تقریب واقعیت با استفاده از مدل‌های ساده‌شده‌ی فیزیکی استفاده می‌شود. یکی از این روش‌ها مدل سایه‌زنی فانگ نام دارد. این مدل بر اساس سه مولفه‌ی اصلی عمل می‌کند. این سه مولفه، نور محیطی^۲، نور پخش شده^۳ و نور آینه‌وار^۴ نام دارند [۱۴].

^۱API

^۲Ambient light

^۳Diffuse light

^۴Specular light

نور محیطی

نور معمولاً از یک منبع نور منفرد ساطع نمی‌شود، بلکه از منابع نوری زیادی که در اطراف ما پراکنده شده‌اند، حتی زمانی که به صورت مستقیم قابل مشاهده نیستند، نشأت می‌گیرد. حتی زمانی که هوا تاریک است، معمولاً هنوز مقداری نور در جایی در جهان وجود دارد. مانند ماه در هنگام شب. بنابراین اجسام تقریباً هرگز به صورت کامل تاریک نیستند. اگر بخواهیم چنین مولفه‌ای را به صورت واقعی مدل‌سازی کنیم، الگوریتمی بسیار پرهزینه خواهد بود. به همین جهت در مدل فانگ برای اینکه بتوانیم نور محیطی را در اجسام سه بعدی مشاهده کنیم، از یک رنگ ثابت کوچک نور استفاده می‌کنیم و آن را به رنگ نهایی هر شی اضافه می‌کنیم. در این صورت به نظر می‌رسد که همیشه مقداری نور پراکنده در محیط سه بعدی وجود دارد [۱۴].

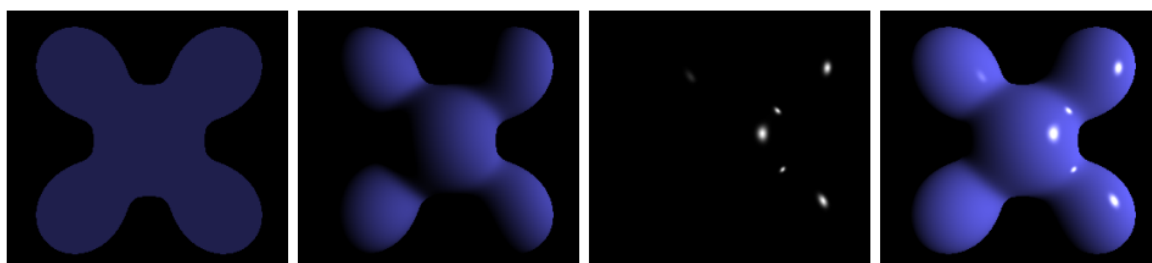
نور پخش شده

می‌دانیم هرچه جسم به یک منبع نور نزدیک تر باشد و هرچه بخشی از یک جسم بیشتر به سمت منبع نور باشد، بیشتر روشن می‌شود. این مولفه، تاثیر جهت قرار گیری اجسام نسبت به منبع نور را شبیه‌سازی می‌کند [۱۴].

نور آینه‌وار

این مولفه وظیفه‌ی شبیه‌سازی نقطه‌ی روشن نوری که بر روی اجسام براق ظاهر می‌شوند، را دارد. جهت قرار گیری اجسام نسبت به جهت نور تاثیر زیادی بر نحوه‌ی شکل گیری و شمایل این نقطه‌ی روشن شده دارد. همچنین نحوه‌ی عملکرد نورپردازی آینه‌وار رابطه‌ی مستقیمی با جنس سطح و خواص بازتابی آن سطح دارد. هرچه سطح اجسام به جنس آینه‌ای نزدیک شود، نور بیشتری را بازتاب می‌دهد و بالعکس ممکن است جنس آن مانند گچی باشد که نور زیادی را جذب خود می‌کند [۱۴].

در تصویر ۴-۲ می‌توانیم عملکرد تمامی این مولفه‌ها را در مدل فانگ مشاهده کنیم.



نورپردازی فانگ = نور آینه‌وار + نور پخش شده + نور محیطی

شکل ۴-۲ - مولفه‌های الگوریتم سایه‌زنی فانگ [۱۵]

۷-۲ پویانمایی اسکلتی

در پویانمایی اسکلتی از مدل‌های اسکلتی استفاده می‌شود. هر مدل اسکلتی از دو بخش مدل و اسکلت تشکیل شده است. برای اینکه بتوانیم پویانمایی اسکلتی را متوجه شویم، نیاز داریم دانش اولیه‌ای نسبت به بعضی از تعاریف گرافیک کامپیوتری مانند "مش‌های چندضلعی"، "زیرمش"، "ماده" و "بافت" پیدا کنیم. در این بخش، ابتدا به این تعاریف می‌پردازیم. سپس به بررسی مدل اسکلتی پرداخته و پس از آن توضیحی از روش‌ها و تعاریف مربوط به پویانمایی اسکلتی را می‌آوریم.

۱-۷-۲ شبکه‌ی^۱ چندضلعی

در گرافیک کامپیوتری سه‌بعدی و مدل‌سازی جامد، شبکه چندضلعی مجموعه‌ای از رئوس، لبه‌ها و وجوه است که شکل یک جسم چندوجهی را مشخص می‌کند. وجوه معمولاً از مثلث‌ها (شبکه مثلثی)، چهار ضلعی‌ها (چهار گوشه)، یا دیگر چند ضلعی‌های محدب ساده (n ضلعی‌ها) تشکیل شده‌اند. دلیل استفاده از این نوع چند ضلعی‌ها، آسان‌تر بودن به نمایش کشیدن آن‌ها در محیط سه‌بعدی است. البته در حالت کلی، اشیاء ممکن است از چندضلعی‌های مقعر و یا حتی چندضلعی‌های دارای سوراخ نیز تشکیل شده باشند.

اشیاء ایجاد شده توسط مش‌های چند ضلعی باید انواع مختلفی از عناصر، از جمله رئوس، لبه‌ها، وجوه، چندضلعی‌ها و سطوح را در خود ذخیره کنند. در بسیاری از نرم‌افزارهای سه‌بعدی، فقط رئوس، لبه‌ها و یکی از دو مورد وجوه یا چندضلعی‌ها ذخیره می‌شوند. در اکثر سیستم‌های رندر^۲ فقط از وجوه سه‌ضلعی (مثلث‌ها) استفاده می‌شود. بنابراین در این حالت چندضلعی‌های مدل، باید به شکل مثلث باشند. البته سیستم‌های رندر ای وجود دارند که از چهارضلعی‌ها یا چندضلعی‌های با تعداد اضلاع بالاتر نیز پشتیبانی می‌کنند و یا در لحظه این چندضلعی‌ها را به مجموعه‌ای از مثلث‌ها تبدیل می‌کنند که در این صورت باعث می‌شود نیازی به ذخیره‌ی مش به شکل مثلثی نباشد.

در شکل ۵-۲ می‌توان یک دلفین را مشاهده کرد که توسط مش مثلثی، مدل شده است.

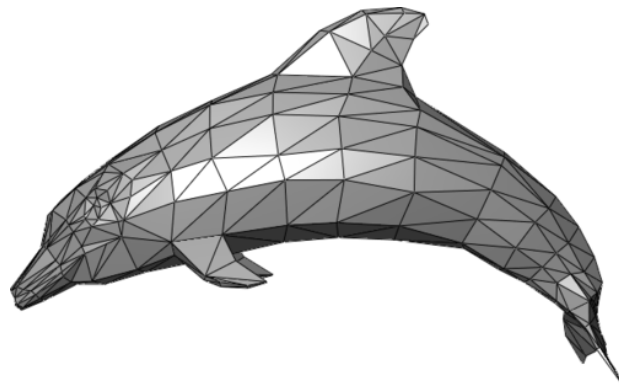
بنابراین چهار قسمت اصلی یک مش چندضلعی، رئوس، لبه‌ها، وجوه و چندضلعی‌ها هستند که در شکل ۶-۲ نیز قابل مشاهده است. توضیح کوتاهی درباره‌ی هر کدام از این مولفه‌ها را در زیر می‌توانیم مشاهده کنیم.

راس

راس‌ها معمولاً یک موقعیت در فضای سه‌بعدی همراه با اطلاعات دیگر مانند رنگ، بردار نرمال و مختصات بافت را شامل می‌شوند. در راس‌های مربوط به مش‌های اسکلتی اطلاعاتی مانند تعداد مفاصلی که بر روی این راس تاثیر

¹Mesh

²renderer



شکل ۲-۵ - نمایش یک دلفین به وسیله‌ی مش مثلثی [۱۶]

می‌گذارند همراه با وزن تاثیر گذاری آن‌ها، می‌تواند اضافه شود.

لبه

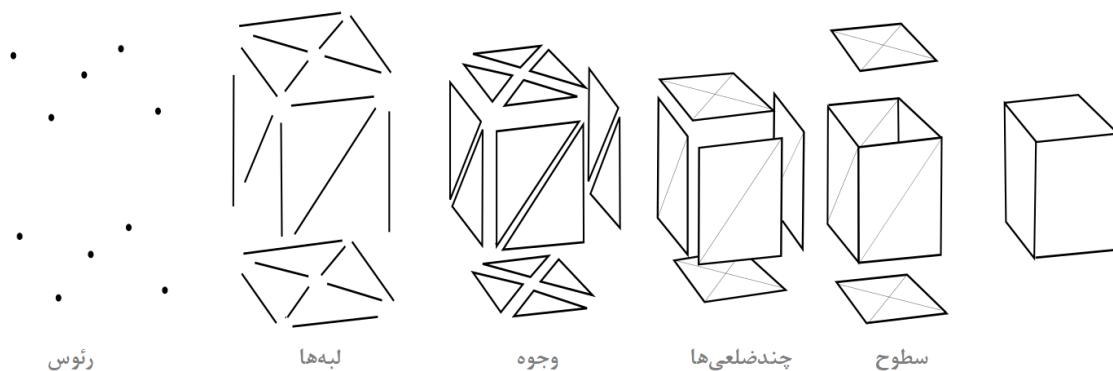
ارتباط بین دو راس را لبه گویند.

وجه

مجموعه‌ای بسته از لبه‌ها را وجه گویند. وجوه می‌توانند از سه لبه (وجه مثلثی) یا از چهار لبه (وجه چهارگوش) تشکیل شوند.

چندضلعی

یک چندضلعی مجموعه‌ای همسطح از وجوه است. در سیستم‌هایی که از وجه‌های چند ضلعی پشتیبانی می‌کنند، وجوه و چندضلعی‌ها یکسان هستند ولی در صورتی که سیستم مورد نظر تنها از سه یا چهار ضلعی‌ها پشتیبانی کند، در این صورت به چند ضلعی‌ها، مجموعه‌ای از وجوه گفته می‌شود.



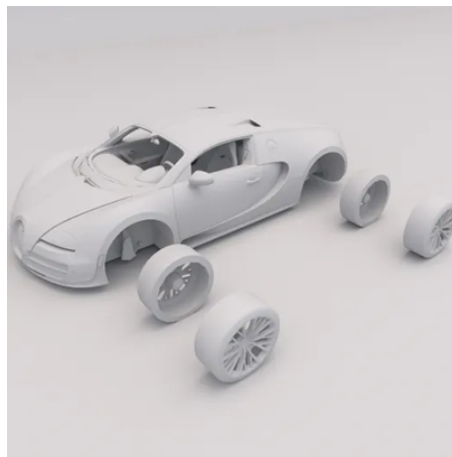
شکل ۲-۶ - عناصر یک مش چندضلعی [۱۶]

۲-۷-۲ مدل

مدل^۱ در واقع هر شی‌ای است که در محیط سه‌بعدی قرار می‌گیرد و به تصویر کشیده می‌شود. هر مدل می‌تواند از چند زیرمش تشکیل شود. به عنوان مثال یک ماشین را در نظر بگیریم. موجودیت ماشین می‌تواند یک مدل باشد که در محیط سه‌بعدی قرار می‌گیرد. مدل ماشین می‌تواند از چند زیرمش مانند چرخ‌ها، لاستیک‌ها و بدنه‌ی ماشین تشکیل شود. دلیل وجود داشتن یک موجودیت کلی به اسم ماشین این است که یک شخصی مانند طراح محیط و یا طراح مرحله نمی‌خواهد هر بار که ماشینی را در محیط قرار دهد، تک تک زیرمش‌ها را به صورت دستی در صحنه وارد کند و در سر جای خودش قرار بدهد بلکه می‌خواهد یک موجودیت، در اینجا ماشین، را در صحنه قرار دهد.

۳-۷-۲ زیرمش^۲

چندضلعی‌های دارای یک نوع ماده^۳ را یک زیرمش گویند. همانطور که اشاره شد، هر مدل از چند زیرمش تشکیل می‌شود. دلیل این تقسیم این است که در هر عملیات به تصویر کشیدن^۴ تنها یک ماده می‌تواند به تصویر کشیده شود. مثلاً در مثال ماشین، قسمت‌های مختلف ماشین از ماده‌های مختلفی تشکیل می‌شوند. به طور مثال چرخ ماشین می‌تواند از جنس آلومینیوم باشد، لاستیک چرخ از جنس پلاستیک باشد و یا حتی قسمت‌های داخلی ماشین مانند صندلی ماشین، از جنس چرم باشد (شکل ۷-۲). بنابراین باید این قسمت‌ها به صورت جدا قرار گیرند تا بتوان هر قسمت را با توجه به ماده‌ی موردنظر آن به تصویر کشاند.



شکل ۷-۲- اجزاء مختلف ماشین [۱۷]

^۱ گاهی به جای استفاده از واژه‌ی مدل، از واژه‌ی مش هم استفاده می‌شود.

^۲ Sub-Mesh

^۳ Material

^۴ Render

۲-۷-۴ ماده^۱

ماده‌ها شامل پارامترهای قابل تنظیمی هستند که با تنظیم آن‌ها، به کارت گرافیک اعلام می‌شود که چگونه باید یک مثلث را به تصویر بکشد. این پارامترها می‌توانند شامل موارد زیر باشند ولی محدود به آن نمی‌شوند

۱. میزان کدورت و شفافیت شیء

۲. میزان براقی شیء

۳. رنگ (بافت) شیء

۴. سایه‌زنی پیکسلی یا راسی^۲

مثالی از چند نوع ماده‌ی متفاوت در شکل ۲-۸ آورده شده است.



شکل ۲-۸ - چند نوع ماده‌ی مختلف [۱۸]

۲-۷-۵ بافت^۳

بافت یک تصویر دوبعدی و یا سه‌بعدی است که می‌تواند در ماده استفاده شود. این تصاویر به عنوان ورودی در برنامه دریافت شده و پس از اینکه یک شناسه به آن‌ها تخصیص داده شد، در کارت گرافیک قرار می‌گیرند. ماده‌ها با استفاده از این شناسه می‌توانند در صورت لزوم به این بافت‌ها دستیابی پیدا کنند.

مثالی از چند نوع بافت متفاوت در شکل ۲-۹ آورده شده است.

^۱Material

^۲Vertex or Pixel shader

^۳Texture



شکل ۲-۹ - چند نوع بافت مختلف [۱۹]

۶-۷-۲ اسکلت

به مجموعه ای از مفاصل که به صورت سلسله مراتبی به یکدیگر متصل می شوند، اسکلت گویند. پس از آنکه هنرمندان مدل شخصیت را طراحی می کنند، در طی یک مرحله که به آن Rigging گویند، ساختار سلسله مراتبی اسکلت را به وجود می آورند. در پویانمایی اسکلتی درواقع این اسکلت است که حرکت می کند و با حرکتش باعث حرکت مدل شخصیت می شود.

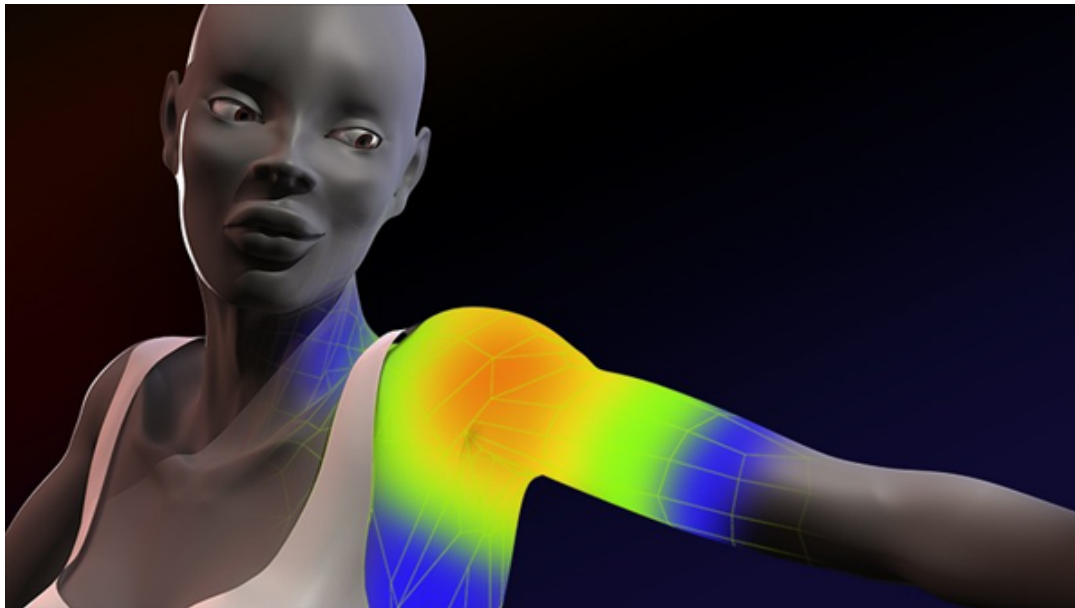
در شکل ۲-۱۰ می توان یک مش یک شخصیت سه بعدی را به همراه اسکلت آن مشاهده کرد.



شکل ۲-۱۰ - در سمت راست مش شخصیت و سمت چپ اسکلت شخصیت قابل مشاهده است. [۲۰]

۷-۷-۲ Skinning

تا اینجا با دو مفهوم مدل و اسکلت آشنایی پیدا کردیم ولی نگفتیم که این دو چگونه به هم مرتبط می‌شوند. به عملیاتی که طی آن مفاصل موجود در اسکلت به مدل متصل می‌شود skinning گویند. (شکل ۱۱-۲) طی این مرحله هر راس موجود در پوسته‌ی مش به یک یا چند مفصل متصل می‌شود. برای اینکه چگونه رئوس مش، این مفاصل را دنبال کنند الگوریتم‌های مختلفی مطرح شده‌است که در فصل پیاده‌سازی به یکی آن‌ها اشاره خواهد شد.



شکل ۱۱-۲ - عکس نشان‌دهنده‌ی میزان تعلق رئوس اطراف مفصل شانه است. هرچه راس به مفصل نزدیک تر باشد، تعلق بیشتری به آن دارد (رنگ قرمز تری می‌گیرند) [۲۱].

۸-۷-۲ ژست شخصیت

ژست یک شخصیت نشان‌دهنده‌ی نحوه‌ی قرارگیری مفصل‌ها در اسکلت است. ژست‌های مختلف با دروان، حرکت یا تغییر اندازه‌ی مفاصل درون اسکلت به وجود می‌آیند. همانگونه که اشاره شد، اسکلت یک مدل در مرحله‌ی Rigging به وجود می‌آید و در همین مرحله با استفاده از Skinning به مدل متصل می‌شود. زمانی که این عمل صورت می‌گیرد مدل در یک ژست به خصوص قرار دارد که به آن ژست حالت اتصال^۱ یا ژست مرجع^۲ گویند. به صورت کلی شخصیت در این حالت به صورتی ایستاده‌است که پاهایش کمی از هم باز است و بازوهایش به شکل حرف T کشیده است. به همین جهت گاهی به ژست حالت اتصال، ژست T^۳ هم گفته می‌شود. دلیل انتخاب این ژست خاص دور نگه‌داشتن اندام‌ها از بدن است که باعث می‌شود فرایند اتصال رئوس به مفاصل آسان‌تر شود.

^۱Bind Pose

^۲Reference Pose

^۳T Pose

همانگونه که اشاره شد مفاصل به صورت سلسله مراتبی به یکدیگر متصل هستند. یعنی نحوه‌ی قرارگیری آن‌ها متناسب با نحوه‌ی قرارگیری والدشان است. این کار باعث می‌شود که مفاصل به صورت طبیعی حرکت کنند. یعنی در صورتی که والد حرکت کند، به واسطه‌ی آن فرزند نیز حرکت می‌کند. زمانی که ژست شخصیت در این حالت والد، فرزند را قرار دارد به آن ژست محلی^۱ گفته می‌شود. حالت دیگری نیز وجود دارد که موقعیت هر مفصل نسبت به فضای مختصاتی مدل در نظر گرفته می‌شود. به ژست شخصیت در این حالت ژست جهانی^۲ گفته می‌شود. [۸]

۹-۷-۲ کلیپ‌های پویانمایی

در یک فیلم پویانمایی شده، تمام بخش‌های یک صحنه قبل از ساخت هر کلیپ پویانمایی به دقت برنامه‌ریزی می‌شود. این شامل حرکات هر شخصیت، لوازم موجود در صحنه و حتی حرکات دوربین نیز می‌شود. این بدان معنی است که کل صحنه را می‌توان به عنوان یک دنباله طولانی و پیوسته از فریم‌ها، متحرک ساخت. در این حالت در صورتی که شخصیت‌ها خارج از دوربین باشند لازم نیست که متحرک شوند.

کلیپ‌های پویانمایی مربوط به بازی‌های کامپیوتری، متفاوت از این هستند. یک بازی، یک تجربه‌ی تعاملی است بنابراین نمی‌توان از قبل چگونه حرکت کردن شخصیت‌ها و رفتار آن‌ها را پیش‌بینی کرد. حتی تصمیمات شخصیت‌های غیربازیکن کامپیوتری نیز می‌توانند تابعی از اقدامات غیر قابل پیش‌بینی بازیکن انسانی باشد. به این ترتیب، کلیپ‌های پویانمایی مربوط به بازی تقریباً هیچ‌گاه از مجموعه‌ای از فریم‌های طولانی و به هم پیوسته تشکیل نمی‌شوند. در عوض، حرکت شخصیت بازی باید به تعداد زیادی حرکات ریز تقسیم شود. منظور از کلیپ‌های پویانمایی این حرکات کوتاه و یکتا است.

بنابراین هر کلیپ به صورتی طراحی شده است که یک عمل کاملاً مشخص را انجام دهد. برخی از این کلیپ‌ها به گونه‌ای طراحی شده‌اند که بتوان آن را به صورت حلقه شونده تکرار کرد. به عنوان مثال چرخه‌ی راه‌رفتن یا دویدن می‌توانند از این نوع کلیپ‌ها باشند و حرکاتی مانند پریدن یا دست تکان دادن از نوعی هستند که تنها یک‌بار پخش می‌شوند.

بنابراین به طور کلی حرکات هر شخصیت بازی معمولاً به هزاران کلیپ تقسیم می‌شود [۸].

۱۰-۷-۲ ترکیب کلیپ‌های پویانمایی

اصطلاح ترکیب کلیپ‌ها به هر تکنیکی اطلاق می‌شود که در آن بیش از یک کلیپ پویانمایی در ژست نهایی کاراکتر سهیم می‌شود. به صورت دقیق‌تر در این عمل دو یا چند ژست برای ایجاد یک ژست خروجی برای اسکلت

^۱Local Pose

^۲Global Pose

شخصیت، با یکدیگر ترکیب می‌شوند. همانطور که در بخش قبل گفته شد، کلیپ‌های پویانمایی، کلیپ‌های کوتاه و یکتایی هستند. با استفاده از روش ترکیب می‌توان مجموعه‌ای از کلیپ‌های پویانمایی را با یکدیگر ترکیب کرد تا مجموعه‌ی جدیدی از کلیپ‌های پویانمایی را بدون نیاز به ایجاد دستی و از پایه‌ی آن‌ها تولید کنیم.

به عنوان مثال، با ترکیب یک کلیپ راه رفتن آسیب دیده با راه رفتن بدون آسیب دیدگی، می‌توانیم سطوح مختلفی از آسیب دیدگی در هنگام راه رفتن را به وجود آوریم. از ترکیب می‌توان برای درونیابی بین حالات مختلف چهره، حالت‌های مختلف بدن و حالت‌های مختلف حرکتی استفاده کرد. علاوه بر این می‌توان از آن برای یافتن یا حالت میانی بین دو حالت شناخته شده در زمان‌های مختلف نیز استفاده کرد. این کار زمانی استفاده می‌شود که بخواهیم ژست یک شخصیت را در نقطه‌ای از زمان پیدا کنیم که دقیقاً با یکی از فریم‌های نمونه موجود در داده‌های کلیپ پویانمایی مطابقت ندارد. همچنین می‌توانیم از ترکیب موقتی پویانمایی برای انتقال هموار از یک کلیپ به کلیپ دیگر، با ترکیب تدریجی کلیپ پویانمایی مبدا به مقصد در مدت زمان کوتاهی استفاده کنیم.

فصل سوم

سیستم گراف پویانمایی در موتور بازی سازی آنریل

موتور آنریل مجموعه کاملی از ابزارهای تولید محتوا برای توسعه‌ی بازی، مصورسازی معماری و خودرو، ایجاد محتوا برای فیلم و تلویزیون، پخش رویدادهای زنده، آموزش، شبیه‌سازی و سایر برنامه‌های بلادرنگ است. این موتور برای اولین بار برای توسعه‌ی بازی "غیرواقعی" در سال ۱۹۹۸ توسعه پیدا کرد. پس از آن نسخه‌های متعددی از این موتور منتشر شده است [۹].

موتور آنریل مانند تمامی موتورهای بازی‌سازی دارای مولفه‌های فراوانی است که برای تولید بازی به کار می‌رود. مولفه‌های پویانمایی، هوش مصنوعی، رندر، رابط کاربری تنها تعداد اندکی از مولفه‌هایی است که می‌توان در آنریل استفاده کرد.

آنریل طیف گسترده‌ای از ابزارهای قدرتمند را برای مدیریت شخصیت‌ها، ایجاد محتوای سینمایی و پویانمایی را ارائه می‌دهد. با استفاده از سیستم پویانمایی مش اسکلتی، کاربران می‌توانند شخصیت‌ها، اسکلت‌ها و کلیپ‌های پویانمایی وارد شده خود را مدیریت کنند. سپس این محتوا می‌تواند برای ایجاد گیم‌پلی تعاملی پویا شده با استفاده از ویژگی‌های مختلف مانند فضاها، ترکیبی^۱، طرح‌های پویانمایی^۲ و ماشین‌های حالت^۳ استفاده شود. سکانس‌های سینمایی را

^۱Blend Spaces

^۲Animation Blueprint

^۳State Machines

می‌توان با استفاده از ابزار Sequencer ایجاد کرد. با استفاده از این ابزار می‌توان دوربین‌ها و شخصیت‌ها را متحرک ساخت. پویانمایی شخصیت‌ها را می‌توان با استفاده از Control Rig که ابزار داخلی موتور آنریل است، انجام داد. با استفاده از این ابزار می‌توان ریگ‌های مناسبی ساخت تا درون Sequencer شخصیت را متحرک ساخت [۲۲]. همانطور که مشخص است، سیستم پویانمایی موجود در موتور آنریل بسیار گسترده است. در این پروژه قسمت طرح پویانمایی آنریل که به گراف پویانمایی نیز شناخته می‌شود بررسی می‌شود.

برای اینکه بتوانیم در مورد سیستم گراف پویانمایی آنریل توضیح دهیم، ابتدا لازم است توضیحاتی را درباره‌ی نحوه‌ی معماری این انجین بیاوریم. بنابراین در این بخش ابتدا توضیح کوتاهی درباره‌ی معماری آنریل با محوریت نحوه‌ی رابطه‌ی اشیا با یکدیگر داده و پس از آن به بررسی ویژگی‌های سیستم گراف پویانمایی می‌پردازیم.

۳-۱ بازیگران، پیاده‌ها و شخصیت‌ها

اشیا در آنریل به سه کلاس کلی بازیگران^۱، پیاده‌ها^۲ و شخصیت‌ها^۳ دسته‌بندی می‌شوند. بازیگران کلاس پایه‌ی تمامی اشیا ای هستند که به صورت فیزیکی می‌توانند در محیط سه‌بعدی قرار گیرند. پیاده‌ها کلاسی مشتق شده از بازیگران هستند که بازیکنان می‌توانند کنترل آن‌ها را بدست گیرند و در محیط حرکت کنند. در نهایت شخصیت‌ها پیاده‌هایی هستند که دارای مش اسکلتونی، توانایی شناسایی برخورد و منطق حرکتی هستند. آنها مسئول تمام تعاملات فیزیکی بین بازیکن یا هوش مصنوعی، با جهان هستند و همچنین مدل‌های اولیه شبکه و دریافت ورودی را پیاده‌سازی می‌کنند. اگر بخواهیم شخصیت درون بازی از پویانمایی اسکلتونی استفاده کند، باید از این کلاس بهره ببریم.

۳-۲ اجزاء

اجزاء^۴ مجموعه‌ای از توابع و ویژگی‌ها است که می‌تواند به یک بازیگر اضافه شود. بنابراین بازیگران می‌توانند حاوی مجموعه‌ای از ActorComponents باشند که این اجزاء می‌توانند برای موارد مختلفی از جمله کنترل نحوه‌ی حرکت بازیگران، نحوه‌ی رندر شدن و غیره استفاده شوند.

زمانی که یک مولفه به یک بازیگر اضافه می‌شود، آن بازیگر می‌تواند عملکردهای موجود در آن مولفه را استفاده کند. به عنوان مثال یک مولفه نور نقطه‌ای باعث می‌شود که بازیگر مانند یک نور نقطه‌ای، نور ساطع کند. یا یک مولفه صوتی به بازیگر این توانایی پخش صدا را می‌دهد.

¹ Actors

² Pawns

³ Characters

⁴ Components

مولفه‌ها حتما باید به یک بازیگر متصل شوند و به خودی خود نمی‌توانند وجود داشته باشند. درواقع وقتی ما مولفه‌های مختلف را به بازیگر خود متصل می‌کنیم، در حال قرار دادن قطعه‌ها و تکه‌هایی هستیم که مجموع آن‌ها یک بازیگر را به عنوان یک موجودیت واحد که در محیط سه‌بعدی قرار می‌گیرد، تعریف می‌کنند. به عنوان مثال چرخ‌های یک ماشین، فرمان ماشین، چراغ‌ها و غیره همه به عنوان مولفه‌های ماشین در نظر گرفته می‌شوند در حالی که خود آن ماشین، بازیگر است.

۳-۳ شخصیت‌ها

هر شخصیت در آنریل از سه مولفه اصلی تشکیل شده است.

Skeletal Mesh Component -

Character Movement Component -

Capsule Component -

مولفه Skeletal mesh Component شامل طرح پویانمایی شخصیت است. طرح پویانمایی، سیستم پویانمایی شخصیت است که جلوتر آن را توضیح می‌دهیم.

مولفه Character Movement Component همانطور که از اسمش مشخص است برای منطق حرکت در حالت‌های مختلف از جمله راه‌رفتن، افتادن و غیره استفاده می‌شود. این مولفه شامل تنظیمات و عملکردهای مربوطه برای کنترل حرکت است.

و در نهایت مولفه Capsule Component وظیفه‌ی تشخیص برخورد در هنگام حرکت را دارد.

۳-۴ مولفه‌ی مش اسکلتی

این مولفه، مولفه‌ای است که به شخصیت امکان پویا شدن را می‌دهد. این کلاس برای ساختن یک نمونه از کلاس SkeletalMesh است که بر روی آن کلیپ‌های پویانمایی اجرا می‌شوند. اینکه چه کلیپ پویانمایی بر روی آن اجرا شود از طریق کلاس AnimInstance که همان طرح پویانمایی^۱ است، انتخاب می‌شود.

همانطور که در فصل گذشته اشاره شد، مش اسکلتی شامل یک هندسه‌ی چندضلعی است که به یک اسکلت که در واقع سلسله‌مراتبی از مفاصل است، متصل است و این اسکلت می‌تواند به منظور تغییر شکل آن هندسه‌ی چندضلعی یا مش، متحرک شود.

مش‌های اسکلتی از دو قسمت ساخته شده‌اند. مجموعه‌ای از چندضلعی‌ها که به منظور تشکیل سطح مش با یکدیگر

^۱ Animation Blueprint

ترکیب می‌شوند و یک اسکلت سلسله‌مراتبی که می‌تواند برای متحرک‌سازی چندضلعی‌ها استفاده شود. مدل‌های سه‌بعدی، اسکلت و کلیپ‌های پویانمایی در یک برنامه مدل‌سازی و ایجاد پویانمایی مانند Maya، 3DSMax و ابزارهای مدل‌سازی دیگر ایجاد می‌شوند.

در آنریل کلاس SkeletalMesh وظیفه‌ی نگهداری این مش اسکلتی را دارد. همانگونه که گفتیم، نیاز داریم تا یک سیستمی داشته باشیم که بتواند کلیپ‌های پویانمایی را بر روی این مش اسکلتی اجرا کند. به زبانی دیگر، این مش اسکلی را پویا و متحرک سازد. در آنریل کلاس AnimInstance وظیفه‌ی این عمل را دارد.

۳-۵ طرح پویانمایی

طرح پویانمایی یک طرح تخصصی است که پویانمایی یک مش اسکلتی را کنترل می‌کند. با ویرایش گراف‌های موجود در این طرح، میتوان کارهای مختلفی را روی پویانمایی شخصیت انجام داد. به عنوان مثال می‌توان کلیپ‌های مختلف را با یکدیگر ترکیب کرد، مستقیماً مفاصل درون اسکلت را کنترل کرد و یا هر تنظیمات منطقی‌ای که باعث تعریف ژست نهایی شخصیت در فریم فعلی می‌شود را انجام داد.

دو جزء اصلی در طرح پویانمایی وجود دارد که با هم کار می‌کنند تا ژست نهایی شخصیت را برای هر فریم ایجاد کنند. این دو مولفه، گراف رویداد و گراف پویانمایی نام دارند.

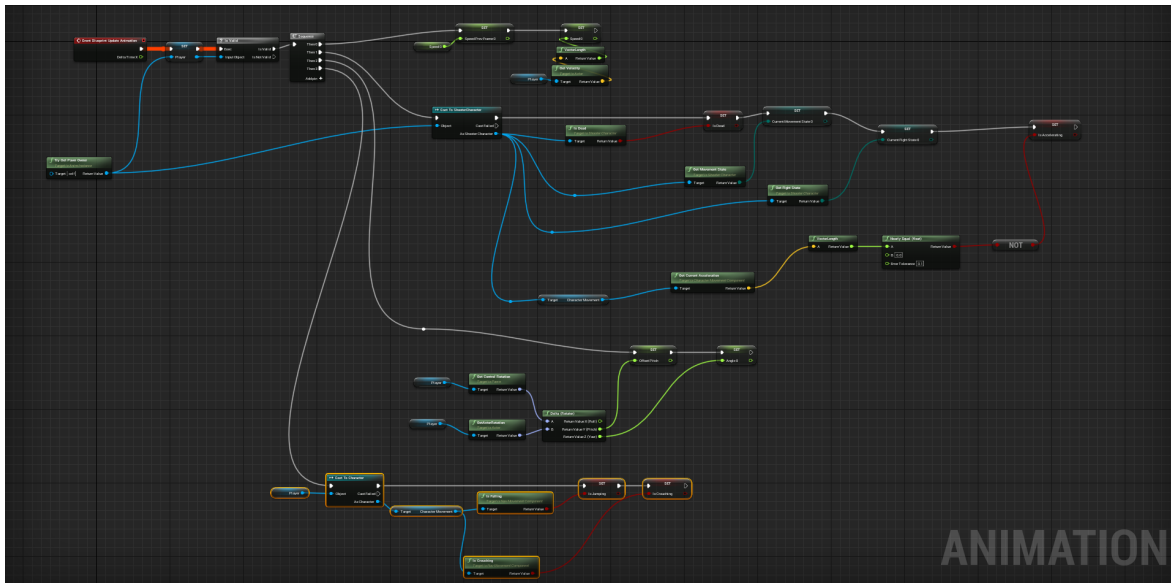
به صورت کلی گراف رویداد مقادیری را که در گراف پویانمایی استفاده می‌شوند را به‌روزرسانی می‌کند تا در ماشین‌های حالت، فضاها و ترکیب و بقیه‌ی گره‌هایی که در گراف پویانمایی استفاده می‌شوند، به کار روند.

۳-۶ گراف رویداد

درون هر طرح پویانمایی، یک گراف رویداد وجود دارد. این گراف برای دریافت مقادیر منطقی از بخش گیمپلی و منطق بازی به کار می‌رود. به عنوان مثال اینکه شخصیت می‌خواهد به چه سمتی حرکت کند، یا اینکه چه سرعتی دارد را از طریق این گراف در متغیرهایی که تعریف می‌کنیم، ذخیره می‌کنیم [۲۳]. نمونه‌ای از گراف رویداد را می‌توان در شکل ۳-۱ مشاهده کرد.

۳-۷ گراف پویانمایی

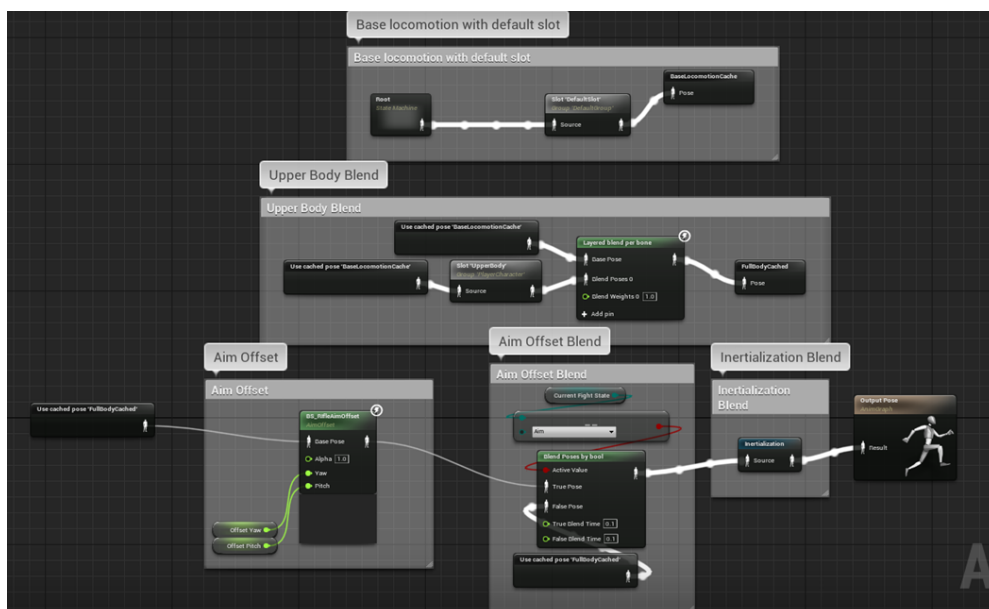
گراف پویانمایی برای ارزیابی ژست نهایی مش اسکلتی در فریم فعلی استفاده می‌شود. به صورت کلی هر طرح پویانمایی دارای یک گراف پویانمایی است که این گراف شامل گره‌های مختلفی است که هر کدام از این گره‌ها



شکل ۳-۱ - نمونه‌ای از یک گراف رویداد

استفاده‌های متفاوتی دارند. به عنوان مثال می‌توان از این گره‌ها برای نمونه‌برداری^۱ از دنباله‌های کلیپ‌های پویانمایی، انجام ترکیب‌های بین کلیپ‌ها یا کنترل تبدیل‌های مربوط به مفاصل استفاده کرد. سرانجام ژست نهایی بدست آمده روی مش اسکلتی در پایان هر فریم اعمال می‌شود.

نمونه‌ای از گراف پویانمایی را می‌توان در شکل ۳-۲ مشاهده کرد.



شکل ۳-۲ - نمونه‌ای از یک گراف پویانمایی

^۱ Sampling

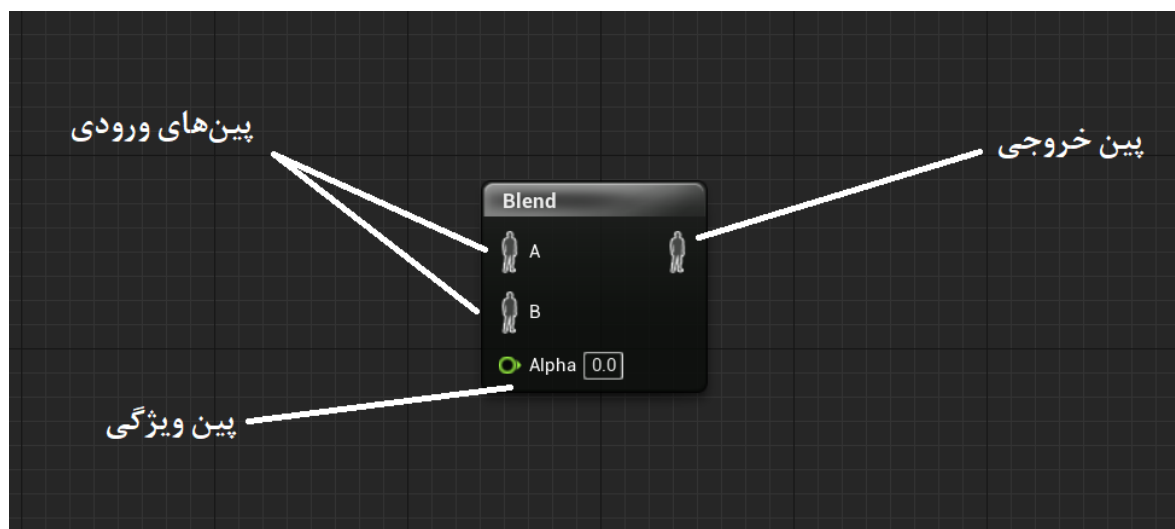
۸-۳ گره‌های گراف پویانمایی

گراف پویانمایی با ارزیابی گره‌های موجود در گراف، عمل می‌کند. بعضی از گره‌های موجود در گراف، عملیات‌های خاصی را بر روی یک یا چند ژست ورودی انجام می‌دهند، در حالی که برخی دیگر برای دسترسی یا نمونه‌برداری از انواع دیگری از دارایی‌ها مانند فضاها یا ترکیب^۱، مونتاژهای پویانمایی^۲ و دنباله‌های پویانمایی^۳ استفاده می‌شوند. ماشین‌های حالت نیز که حاوی شبکه‌ی نموداری خودشان هستند، می‌توانند به صورت تنهایی یا با ترکیب با یکدیگر در گراف پویانمایی استفاده شوند.

در این بخش ابتدا با نحوه‌ی جریان اجرا در گراف پویانمایی آشنا می‌شویم، سپس ساختار کلی گره‌های موجود را بررسی کرده و در نهایت به بررسی انواع گره‌های موجود در گراف پویانمایی می‌پردازیم.

۹-۳ ساختار گره‌های گراف پویانمایی

گره‌ها می‌توانند شامل چند پین ورودی که در واقع ژست‌های ورود هستند، باشند. به صورت کلی پین‌ها شامل یک خروجی هستند که این خروجی نشان‌دهنده‌ی ژست شخصیت پس از انجام عملیات‌های مربوط به آن پین است. همچنین می‌توانند شامل پین‌های ویژگی باشند. مقادیر این پین‌های ویژگی از متغیرهایی که در گراف رویداد تعریف و مقداردهی شده‌اند، می‌توانند بدست آیند. در شکل ۳-۳ این ساختار آورده شده‌است.



شکل ۳-۳ - ساختار کلی گره‌های موجود در گراف پویانمایی

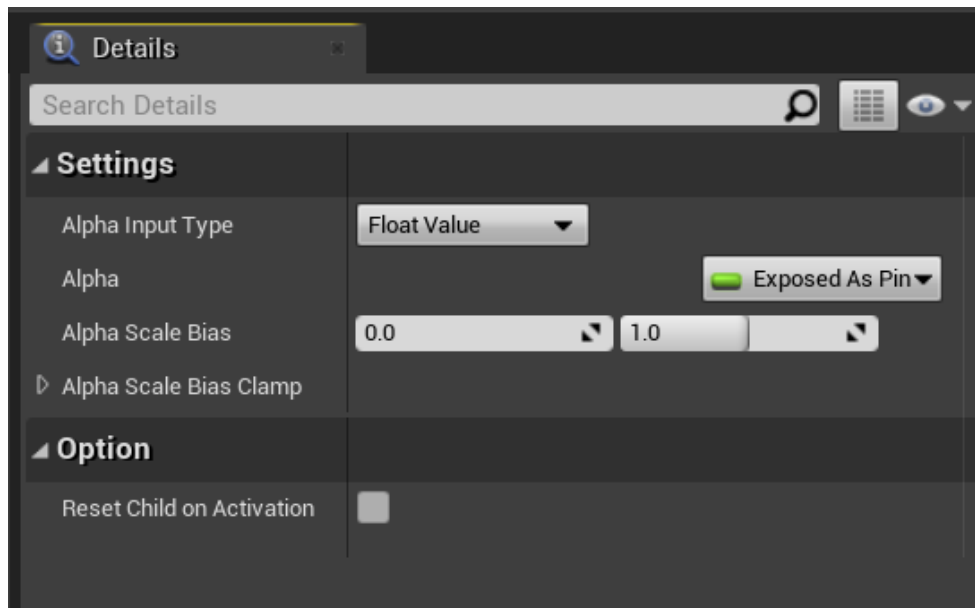
قابل ذکر است با انتخاب هر گره می‌توان به پتل جزئیات آن هم دسترسی پیدا کرد که به وسیله‌ی آن می‌توان

^۱Blend Spaces

^۲Animation Montages

^۳Animation Sequence

تنظیمات لازم را بر روی آن گره انجام داد. (شکل ۴-۳)



شکل ۴-۳ - پنل تنظیمات گرهی ترکیب

۳-۱۰ جریان اجرا در گراف پویانمایی

تمامی گراف‌ها دارای یک جریان اجرا هستند که به صورت پیوندهای ضربانی میان ورودی و خروجی بین‌ها قابل مشاهده هستند. این جریان‌ها در واقع نحوه حرکت داده را در گراف ترسیم می‌کنند. (شکل ۳-۵) در گراف پویانمایی، این جریان نشان‌دهنده‌ی ژست‌هایی است که از یک گره به گره‌ی دیگر منتقل می‌شود. در برخی از گره‌ها مانند گرهی ترکیب، ورودی‌های متعددی وجود دارند و به صورت درونی با مقادیری که در متغیرها داریم تصمیم می‌گیرند که کدام یک از ورودی‌ها در حال حاضر بخشی از جریان اجرا است [۲۴].

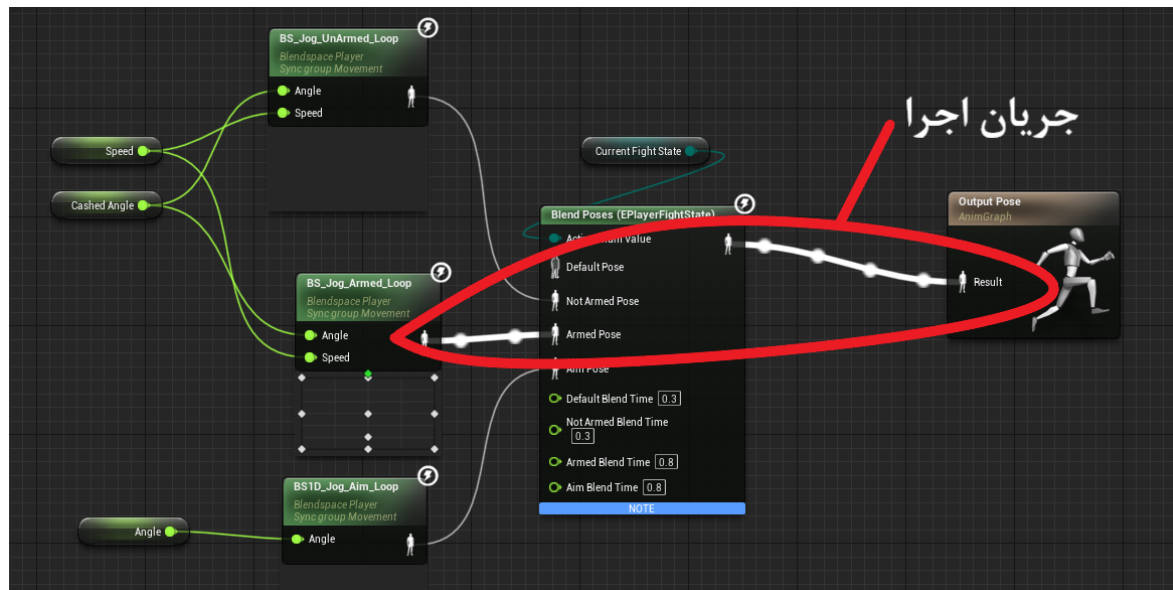
۳-۱۱ دنباله‌ی پویانمایی

کلیپ‌های پویانمایی در آنریل به اسم دنباله‌ی پویانمایی شناخته می‌شوند. دنباله‌ی پویانمایی یک دارایی پویانمایی و حاوی داده‌های پویانمایی است که می‌تواند روی یک مش اسکلتی پخش شود تا شخصیت مربوط به آن اسکلت را متحرک سازد. یک دنباله‌ی پویانمایی شامل فریم‌های کلیدی هستند که این فریم‌های کلیدی بیانگر موقعیت^۱، دوران^۲، مقیاس^۳ اسکلت مش در نقطه‌ی خاصی از زمان است. بنابراین کلیپ‌های پویانمایی یکی از گره‌های مهم در گراف انیمیشن حساب می‌آیند. در شکل ۳-۶ یک دنباله‌ای از فریم‌های کلیدی کلیپ پویانمایی دویدن به همراه گرهی مربوط

^۱Position

^۲Rotation

^۳Scale



شکل ۳-۵ - نمونه‌ای از جریان اجرا

به آن آورده شده‌است.

قابل ذکر است این گره مانند بقیه‌ی گره‌ها دارای تنظیماتی است که در پنل تنظیمات قابل مشاهده هستند. به عنوان مثال می‌توان سرعت حرکت کلیپ را در این تنظیمات مشخص کرد یا اینکه کلیپ به صورت حلقه‌وار تکرار شود یا خیر.



شکل ۳-۶ - نمونه‌ای از گره‌ی دنباله‌ی پویانمایی

۱۲-۳ فضای ترکیب

فضای ترکیب یک دارایی به خصوص است که امکان ترکیب کلیپ‌های پویانمایی بر اساس مقدار مختلف ورودی را دارد.

به عنوان مثال فرض کنیم که چند کلیپ داریم که وضعیت حرکت شخصیت را مشخص می‌کنند. این کلیپ‌ها می‌توانند به حالت ایستاده، حرکت با سرعت کم، حرکت با سرعت متوسط و حرکت با سرعت سریع تقسیم شوند.

علاوه بر این‌ها می‌توان این پویانمایی‌ها را برای جهت‌های مختلف داشت. در این صورت می‌توان یک فضای ترکیب داشت که بر اساس دو ورودی، سرعت و جهت عمل می‌کند. و می‌توان هر کدام از این کلیپ‌ها را در جای مشخص خودش (با توجه به سرعت و جهت) قرار داد. (شکل ۳-۷)

در جلوتر اشاره می‌شود که می‌توان کلیپ‌ها را با استفاده از گره‌ی ترکیب نیز، با یکدیگر ترکیب نمود. فضای ترکیب، ابزاری برای انجام ترکیب‌های پیچیده‌تر بین کلیپ‌های پویانمایی متعدد بر اساس مقادیر متفاوت است. هدف این گره، کاهش نیاز به ایجاد گره‌های منفرد در هنگامی که می‌خواهیم ترکیب بر اساس ویژگی‌ها یا شرایط خاصی صورت گیرد، است.



شکل ۳-۷- در این فضای حالت، سرعت بین ۰ تا ۵۰۰ متغیر است و زاویه بین -۱۲۰ تا ۱۲۰ متغیر است. (در شکل به صورت کمی نوشته شده است) در کل از ۱۰ کلیپ پویانمایی شده (نقاط سفیدرنگ در تصویر) ولی با استفاده از ترکیب بین این کلیپ‌ها می‌توان تمامی مقادیر بین کلیپ‌ها را نیز بدست آورد.

۱۳-۳ گره‌های ترکیب

گره‌های ترکیب برای ترکیب چندین کلیپ پویانمایی با یکدیگر استفاده می‌شوند. این گره‌ها تنها مختص گراف پویانمایی هستند و در گراف‌های دیگر مانند گراف رویداد نمی‌توان از آن‌ها استفاده کرد. به صورت کلی هر کدام از این نوع گره‌ها دارای چند پین ورودی و یک آلفا یا وزن است که برای محاسبه‌ی وزن هر کدام از ژست‌های ورودی در ژست خروجی به کار می‌رود. بعضی از این گره‌ها می‌توانند پیچیده‌تر نیز باشند و نیاز به داده‌های بیشتری به عنوان ورودی باشند.

در ادامه به بررسی چند نمونه از گره‌های ترکیب می‌پردازیم.

۱-۱۳-۳ گره‌ی ترکیب استاندارد

این گره برای ترکیب کردن دو ژست ورودی با گرفتن یک آلفا عمل می‌کند. اگر ژست‌های ورودی را A و B و خروجی نهایی را Output در نظر بگیریم، (شکل ۳-۸) خروجی به صورت زیر محاسبه می‌شود.

$$Output = A * (1 - alpha) + B * alpha \quad (۱-۳)$$



شکل ۳-۸- گرهی ترکیب

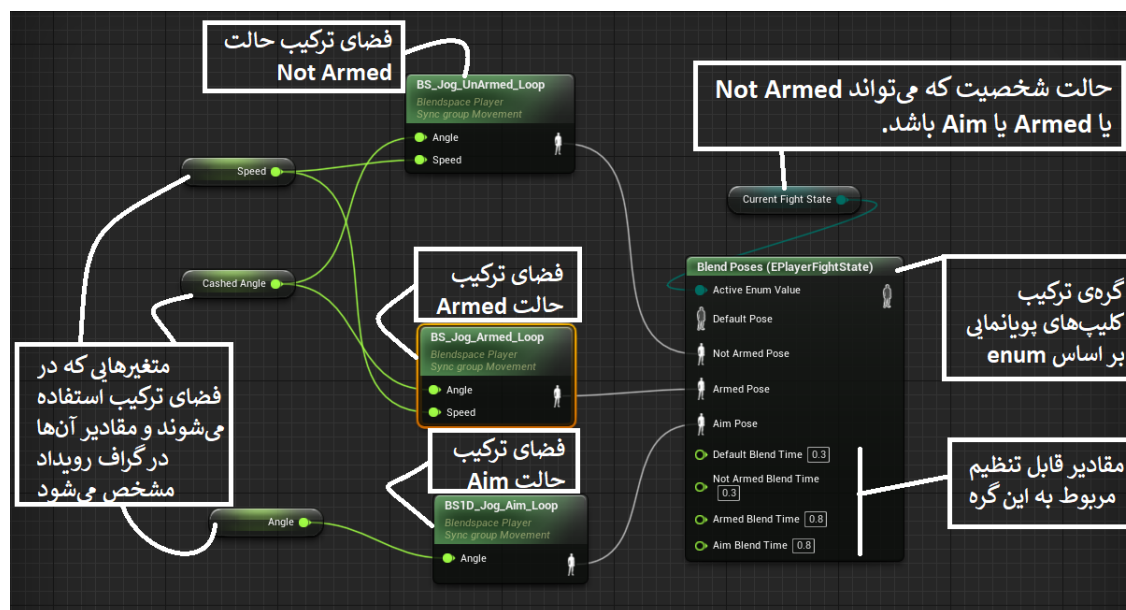
۳-۱۳-۲ گرهی ترکیب بر اساس یک مقدار

این نوع از گره‌ها برای ترکیب کلیپ‌های پویانمایی بر اساس یک مقدار استفاده می‌شوند، که این مقدار می‌تواند عدد صحیح، مقدار بولین و یا مقداری از نوع داده‌ی Enum باشد [۲۵].

به عنوان مثال فرض کنیم شخصیت درون بازی می‌تواند در وضعیت‌های مختلفی از نظر حالت مبارزه با اسلحه قرار گیرد. این حالت‌ها می‌توانند حالت بدون اسلحه، حالت با اسلحه و حالت گرفتن نشانه با اسلحه باشد. می‌توان این حالت‌ها را با Enum نشان داد. اگر برای هر کدام از این حالات یک کلیپ پویانمایی داشته باشیم و بخواهیم با توجه به حالت فعلی شخصیت یکی از این کلیپ‌ها را روی شخصیت پخش کنیم، می‌توانیم از این گره استفاده کنیم. این مثال در شکل ۳-۹ آمده است.

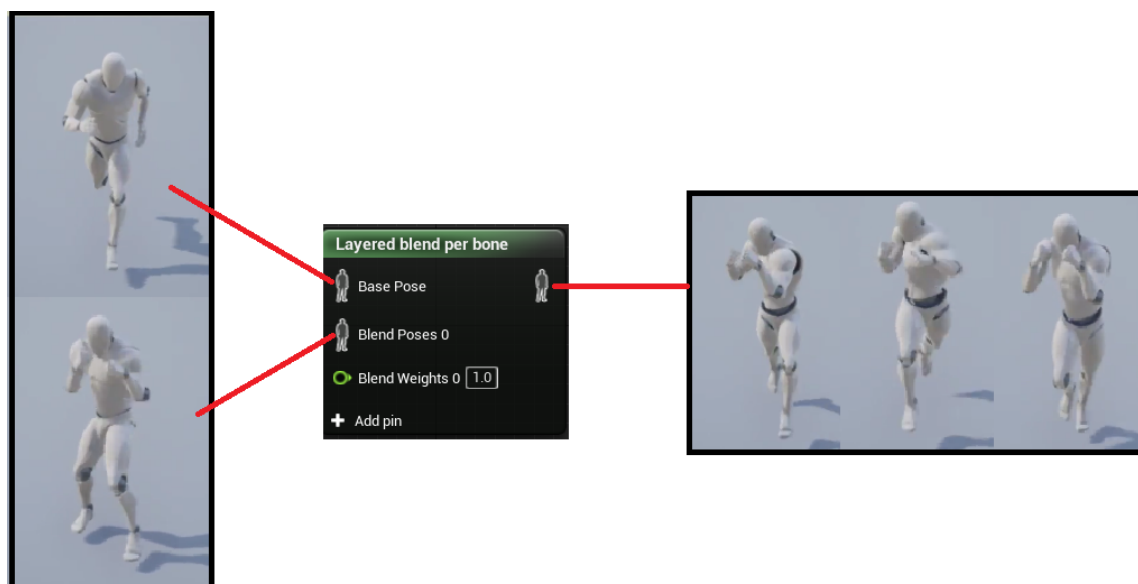
۳-۱۳-۳ گرهی ترکیب لایه‌ای برای هر مفصل

با استفاده از این گره می‌توانیم کلیپ پویانمایی را بر روی مفاصل محدودی از اسکلت اجرا کنیم [۲۵]. به عنوان مثال فرض کنید یک کلیپ پویانمایی مربوط به مشت زدن و کلیپ دیگری مربوط به حرکت شخصیت داریم. اگر بخواهیم از این دو کلیپ استفاده کنیم تا شخصیت در حال حرکت بتواند مشت هم بزند، می‌توان از این گره استفاده



شکل ۳-۹ - مثال استفاده از گرهی ترکیب

کرد. با استفاده از این گره، می‌توان کلیپ مربوط به مشت زدن را تنها بر روی قسمت کمر به بالای شخصیت و کلیپ حرکت را بر روی کمر به پایین شخصیت اجرا کنیم. این مثال در شکل ۳-۱۰ آمده است.



شکل ۳-۱۰ - اجرای کلیپ‌های پویانمایی بر روی مفاصل متفاوت

۱۴-۳ گره‌های کنترل اسکلت

این گره‌ها امکان دستکاری مستقیم مفاصل موجود در اسکلت را فراهم می‌کنند. مجموعه‌ی این گره‌ها شامل حل‌کننده‌های مختلف هستند که می‌توان به حل‌کننده‌ی IK به عنوان مثال اشاره کرد. علاوه بر این بعضی از گره‌های مربوط به این قسمت، می‌توانند برای اعمال فیزیک بر روی مفاصل استفاده شوند [۲۶].

۱۵-۳ گره‌های تبدیل فضا

در موتور آنریل ژست‌ها می‌توانند در فضای محلی یا فضای مولفه قرار گیرند. در فضای محلی مفاصل نسبت به والد خود قرار می‌گیرند، در صورتی که در فضای مولفه، مفاصل نسبت به مولفه‌ی مش اسکلتی^۱ قرار می‌گیرند. اکثر گره‌ها با ژست‌ها در هنگامی که در فضای محلی قرار دارند، کار می‌کنند. اما بعضی از گره‌های ترکیب و تمامی گره‌های کنترل اسکلت با ژست در فضای مولفه کار می‌کنند. بنابراین لازم از در مواقع لازم با استفاده از این گره، ژست اسکلت را از یک فضا به فضای دیگر منتقل کنیم [۲۷]. در شکل ۱۱-۳ این دو گره‌ی تبدیل آورده شده‌است.

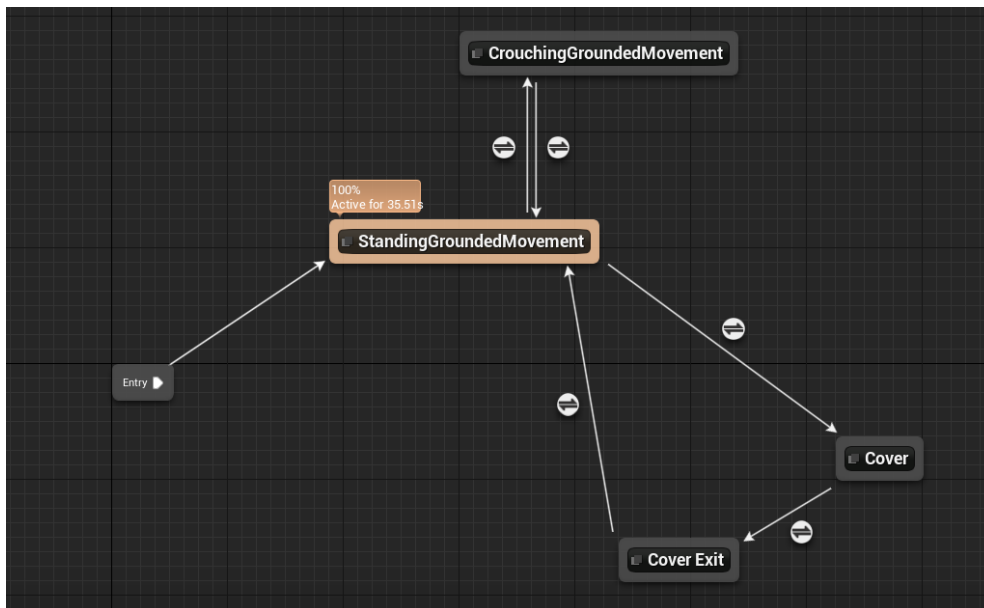


شکل ۱۱-۳ - گره‌های تبدیل حالت

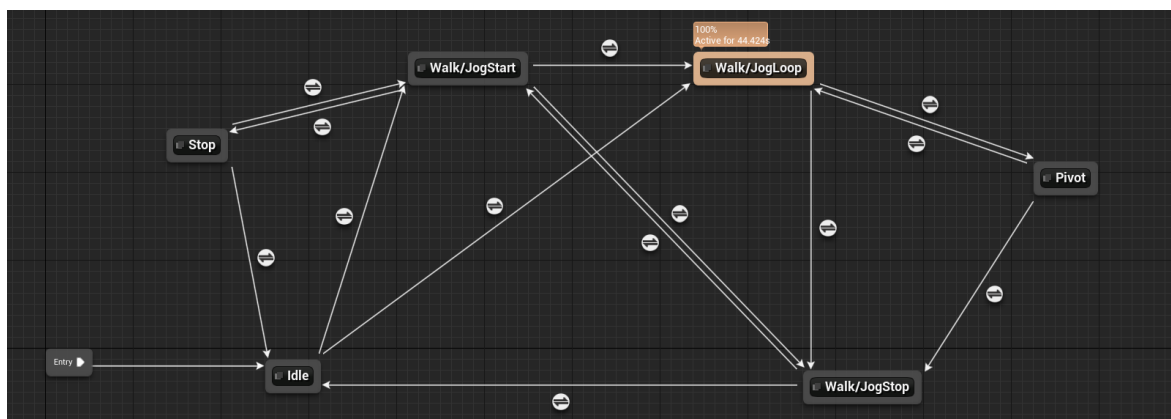
۱۶-۳ گره‌ی ماشین حالت

ماشین‌های حالت، یک راه گرافیکی برای شکستن پویانمایی شخصیت‌ها به یک سری حالت را ارائه می‌دهند. در آنریل می‌توان ماشین‌های حالت پیچیده و تو در تو بر اساس نیاز کاربران به وجود آورد. به عنوان مثال شکل ۱۲-۳ یک ماشین حالت برای حرکت شخصیت را نشان می‌دهد. هر کدام از این حالت‌ها خود دوباره یک ماشین حالت هستند و در شکل ۱۳-۳ ماشین حالت مربوط به حالت StandingGroundedMovement را می‌توان مشاهده کرد.

^۱ SkeletalMeshComponent



شکل ۳-۱۲ - ماشین حالت برای حرکت شخصیت



شکل ۳-۱۳ - شخصیت ماشین حالت برای حرکت شخصیت در هنگام روی زمین قرار گرفتن

۱۲-۳ نتیجه گیری

در این فصل نگاهی بر موتور بازی سازی آنریل با تاکید بر گراف پویانمایی انداختیم. همچنین توضیحات کاملی، درباره‌ی ویژگی‌هایی که گراف پویانمایی در اختیار کاربران می‌گذارد، آورده شد. بنابراین آشنایی کافی با ابزارهایی که یک سیستم پویانمایی در اختیار کاربران می‌گذارد و اینکه این ابزارها در چه مواردی استفاده می‌شوند، پیدا کردیم. حال برای درک بیشتر آنچه در داخل این سیستم‌ها اتفاق می‌افتد، به پیاده‌سازی یک سیستم پویانمایی از پایه می‌پردازیم.

فصل چهارم

پیاده سازی

برای پیاده سازی سیستم پویانمایی از OpenGL که یک واسط برنامه نویسی کاربردی برای دسترسی به توابع گرافیکی است، استفاده شده. در این فصل ابتدا به بررسی کتابخانه های کمکی استفاده شده در این پیاده سازی می پردازیم. سپس نحوه ی نمایش مدل های گرافیکی را بررسی کرده و پس از آن به بررسی پیاده سازی سایه زنی فانگ و فرمول های مرتبط با آن می پردازیم. در نهایت نحوه ی پیاده سازی سیستم پویانمایی و الگوریتم های به کار رفته در آن بررسی می شوند.

۴-۱ کتابخانه های کمکی

در این پیاده سازی از کتابخانه های مختلفی استفاده شده است. هر کدام از این کتابخانه ها وظیفه ی به خصوص خود را دارند. در این بخش به معرفی و بررسی این کتابخانه ها می پردازیم.

۴-۱-۱ GLFW

از آنجایی که به وجود آوردن یک پنجره ی جدید و همچنین context وابسته به نوع سیستم عامل است، بنابراین نیازمند کتابخانه ای هستیم که بتواند این موارد را برای ما مدیریت کند. GLFW یک کتابخانه ی منبع باز و چند پلتفرمی

برای OpenGL است که یک API ساده و مستقل از پلتفرم برای تولید پنجره‌ها، زمینه‌ها^۱ و سطوح، خواندن ورودی و مدیریت رویدادها^۲ را ارائه می‌کند. این کتابخانه از سیستم‌عامل‌های ویندوز، مک و لینوکس و سیستم‌های مشابه یونیکس پشتیبانی می‌کند [۲۸].

GLAD ۲-۱-۴

کتابخانه‌های گرافیکی مانند OpenGL وظیفه‌ی پیاده‌سازی توابع گرافیکی را ندارند بلکه می‌توان آن‌ها را مانند یک هدر در زبان برنامه‌نویسی C++ دانست که تعریف اولیه توابع را دارند. پیاده‌سازی این توابع در درایورهای GPU قرار دارند. دسترسی به این اشاره‌گرهای تابع به خودی خود سخت نیست ولی از آنجایی که این اشاره‌گرها وابسته به پلتفرم هستند بنابراین کار طاقت فرسایی است. وظیفه‌ی کتابخانه‌ی GLAD فراهم سازی و کنترل این اشاره‌گرهای تابع است [۲۹].

GLM ۳-۱-۴

GLM یک کتابخانه‌ی ریاضی برای نرم‌افزارهای گرافیکی مبتنی بر زبان برنامه‌نویسی سایه‌ی OpenGL^۳ است. این کتابخانه تنها شامل یک هدر C++ است. توابع و کلاس‌های موجود در این کتابخانه به صورتی نامگذاری و طراحی شده‌اند تا بسیار به GLSL نزدیک باشند.

Assimp ۴-۱-۴

Assimp یک کتابخانه برای بارگذاری و پردازش صحنه‌های هندسی با فرمت‌های مختلف است. می‌توان با استفاده از آن مواردی همچون مش‌های استاتیک و یا اسکلتی، مواد^۴، کلیپ‌های پویانمایی اسکلتی و داده‌های بافت را از فایل بارگذاری کرد. زمانی که این مدل‌ها بارگذاری می‌شوند این کتابخانه آن‌ها را در ساختاری مانند تصویر ۴-۱ ذخیره می‌کند و بعد از آن می‌توان از این ساختار، داده‌های مورد نظر خود را خواند و از آن‌ها استفاده کرد [۳۰] [۳۱].

stb ۵-۱-۴

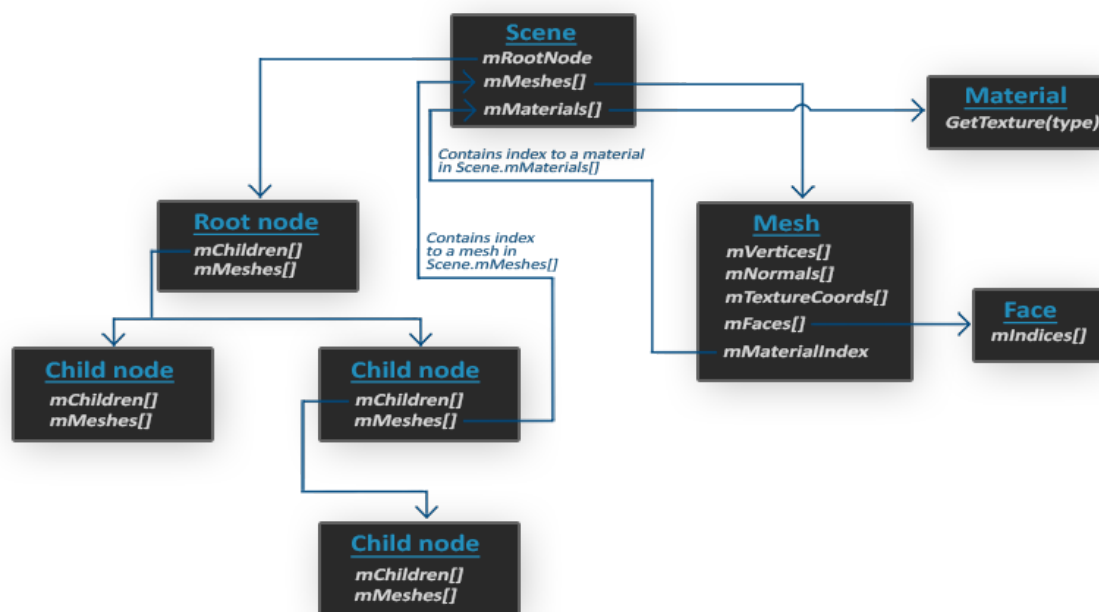
این کتابخانه برای بارگذاری تصاویر استفاده می‌شود. در این پروژه از این کتابخانه برای بارگذاری تصاویر بافت‌ها در کنار کتابخانه‌ی Assimp استفاده شده است [۳۲].

¹Contexts

²Events

³OpenGL Shading Language(GLSL)

⁴Materials



شکل ۴-۱ - ساختار کلاس‌های کتابخانه‌ی Assimp [۳۱]

۲-۴ پیاده‌سازی سیستم پویانمایی

این بخش دو هدف کلی را دنبال می‌کند.

- نمایش مدل گرافیکی و نورپردازی محیط

- اجرا و ترکیب کلیپ‌های پویانمایی توسط ماشین حالت متناهی

۱-۲-۴ نمایش مدل گرافیکی

همانطور که گفته شد مدل‌ها یا اشیاء سه‌بعدی به خودی خود مفهومی در OpenGL ندارند. آنچه برای OpenGL اهمیت دارد لیستی از مثلث‌ها است تا آن‌ها را به تصویر بکشد. مدل‌های سه‌بعدی از رئوس، لبه و وجوه تشکیل می‌شوند و در فرمت‌های مختلفی مانند FBX ذخیره می‌شوند. در این پیاده‌سازی، از کتابخانه‌ی Assimp برای خواندن این داده‌ها استفاده شده است.

۲-۲-۴ قراردگیری مدل سه‌بعدی در کارت گرافیک

آنچه برای OpenGL اهمیت دارد این است که به آن مجموعه‌ای از مثلث‌ها داده شود تا برایمان ترسیم کند. برای اینکار به صورت عمومی از ۳ آرایه مختلف استفاده می‌شود که به نام‌های VBO، VAO و EBO شناخته می‌شوند. VBOs^۱ یک آرایه یا بافری است که تمامی رئوس مدل سه‌بعدی ما را در خود جای می‌دهد.

^۱Vertex Buffer Objectss

همانطور که در بخش ۲-۷-۱ اشاره شد، رئوس علاوه بر اینکه شامل اطلاعات موقعیت مکانی در محیط سه بعدی هستند، شامل اطلاعات دیگری نظیر رنگ، بردار نرمال و مختصات بافت نیز می توانند باشند. بنابراین باید به صورتی به کارت گرافیک اعلام کنیم که این داده ای که در آرایه ی VBOs قرار دارد را چگونه تفسیر کند. اینکار با استفاده از یک آرایه ی دیگر به نام VAO^۱ صورت می گیرد. در نهایت گفتیم که آنچه برای کارت گرافیک اهمیت دارد دریافت مثلث ها است. بنابراین باید به طریقی بگوییم کدام رئوس با اتصال به یکدیگر مثلث تشکیل می دهند. اینکار نیز با استفاده از آرایه ی EBOs^۲ صورت می گیرد.

۳-۲-۴ سایه زنی فانگ

پس از نمایش مدل سه بعدی در محیط گرافیکی به بررسی نحوه ی نورپردازی آن با استفاده از سایه زنی فانگ، می پردازیم. همانطور که در ۲-۶-۲ مطرح شد، الگوریتم سایه زنی فانگ شامل سه مولفه ی اصلی نور محیطی^۳، نور پخش شده^۴ و نور آینه وار^۵ می شود. در نور محیطی تنها عامل تاثیر گذاری، میزان قدرت منبع نور است. در نور پراکنده، جهت قرار گیری منبع نور برای ما اهمیت پیدا می کند و در نهایت در نورپردازی آینه وار، موقعیت بیننده به معادله اضافه می شود. هر کدام از این مولفه ها یک میزان روشنایی به ما داده و رنگ نهایی شیء به صورت زیر محاسبه می شود.

$$result = (ambient + diffuse + specular) * objectColor$$

نور محیطی

نور محیطی قدرت نورپردازی منبع نور را برای ما شبیه سازی می کند. برای افزودن نور محیطی به اشیاء موجود در صحنه ی سه بعدی بدین صورت عمل می کنیم. رنگ نور منبع نور را گرفته و آن را در یک ضریب محیطی ثابت کوچک ضرب کرده. این خروجی نور محیطی است.

$$ambientStrength = 0.1$$

$$ambient = ambientStrength * lightColor$$

¹Vertex Array Objects

²Element Buffer Objects

³Ambient

⁴Diffuse

⁵Specular

نور پخش شده

هرچه قطعات یک چندضلعی در جهت پرتوهای نور منبع نور باشند، نور پخش شده به آن قسمت، روشنایی بیشتری می‌بخشد. بنابراین این مولفه تاثیر زاویه قرارگیری قطعات شیء با منبع نور را شبیه‌سازی می‌کند.

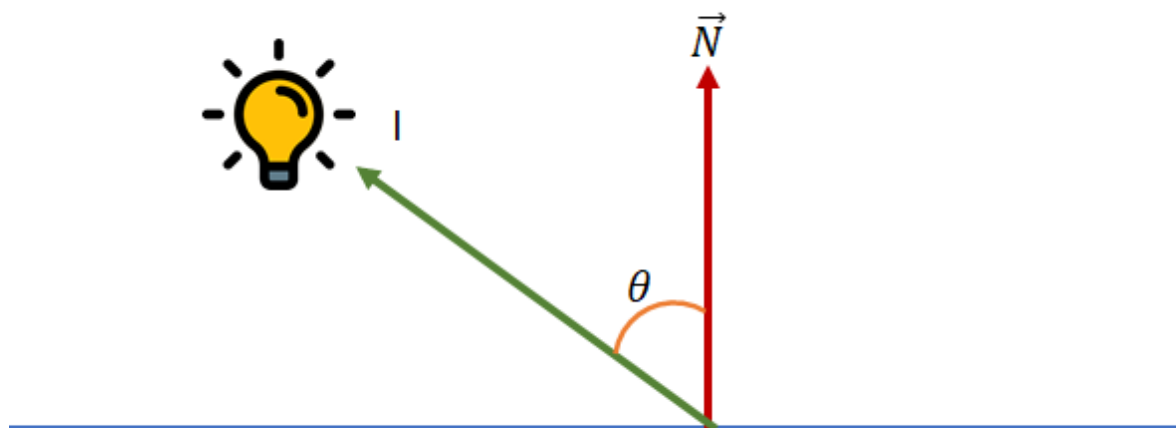
هر یک از قسمت‌های شیء شامل یک بردار نرمال است. بردار نرمال برداری واحد و عمود بر شیء است. با دانستن موقعیت مکانی منبع نور، می‌توان برداری از قطعه‌ی موجود در شیء به سمت منبع نور را بدست آورد. زاویه‌ی بین این بردار و بردار نرمال، زاویه‌ی بین قطعه‌ی شیء و منبع نور است. (شکل ۴-۲) برای اینکه بفهمیم میزان تاثیرگذاری این نور بر روی آن قطعه چقدر است می‌توان از ضرب نقطه‌ای^۱ این دو بردار استفاده کرد. سپس مقدار خروجی را در مقدار رنگ منبع نور ضرب کرده و این میزان نور پخش شده‌ی ما می‌شود.

$$norm = normalize(Normal)$$

$$lightDir = normalize(lightPos - FragPos)$$

$$diff = max(dot(norm, lightDir), 0.0)$$

$$diffuse = diff * lightColor$$



شکل ۴-۲ - نور پخش شده

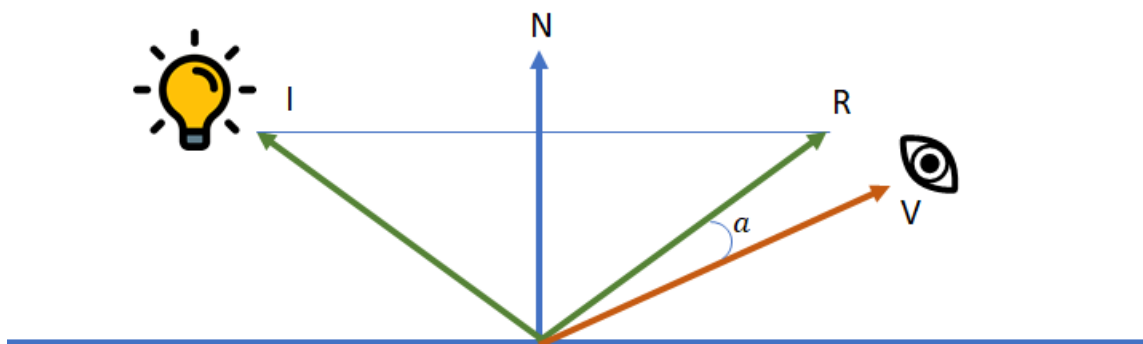
^۱ Dot product

نور آینه‌وار

مانند نور پخش شده، نور آینه‌وار نیز بستگی به زاویه‌ی قرار گیری منبع نور و بردار نرمال قطعه دارد. تفاوت این مولفه این است که علاوه بر مورد اشاره شده وابسته به جهت مشاهده نیز است. در شکل ۳-۴ می‌توان دیاگرام مربوط به این نوع نور را مشاهده کرد.

نورپردازی آینه‌وار وابستگی بسیاری به خواص بازتابی سطوح دارد. اگر سطح جسم را مانند یک آینه در نظر بگیریم، جایی را که بتوانیم نور منعکس شده را بر روی سطح بینیم شدت آن نور قوی‌تر خواهد بود. بنابراین نتیجه‌ی نهایی این نورپردازی بدین صورت است که اجسام از زوایای خاصی روشن‌تر به نظر می‌رسند و هرچه از اجسام دورتر شویم، از این روشنایی کاسته می‌شود.

برای بدست آوردن شدت این نور نیاز داریم ابتدا زاویه‌ی قرار گیری منبع نور با قسمت مورد نظر روی شیء را پیدا کنیم. این زاویه را در نور پخش شده نیز بدست آوردیم. سپس لازم است آن را نسبت به بردار نرمال شیء بازتاب دهیم. (در شکل ۴-۴ نحوه‌ی بدست آوردن بردار منعکس شده R آورده شده است.) همچنین لازم است بردار بین بیننده و آن قطعه را نیز بدست آوریم. در نهایت زاویه‌ی بین بردار نور بازتاب داده شده و بردار موقعیت بیننده، زاویه‌ی مورد نظر ما است و برای بدست آوردن تاثیر شدت نور آینه‌وار می‌توان از ضرب نقطه‌ای این دو بردار استفاده کرد. از آنجایی که این نور بستگی به میزان براق بودن شیء دارد، بنابراین مانند فرمول زیر لازم است خروجی ضرب نقطه‌ای را به توان یک عددی که این عدد میزان براق بودن است برسانیم [۳۳] [۳۴].



شکل ۳-۴ - نور آینه‌وار

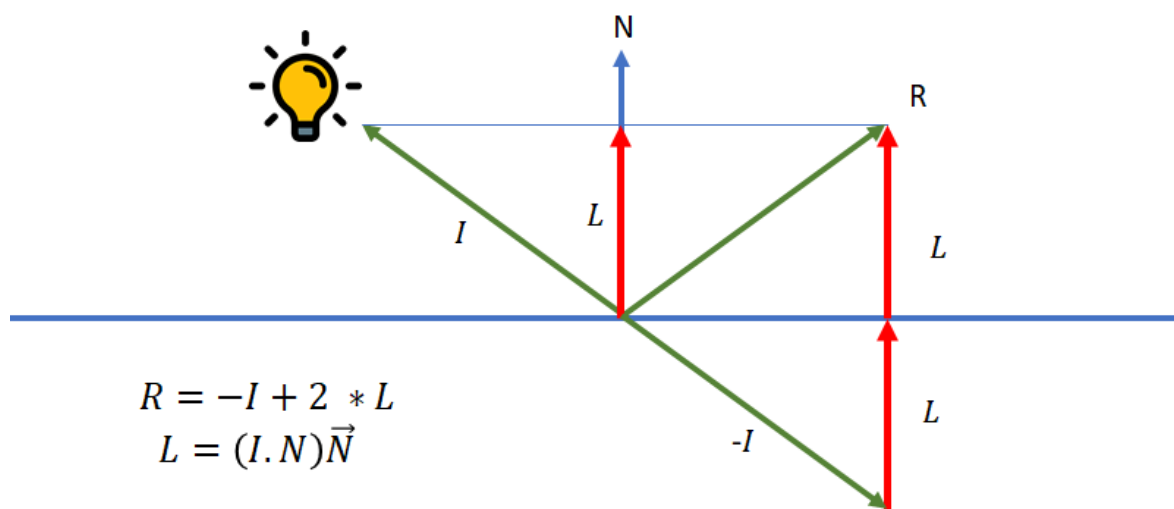
$specularStrength = 0.5$

$viewDir = normalize(viewPos - FragPos)$

$reflectDir = reflect(-lightDir, norm)$

$spec = pow(max(dot(viewDir, reflectDir), 0.0), 32)$

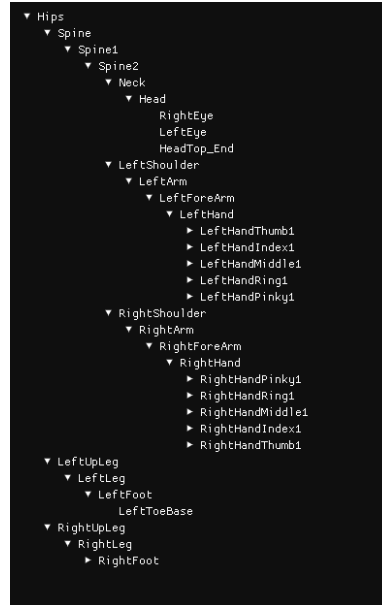
$specular = specularStrength * spec * lightColor$



شکل ۴-۴ - بدست آوردن بردار پرتوی منعکس شده

۴-۲-۴ اسکلت شخصیت

اسکلت یک شخصیت به صورت مجموعه‌ای از مفاصل که به صورت سلسله مراتبی به یکدیگر متصل‌اند، تعریف می‌شود. در این پیاده‌سازی کلاس Bone نشان‌دهنده‌ی هر مفصل است. هر Bone یک والد دارد و می‌تواند به هر تعدادی فرزند داشته باشد. با توجه به تعریف آورده شده از اسکلت، کلاس اسکلت که با Skeleton مشخص شده، شامل لیستی از این مفاصل به همراه اشاره‌گری به مفصل ریشه است. به عنوان مثال یک اسکلت انسانی در شکل ۴-۵ به صورت یک درخت نشان داده شده است.



شکل ۴-۵- نمایی از ساختار سلسله مراتبی اسکلت شخصیت. در اینجا مفصل Hips، مفصل ریشه است.

۴-۲-۵ اتصال اسکلت و مدل سه بعدی

اصطلاحی که برای اتصال اسکلت و مدل سه بعدی استفاده می شود Skinning است. در این روش هر راس موجود در مدل، به یک یا چند مفصل متصل می شود. الگوریتم به کاررفته در این پیاده سازی، الگوریتم linear blend skinning نام دارد. در این الگوریتم زمانی که یک راس به یک مفصل می شود به آن یک وزن نسبت داده می شود. این وزن نشان دهنده میزان تاثیر گذاری این مفصل بر روی این راس است. به بیانی دیگر، این وزن نشان می دهد که اگر این مفصل به مکان جدید منتقل شود، این انتقال چقدر بر روی آن راس تاثیر می گذارد. بنابراین برای بدست آوردن انتقال نهایی راس، باید انتقال راس را نسبت به هر کدام از مفاصلی که به آن متصل است را بدست آوریم، سپس انتقال نهایی برابر مجموع وزن دار تمامی این انتقال ها خواهد بود.

فرمول نهایی انتقال هر راس بر اساس این الگوریتم به صورت زیر محاسبه می شود.

$$v_M^C = \sum_{i=1}^N w_i^j K_i v_M^B$$

مقدار v_M^C بیانگر مکان راس نسبت به مختصات ریشه مش در حالت فعلی است.

مقدار v_M^B بیانگر مکان راس نسبت به مختصات ریشه مش در هنگامی که مش در حالت اتصال قرار دارد، است. ماتریس K_i انتقال است که بر روی راس انجام می شود. مقدار w_i^j بیانگر میزان وزن تاثیر گذاری مفصل j بر روی این راس است. مقدار N بیانگر تعداد مفاصلی است که روی این راس تاثیر می گذارند.

برای انتقال رئوس از ماتریس انتقال K_i استفاده شده است. این ماتریس ابتدا راس را از فضای مختصاتی مش به فضای مفصل مورد نظر برده، سپس مفصل را با استفاده از ماتریس انتقال کلیپ پویانمایی، انتقال داده و پس از آن راس را دوباره به فضای مختصاتی مش باز می گرداند. زمانی که رئوس در فضای مفصل قرار می گیرند و پس از آن مفصل انتقال پیدا می کند، مکان راس نسبت به مفصل تغییر نمی کند. بنابراین به همین علت است که ابتدا رئوس به فضای مفصل موردنظر برده می شود و سپس آن مفصل را انتقال داده و در نهایت راس را به فضای مختصاتی مش باز می گرداند. می توان حالت باز شده ی این ماتریس را در فرمول زیر مشاهده کرد.

$$K_i = P_{J \rightarrow M} A_J B_{M \rightarrow J}$$

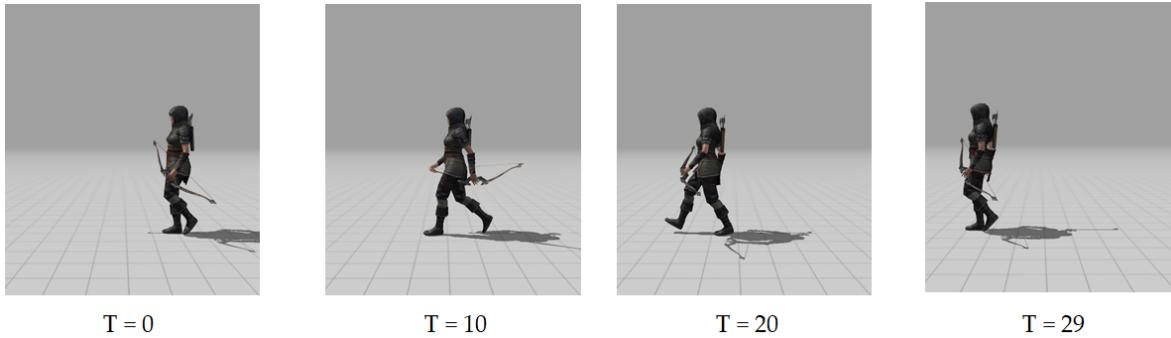
این ماتریس ها برای انتقال و تغییر فضای مختصات استفاده می شوند. زمانی که ماتریس K_i در راس ضرب می شود، اولین ماتریسی که روی راس تاثیر می گذارد، ماتریس $B_{M \rightarrow J}$ است. این ماتریس فضای مختصات راس را از فضای مختصات مش به فضای مختصات آن مفصل مورد نظر تغییر می دهد. پس از این ماتریس، ماتریس انتقال A_J اعمال می شود که این ماتریس از کلیپ پویانمایی برای این مفصل بدست می آید. و در نهایت ماتریس $P_{J \rightarrow M}$ اعمال می شود که این ماتریس، راس را به فضای مختصاتی مش باز می گرداند.

۶-۲-۴ کلیپ پویانمایی

در بازی های کامپیوتری هر کلیپ پویانمایی شامل یک حرکت منحصر به فرد شخصیت داخل بازی است. هر کلیپ شامل ژست های اسکلت در فاصله های زمانی مشخصی است. (شکل ۶-۴) در واقع آنچه باعث حرکت شخصیت می شود حرکت اسکلت شخصیت است. زمانی که اسکلت شخصیت با استفاده از یک کلیپ جابه جا می شود، مدل شخصیت نیز با استفاده از روش های skinning که در بالا توضیح داده شد همراه این اسکلت حرکت می کند. کلیپ های پویانمایی از طریق کلاسی به اسم Animation Clip مدل سازی شده اند. این کلاس شامل آرایه ای از ژست های شخصیت در مدت زمان های مشخصی است. همراه یک اشاره گری به اسکلت شخصیت. نکته ی قابل توجه این است که هر کلیپ پویانمایی مربوط به یک نوع اسکلت می شود. به زبانی دیگر نمی توان کلیپی که برای اسکلت شخصیت انسانی طراحی شده است را بر روی یک حیوان، مانند فیل اجرا کرد.

۷-۲-۴ پخش کننده ی کلیپ های پویانمایی

این سیستم وظیفه اش پخش کردن کلیپ پویانمایی بر روی اسکلت شخصیت است. این سیستم با گرفتن یک کلیپ و یک اسکلت، کلیپ موردنظر را بر روی آن اسکلت اجرا می کند. همانطور که گفتیم، کلیپ های پویانمایی ژست



شکل ۴-۶- نمایش از ژست شخصیت در یک کلیپ پویانمایی

شخصیت را در فاصله‌های زمانی مشخصی در خود ذخیره می‌کنند. وظیفه‌ی این سیستم این است که با استفاده از یک زمان‌سنج که نشان‌دهنده‌ی زمان فعلی بازی است، ژست مناسب شخصیت را از داخل کلیپ بدست آورد. قابل ذکر است که ممکن است این ژست با توجه به زمان بازی و فاصله‌های زمانی داخل کلیپ از درون‌یابی دو ژست پشت سر هم در آن کلیپ بدست آید.

۸-۲-۴ الگوریتم پخش‌کننده‌ی کلیپ‌های پویانمایی

هر شخصیت درون بازی، اگر از نوع شخصیت اسکلتونی باشد، دارای یک پخش‌کننده‌ی کلیپ پویانمایی خواهد بود.

در تصویر زیر تابع به‌روزرسانی اسکلت به وسیله‌ی کلیپ پویانمایی را می‌توان مشاهده کرد.

```
currentTime += deltaTime;

const double currentAnimationTime = (currentTime - startTimeForCurrentAnim);
AnimationPose currentPose = currentClip->GetPoseForCurrentFrame(currentAnimationTime *
    currentClip->GetFramePerSecond());

SetSkeletonPose(currentPose);
```

برای اینکه بتوان یک کلیپ را پخش کرد نیاز است دو مورد زیر را بدانیم.

- زمان فعلی درون بازی (CurrentTime)

- زمان شروع پخش کلیپ فعلی (StartTimeForCurrentAnimation)

در ابتدا زمان فعلی درون بازی را برای این پخش‌کننده به‌روزرسانی می‌کنیم. سپس برای بدست آوردن زمان فعلی

کلیپ می‌توان از فرمول زیر استفاده کرد

$$CurrentAnimationTime = CurrentTime - StartTimeForCurrentAnimation$$

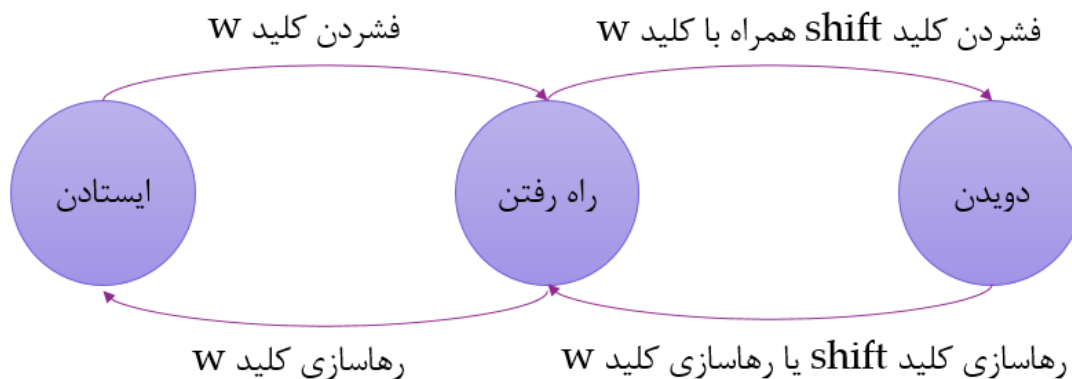
در نهایت با استفاده از این مقدار می‌توان ژست مورد نظر را از داخل کلیپ پویانمایی بدست آورد. در نهایت نیز این ژست را بر روی اسکلت شخصیت اعمال می‌کنیم.

۹-۲-۴ ماشین حالت پویانمایی

یکی از روش‌های ترکیب کلیپ‌های مختلف با یکدیگر، استفاده از ماشین حالت متناهی است. یک ماشین حالت متناهی شامل چندی حالت مختلف است که هر کدام از این حالات، حالتی از وضعیت سیستم را مشخص می‌کنند. زمانی که از ماشین حالت استفاده می‌شود سیستم می‌تواند در هر لحظه تنها در یکی از این حالات قرار گیرد. البته سیستم می‌تواند با دریافت ورودی از یک حالت به حالت دیگری رود.

دلیل استفاده از ماشین حالت متناهی برای سیستم پویانمایی این است که همانگونه که گفتیم، کلیپ‌های پویانمایی، شامل ویدیوهای کوتاهی هستند که یک حالت مشخصی از شخصیت را بیان می‌کنند. در یک بازی، با توجه به ورودی بازیکن، شخصیت درون بازی می‌تواند در حالت‌های متفاوتی قرار گیرد. با استفاده از ماشین حالت می‌توان به تمامی این حالت‌ها رسیدگی کرد.

به عنوان مثال، شکل ۷-۴ نشان‌دهنده‌ی یک ماشین حالت برای حرکت شخصیت است. شخصیت در ابتدا در حالت ایستاده قرار دارد و با گرفتن ورودی‌های مختلف از کیبورد، می‌تواند به حالت‌های دیگری رود.



شکل ۷-۴ - ماشین حالت برای حرکت شخصیت

برای پیاده‌سازی ماشین حالت متناهی، این سیستم به سه کلاس کلی شکسته شده است. کلاس AnimationStateMachine که وظیفه‌ی مدیریت حالت‌ها و انتقال از یک حالت به حالت دیگری را دارد. کلاس AnimationState که نشان‌دهنده‌ی حالت شخصیت است. هر AnimationState شامل یک کلیپ است و هر زمانی که این حالت فعال می‌شود این کلیپ پخش می‌شود. در نهایت کلاس Transition که شامل توابع انتقال است. هر حالت می‌تواند شامل چندین انتقال باشد. و وظیفه‌ی AnimationStateMachine است که بررسی کند، اگر انتقالی امکان‌پذیر بود، آن را انجام دهد.

۴-۲-۱۰ به روزرسانی ماشین حالت پویانمایی

وضعیت توابع انتقال تاثیر گذاری مستقیمی در وضعیت سیستم به روزرسانی ماشین حالت دارد. وضعیت انتقال می تواند سه حالت زیر را داشته باشد.

- حالت عادی^۱
- حالت در حال انتقال^۲
- حالت اتمام انتقال^۳

حالت اول حالت عادی است که نشان دهنده ی وضعیت عادی ماشین حالت است. در این وضعیت، توابع انتقال حالت فعلی بررسی می شوند تا در صورتی که شرایطشان برقرار شود، تغییر حالت رخ دهد. علاوه بر آن کلیپ حالت فعلی با استفاده از کلاس پخش کننده آپدیت می شود.

در صورتی که توابع انتقال مقدار درست^۴ را بازگردانند، ماشین به وضعیت دوم که وضعیت در حال انتقال است، تغییر وضعیت می دهد. در این وضعیت با توجه به زمانی که مشخص شده، ژست شخصیت با استفاده از درونیایی خطی از حالت فعلی به حالت جدید تغییر می کند.

پس از اینکه انتقال به صورت کامل انجام شد، وضعیت ماشین حالت به اتمام انتقال تغییر می یابد. زمانی که ماشین در این وضعیت قرار گرفته یعنی به حالت جدید منتقل شده، بنابراین لازم است کلیپ پویانمایی را از حالت جدید گرفته و آن را به کلاس پخش کننده داده تا آن را پخش کند. پس از این کار وضعیت ماشین دوباره به حالت عادی تغییر می یابد و همه ی این موارد دوباره تکرار می شوند.

```

if(transitionStatus == TransitionStatus::normal)
{
    for (const auto& transition : currentState->GetTransitions()) // loop through
        transitions of the current state
    {
        if (transition->Evaluate())
        {
            transitionStatus = TransitionStatus::transitioning;
            currentState = animationStatesMap.at(transition->to);
            TransitionFromPose = animator->GetPoseAtCurrentTime();
            TransitionToPose = currentState->GetAnimClip()->GetPoseForCurrentFrame(0);
            currentTime = 0;
            transitionTime = transition->transitionTime;
            break;
        }
    }
}

if (transitionStatus == TransitionStatus::normal)

```

¹Normal

²Transitioning

³Finished

⁴True

```

{
    animator->Update(deltaTime);
}
else if(transitionStatus == TransitionStatus::transitioning)
{
    if(TransitionUpdate(deltaTime))
    {
        transitionStatus = TransitionStatus::finished;
    }
}
else if(transitionStatus == TransitionStatus::finished)
{
    animator->ChangeAnimationClip(*(currentState->GetAnimClip()), 0);
    transitionStatus = TransitionStatus::normal;
}

```

۳-۴ نتیجه‌گیری

خروجی نهایی این فصل، یک برنامه‌ی گرافیکی برای بارگذاری اشیاء سه‌بعدی است. از آنجایی که این اشیاء می‌توانند شامل مش‌های اسکلتی باشند، بنابراین توانایی اجرای کلیپ‌های پویانمایی را خواهند داشت. در نهایت با پیاده‌سازی سیستم ترکیب با استفاده از ماشین حالت متناهی، این مش‌های اسکلتی می‌توانند در حالت‌های مختلف قرار گیرند و بر اساس هر کدام از این حالات، کلیپ متفاوتی را پخش کنند. فایل نهایی و تصاویر و ویدیوهای خروجی این نرم‌افزار در [۳۵] قابل مشاهده است.

فصل پنجم

نتیجه گیری

هدف این پروژه آشنایی با محیط‌های گرافیکی و الگوریتم‌های موجود در آن با تاکید بر سیستم پویانمایی کامپیوتری بود. برای بدست آوردن این هدف، در این پروژه ابتدا به بررسی سیستم گراف پویانمایی یکی از بزرگترین موتورهای بازی‌سازی جهان، یعنی موتور بازی‌سازی آنریل پرداختیم. با این بررسی متوجه شدیم که یک سیستم پویانمایی چه ابزارهایی را در اختیار کاربران قرار می‌دهد و نحوه‌ی کلی استفاده از این ابزارها چگونه است.

در نهایت برای تحلیل عمیق این سیستم‌ها به پیاده‌سازی یک سیستم مشابه با استفاده از واسط برنامه نویسی کاربردی OpenGL پرداختیم. خروجی این پیاده‌سازی، یک نرم‌افزاری گرافیکی است که به وسیله‌ی آن می‌توان مش‌های اسکلتی را بارگذاری کرد و روی آن‌ها کلیپ‌های پویانمایی مختلفی را اجرا کرد.

نرم‌افزار گرافیکی پیاده‌سازی شده می‌تواند از جهات مختلفی گسترش یابد. یکی از مواردی که می‌توان اشاره کرد اضافه کردن ویژگی‌های جدید به نرم‌افزار فعلی است. همانطور که در تحقیق درباره‌ی سیستم آنریل متوجه شدیم، سیستم‌های پویانمایی، سیستم‌های بسیار گسترده‌ای هستند. ویژگی‌هایی مانند اضافه کردن الگوریتم‌های مختلف برای ترکیب، اضافه کردن پویانمایی بر اساس فیزیک، اضافه کردن مواردی همچون سینماتیک معکوس برای ایجاد پویانمایی رویه‌ای می‌توانند تنها سطحی از دریای عمیق ویژگی‌ها باشند.

علاوه بر این، از آنجایی که این برنامه‌ی در حال حاضر از کتابخانه‌ی Assimp برای بارگذاری مدل‌های سه‌بعدی

استفاده می‌کند، سرعت مناسبی ندارد. می‌توان با نوشتن یک سیستم جداگانه برای بارگذاری اشیاء به سرعت این بارگذاری افزود.

علاوه بر این موارد، این برنامه را می‌توان از جهت موتور بازی‌سازی نیز ارتقا بخشید. به عنوان مثال، در سیستم فعلی هیچگونه الگوریتم برخوردی، پیاده‌سازی نشده است. با پیاده‌سازی چنین مواردی، می‌توان به واقع‌گرایانه‌تر شدن این برنامه کمک کرد.

فهرست شکل ها

صفحه	عنوان
۷.....	شکل ۲-۱: فریم های کلیدی و درمیان [۴].....
۷.....	شکل ۲-۲: چشم انداز چند منظوره [۵].....
۸.....	شکل ۲-۳: لایه های مختلف [۶].....
۱۳.....	شکل ۲-۴: مولفه های الگوریتم سایه زنی فانگ [۱۵].....
۱۵.....	شکل ۲-۵: نمایش یک دلفین به وسیله ی مش مثلثی [۱۶].....
۱۵.....	شکل ۲-۶: عناصر یک مش چندضلعی [۱۶].....
۱۶.....	شکل ۲-۷: اجزاء مختلف ماشین [۱۷].....
۱۷.....	شکل ۲-۸: چند نوع ماده ی مختلف [۱۸].....
۱۸.....	شکل ۲-۹: چند نوع بافت مختلف [۱۹].....
۱۸.....	شکل ۲-۱۰: در سمت راست مش شخصیت و سمت چپ اسکلت شخصیت قابل مشاهده است. [۲۰].....
۱۸.....	شکل ۲-۱۱: عکس نشان دهنده ی میزان تعلق رئوس اطراف مفصل شانه است. هرچه راس به مفصل نزدیک تر باشد، تعلق بیشتری به آن دارد (رنگ قرمز تری می گیرند) [۲۱].....
۲۶.....	شکل ۳-۱: نمونه ای از یک گراف رویداد.....
۲۶.....	شکل ۳-۲: نمونه ای از یک گراف پویانمایی.....
۲۷.....	شکل ۳-۳: ساختار کلی گره های موجود در گراف پویانمایی.....
۲۸.....	شکل ۳-۴: پنل تنظیمات گره ی ترکیب.....
۲۹.....	شکل ۳-۵: نمونه ای از جریان اجرا.....
۲۹.....	شکل ۳-۶: نمونه ای از گره ی دنباله ی پویانمایی.....
۳۰.....	شکل ۳-۷: در این فضای حالت، سرعت بین ۰ تا ۵۰۰ متغیر است و زاویه بین -۱۲۰ تا ۱۲۰ متغیر است. (در شکل به صورت کمی نوشته شده است) در کل از ۱۰ کلیپ پویانمایی شده (نقاط سفید رنگ در تصویر) ولی با استفاده از ترکیب بین این کلیپ ها می توان تمامی مقادیر بین کلیپ ها را نیز بدست آورد.....
۳۱.....	شکل ۳-۸: گره ی ترکیب.....
۳۲.....	شکل ۳-۹: مثال استفاده از گره ی ترکیب.....
۳۲.....	شکل ۳-۱۰: اجرای کلیپ های پویانمایی بر روی مفاصل متفاوت.....
۳۳.....	شکل ۳-۱۱: گره های تبدیل حالت.....
۳۴.....	شکل ۳-۱۲: ماشین حالت برای حرکت شخصیت.....
۳۴.....	شکل ۳-۱۳: شخصیت ماشین حالت برای حرکت شخصیت در هنگام روی زمین قرار گرفتن.....
۳۷.....	شکل ۴-۱: ساختار کلاس های کتابخانه ی Assimp [۳۱].....
۳۹.....	شکل ۴-۲: نور پخش شده.....
۴۰.....	شکل ۴-۳: نور آینه وار.....
۴۱.....	شکل ۴-۴: بدست آوردن بردار پرتوی منعکس شده.....
۴۲.....	شکل ۴-۵: نمایی از ساختار سلسله مراتبی اسکلت شخصیت. در اینجا مفصل Hips، مفصل ریشه است.....
۴۴.....	شکل ۴-۶: نمایی از ژست شخصیت در یک کلیپ پویانمایی.....

شکل ۴-۷: ماشین حالت برای حرکت شخصیت ۴۵

مراجع

- [1] “Animation wikipedia”, <https://en.wikipedia.org/wiki/Animation>.
- [2] Barczak, A. M. and Woźniak, H., “Comparative study on game engines”, *Studia Informatica. Systems and Information Technology. Systemy i Technologie Informacyjne*, No. 1-2, 2019.
- [3] “Unreal engine skeletalmesh”, https://ocw.mit.edu/courses/6-837-computer-graphics-fall-2012/9ab6e7460a5b93d6f97b08f95de23ebc_MIT6_837F12_Lec06.pdf.
- [4] “Keyframe animarion berkeley”, <https://cs184.eecs.berkeley.edu/sp19/lecture/17-28/intro-to-animation-kinematics-mo>.
- [5] Wood DN, Finkelstein A, H. J. T. C. and DH, S., “Multiperspective panoramas for cel animation”, *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997.
- [6] “Animation cells wikipedia”, https://en.wikipedia.org/wiki/File:Animation_cells.png.
- [7] Lasseter, J., “Principles of traditional animation applied to 3d computer animation”, *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987.
- [8] Gregory, J., *Game Engine Architecture*, A K Peters/CRC Press, 3rd ed. , 2018.
- [9] “Unreal engine wikipedia”, https://en.wikipedia.org/wiki/Unreal_Engine.
- [10] “Unreal engine blueprint”, <https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Blueprints/>.
- [11] “Computer graphics wikipedia”, [https://en.wikipedia.org/wiki/Computer_graphics_\(computer_science\)](https://en.wikipedia.org/wiki/Computer_graphics_(computer_science)).
- [12] “Using opengl”, https://www.khronos.org/opengl/wiki/Getting_Started.
- [13] “Opengl state machine”, <https://learnopengl.com/Getting-started/OpenGL>.
- [14] “Phone shading learnopengl”, <https://learnopengl.com/Lighting/Basic-Lighting>.
- [15] “Phong shading wikipedia”, https://en.wikipedia.org/wiki/Phong_reflection_model.
- [16] “Polygon mesh wikipedia”, https://en.wikipedia.org/wiki/Polygon_mesh.
- [17] “Different parts of a car”, <https://cults3d.com/en/3d-model/game/bugatti-veyron-printable-car-3d-digital-stl-file>.
- [18] “Different materials”, <https://cleverlottery.weebly.com/vray-materials-download.html>.
- [19] “Different textures”, <https://www.pinterest.com/pin/56787645294557990/>.

- [20] “Skeletal model”, <https://www.fudgeanimation.com/experiments/top-5-rigging-tips/>.
- [21] “Skinning”, <https://www.pluralsight.com/blog/film-games/understanding-skinning-vital-step-rigging-project>.
- [22] “Unreal engine animation”, <https://docs.unrealengine.com/5.0/en-US/animating-characters-and-objects-in-unreal-engine/>.
- [23] “Unreal engine event graph”, <https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/EventGraph/>.
- [24] “Unreal engine animation graph”, <https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/AnimBlueprints/AnimGraph/>.
- [25] “Unreal engine anim graph blend node”, <https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/NodeReference/Blend/>.
- [26] “Unreal engine anim graph skeletal controls node”, <https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/NodeReference/SkeletalControls/>.
- [27] “Unreal engine anim graph change space node”, <https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/NodeReference/SpaceConversion/>.
- [28] “Glfw”, <https://github.com/glfw/glfw>.
- [29] “Glad”, <https://github.com/Dav1dde/glad>.
- [30] “Assimp”, <https://assimp-docs.readthedocs.io/en/v5.1.0/about/introduction.html>.
- [31] “Assimp class hierarchy”, <https://learnopengl.com/Model-Loading/Assimp>.
- [32] “stb”, <https://github.com/nothings/stb>.
- [33] “ogldev specular lighting”, <https://ogldev.org/www/tutorial19/tutorial19.html>.
- [34] “Mit shading and material appearance”, https://ocw.mit.edu/courses/6-837-computer-graphics-fall-2012/b86de2fe1a1084e56148f293da4a4dfd_MIT6_837F12_Lec15.pdf.
- [35] Naziri, N., “Skeletal animation implementation”, <https://github.com/NamiNaziri/Skeletal-Animation>.

Analysis of the animation graph in Unreal Engine and implementation of an animation system using OpenGL

Nami Naziri

nami.naziri@yahoo.com

July 20, 2022

Department of Electrical and Computer Engineering

Isfahan University of Technology, Isfahan 84156-83111, Iran

Degree: Bachelor of Science

Language: Farsi

Supervisor: Maziar Palhang, Assoc. Prof., palhang@cc.iut.ac.ir.

Abstract

Computer animation is the process used for digitally generating animated images. Modern computer animation usually uses 3D computer graphics to generate a three-dimensional picture. In most 3D computer animation systems, an animator creates a simplified representation of a character's anatomy, which is analogous to a skeleton or stick figure. In human and animal characters, many parts of the skeletal model correspond to the actual bones.

In the first phase, the animation graph will be examined in Unreal Engine. The animation graph is used to evaluate a final pose for the Skeletal Mesh for the current frame. This graph is used to sample, blend, and manipulate poses to be applied to Skeletal Meshes by the Animation Blueprints. We will examine the graph and the algorithm it uses at this stage.

In the second phase, the skeletal animation system is implemented from the ground up, based on the methods obtained. This step has three objectives. The first objective is to render a skeleton in a 3D environment created using OpenGL. As part of this step, we will write a program in C++ that will display the animation created by key frames. As the Second objective, skinning methods are used to add a mesh to the skeleton. Ultimately, we want to use the animation blending method to blend together different animations.

Keywords

Computer Animation, Game Engine, Unreal Engine, Animation Graph, 3D Computer Graphics



Isfahan University of Technology

Department of Electrical and Computer Engineering

Analysis of the animation graph in Unreal Engine and implementation of an animation system using OpenGL

A Thesis

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science

By

Nami Naziri

Evaluated and Approved by the Thesis Committee, on July 20, 2022

1- Maziar Palhang, Assoc. Prof. (Supervisor)

2- Zeinab Zali, Assist. Prof. (Examiner)

Department Graduate Coordinator: Ahmadreza Tabesh, Assoc. Prof.