بسم الله الرّحمن الرّحیم

Delta-time
and
framerate independence

مجتبی قاسم زاده تهرانی

si2.ir

شهریور ۱۳۹۷

مقدّمه

فریم‌ریت چیست؟ همان طور که می‌دانید تصاویر متحرّک از نمایش تعداد بسیار زیادی تصویر ثابت به صورت پشت سر هم به وجود می‌آیند. مثلاً ۳۰ تصویر ثابت پشت سر هم در یک ثانیه به نمایش در می‌آید و در نتیجه چشم ما قادر به تشخیص نیست و آن را متحرّک می‌بیند، در این حالت می‌گوییم فریم‌ریت 30 FPS است، یعنی در یک ثانیه ۳۰ تصویر ثابت پشت سر هم دیده می‌شود (FPS مخفف Frames Per Second به معنی فریم بر ثانیه است). در بازی‌های کانستراکت هم دقیقاً همین قضیه وجود دارد و اگر ما چیزی را در بازی متحرّک می‌بینیم به خاطر وجود همین عکس‌های ثابت پشت سر هم است.

بازی «**مستقل از فریم‌ریت**» به بازی‌هایی گفته‌می‌شود که مهم نیست چه فریم‌ریتی داشته باشند، در نتیجه سرعت آن‌ها همیشه یکسان است. برای مثال، فرض بگیرید که کامپیوتر ضعیفی داریم که بازی را با فریم‌ریت 30 FPS اجرا می‌کند، ولی یک کامپیوتر قوی آن را با فریم‌ریت 60 FPS اجرا می‌کند. حالا اگر بازی‌مان **مستقل از فریم‌ریت** باشد در هر دو کامپیوتر با سرعتی یکسان اجرا می‌شود (اشیاء با سرعتی یکسان در بازی حرکت می‌کنند). ولی اگر بازی‌مان «**وابسته به فریم‌ریت**» باشد، سرعتش در کامپیوترِ ضعیف‌تر نصف کامپیوتر قوی خواهد بود، یعنی بازی حالت اسلوموشن پیدا می‌کند. برای این‌که خیالتان راحت باشد که هرکسی می‌تواند در هر کامپیوتری به راحتی بازی‌تان را بازی کند و از آن لذّت ببرد، باید بازی خود را مستقل از فریم‌ریت طرّاحی کنید. اگر سرعت بازی در هنگام کم‌شدنِ فریم‌ریت کاهش یابد شدیداً روی گیم‌پلی اثر بدی می‌گذارد، و تا حدّی می‌تواند بازی‌کن را ناامید کند که او ترجیح دهد از بازی خارج شود.

در این آموزش یاد می‌گیرید که چگونه بازی‌های‌تان را مستقل از فریم‌ریت طرّاحی کنید. با این کار شما مزیّت دیگری نیز خواهید داشت و آن قابلیّت «**تناسب زمانی**» است، که توسّط آن به راحتی می‌توانید اسلوموشن عمدی ایجاد کنید یا استوپ (Pause) بسازید.

اکسپرشن سیستمی dt

کلید دست‌یابی به استقلال از فریم‌ریت، اکسپرشن سیستمی dt است. dt مخفف delta-time می‌باشد. دلتا (Δ) به معنی تغییر در یک کمیّت است، بنابراین دلتا تایم (Δt یا dt) یعنی تغییرات زمان از تیک قبلی تا حالا در واحد ثانیه (به زمان سپری شدن یک فریم در بازی تیک گفته می‌شود).

برای مثال dt در 100 fps می‌شود 0.01 (یک صدم ثانیه)، و در 10 fps می‌شود 0.1 (یک دهم ثانیه)، مقدار dt در هر تیک نسبت به تیک دیگر فرق دارد (چون فریم‌ریت همیشه ثابت نیست)، پس بعید است که در مدّتی طولانی dt ثابت باقی بماند.

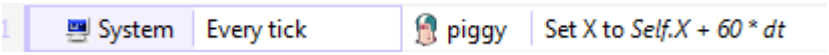
اگر یک متغیّر بسازید و تنظیم کنید در هر تیک (Every tick) به اندازه‌ی dt به آن اضافه شود، آنگاه می‌بینید که در هر ثانیه یکی به آن اضافه می‌شود، زیرا جمع تمام زمان‌های تیک‌ها در یک ثانیه همان یک ثانیه می‌شود! (می‌توانید متغیّری بسازید و در هر تیک به اندازه‌ی dt به آن بیفزایید تا یک تایمر بسازید.)

چگونه از dt استفاده کنیم

گاهی اوقات اشیاء حرکتی انجام می‌دهند که وابسته به فریم‌ریت است، مثل تصویر زیر:

در تصویر بالا، خوک‌کوچولو در هر تیک (یک‌بار در هر فریم) یک پیکسل به سمت راست حرکت می‌کند. یعنی در 30 fps با سرعت ۳۰ پیکسل بر ثانیه حرکت می‌کند و در 60 fps با سرعت ۶۰ پیکسل بر ثانیه. این سرعت‌ها نسبت به فریم‌ریت متفاوت هستند.

همان طور که در بالا گفتیم در هر ثانیه یکی به مجموع dt اضافه می‌شود، پس ایونت را به شکل زیر تغییر می‌دهیم:



حالا خوک‌کوچولو دقیقاً ۶۰ پیکسل در هر ثانیه حرکت می‌کند. زیرا dt در هر ثانیه یکی افزایش می‌یابد پس 60*dt در هر ثانیه ۶۰ تا افزایش می‌یابد (60 x dt = 60 x 1 = 60). این یعنی چه در 30 fps و چه در 60 fps شیء مورد نظرمان در هر ثانیه دقیقاً ۶۰ پیکسل حرکت می‌کند، با همان سرعت، بدون توجّه به فریم‌ریت.

در همه‌جا از dt استفاده کنید

هر وقت که می‌خواهید شیئی را با سرعتی یکنواخت حرکت دهید، باید از dt استفاده کنید تا مستقل از فریم‌ریت شوید. برای مثال، اکشن Move forward اسپرایت، تعداد پیکسل‌هایی را که قرار است به آن مقدار به جلو حرکت کند از شما می‌گیرد. حالا اگر می‌خواهید به طور ثابت به جلو حرکت کند می‌توانید تعیین کنید که dt * 60 پیکسل در هر ثانیه حرکت کند، تا با سرعت ۶۰ پیکسل بر ثانیه در همان جهتی که قرار دارد به جلو حرکت کند.

رفتارها به صورت خودکار از dt استفاده کرده‌اند

همه‌ی رفتارهای کانستراکت ۲ در محاسبات داخلی حرکتشان از dt استفاده می‌کنند. یعنی هرچیزی که با رفتارها حرکت کند مثل رفتار Platform و Direction 8 به هیچ چیز خاصی نیاز ندارد، این کار به طور خودکار انجام می‌شود.

رفتار Physics یک استثنا هست. فیزیک به طور پیش‌فرض از dt استفاده نمی‌کند، بنابراین وابسته به فریم‌ریت است. این به خاطر بی‌ثباتی dt است. این بی‌ثباتی باعث می‌شود که موقع استفاده از فیزیک، هر بار با نتیجه‌ای متفاوت روبرو شوید، حتّی اگر دقیقاً همان یک کار خاص را دو بار با فیزیک انجام دهید. این مشکل برای بازی‌های فیزیکی آزاردهنده است، به همین دلیل به طور پیش‌فرض، فیزیک وابسته به فریم‌ریت است. با این حال، شما می‌توانید با استفاده از اکشن Set Stepping Mode فیزیک در On start of layout و انتخاب Framerate independent رفتار فیزیک را هم مستقل از فریم‌ریت کنید.

تناسب زمانی (Timescaling)

یکی از خصوصیّات خیلی عالی کانستراکت ۲ «تناسب زمانی» است. این قابلیّت که تحت عنوان تایم‌اسکیل (time scale) هم شناخته می‌شود به شما اجازه می‌دهد تا سرعت زمان در بازی را تغییر دهید. این کار توسّط اکشن Set Time Scale سیستم انجام می‌شود. تناسب زمانیِ ۱ به معنی سرعت عادی است. ۰٫۵ یعنی نصف سرعت عادی ، و ۲٫۰ یعنی دوبرابر سریع‌تر. اگر تناسب زمانی بازی‌تان را روی ۰٫۱ قرار دهید،

سرعت بازی ۱۰ برابر کاهش می‌یابد امّا اشیاء در این وضعیّت نیز خیلی نرم و روان حرکت می‌کنند، یک افکت اسلوموشن عالی!

تناسب زمانی با تغییر مقدار dt کار می‌کند. یعنی مقدار dt را کمتر یا بیشتر از مقدار واقعی آن می‌کند، و در نتیجه رفتارها و نیز هر حرکتی که از dt استفاده کرده باشد تحت تأثیر آن قرار می‌گیرد و سریع‌تر یا کُندتر می‌شود. اگر برای حرکت دادن اشیاء خود از dt استفاده نکرده باشید (مثل همان اوّلین ایونتی که مثال زدیم)، این حرکت تحت تأثیر تناسب زمانی قرار نمی‌گیرد! پس اگر می‌خواهید از تناسب زمانی بهره‌مند شوید باید در تمام حرکات بازی از dt استفاده کرده باشید.

ساخت استوپ (Pause)

اگر مقدار تناسب زمانی را 0 قرار دهید، تمام حرکات بازی متوقّف می‌شود. به همین سادگی می‌توانید بازی خود را Pause کنید. بعداً دوباره مقدار آن را 1 کنید تا بازی از حالت استوپ خارج شود.

شاید ببینید هنوز می‌توان کارهایی مثل شلّیک گلوله را انجام داد، یعنی کارهایی که توسّط ورودی‌های کاربر انجام می‌شود. برای حل این مشکل می‌توانید همه‌ی ایونت‌های اصلی بازی خود را درون یک گروه (Group) قرار دهید، و هنگام Pause شدن یا ادامه یافتن بازی، آن گروه را فعّال یا غیرفعّال کنید.

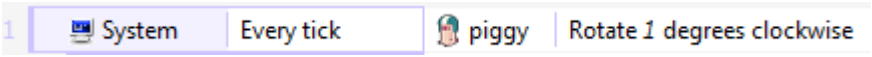
یکی دیگر از مزایای این کار، تستِ مستقل از فریم‌ریت بودن بازی است. به این صورت که اگر تناسب زمانی صفر شود همه چیز باید متوقّف شود. اگر مستقل از فریم‌ریت بودن بازی‌تان درست انجام نشده باشد می‌بینید که بعضی اشیاء همچنان به حرکت خود ادامه می‌دهند. در این صورت باید نگاه کنید که آن اشیاء چگونه حرکت می‌کنند و مطمئن شوید که در حرکت آن‌ها از dt استفاده شود.

انواع دیگر حرکت

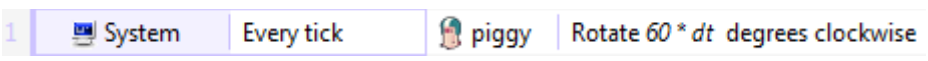
خیلی مهم است که بدانید dt باید در هر نوع حرکتی به کار برود، که شامل حرکت‌هایی از نوع چرخش و شتاب نیز می‌شود.

چرخش

مشابه مثال قبلی، ایونت زیر خوک‌کوچولوی ما را یک درجه در هر تیک می‌چرخاند.



این یعنی ۳۰ درجه بر ثانیه در 30 FPS و ۶۰ درجه بر ثانیه در 60 FPS. دوباره در فریم‌ریت‌های متفاوت سرعت‌های متفاوتی داریم. استفاده از dt به همان شکل قبلی باز هم مشکل را حل می‌کند. با این کار خوک‌کوچولو در هر ثانیه ۶۰ پیکسل حرکت می‌کند و هیچ اهمّیّتی ندارد که فریم‌ریت چه قدر باشد.



شتاب

مستقل کردن شتاب از فریم‌ریت هم نسبتاً ساده است، و معمولاً وقتی به کارتان می‌آید که دارید حرکتی دلخواه را با استفاده از ایونت‌ها پیاده‌سازی می‌کنید.

فرض کنید شیء مورد نظر ما متغیّری به نام Speed دارد، حالا این شیء باید با سرعت Object.Speed * dt پیکسل بر تیک حرکت کند. بنابراین در اینجا در نظر گرفتیم واحدِ متغیّرِ Speed پیکسل بر ثانیه است (که با ضرب شدن در dt شده است پیکسل بر تیک).

حالا فرض کنید میخواهیم شیءمان را شتاب دهیم تا در هر ثانیه ۱۰۰ پیکسل بر ثانیه به سرعتش اضافه شود. در این صورت شما باید در هر تیک، به متغیّر Speed به اندازه‌ی 100 * dt اضافه کنید. حالا شیءمان به صورت مستقل از فریم‌ریت حرکت می‌کند. به عبارتی باید برای تنظیم مکان شیء و همچنین برای تنظیم سرعت شیء، در هر دو از dt استفاده کنیم.

اشتباهات رایج

هرگز در ایونت Every X seconds از dt استفاده نکنید! ایونتی مثل Every 1 second در هر ثانیه یک‌بار اجرا خواهد شد و کاری ندارد که فریم‌ریت چه قدر است، یعنی این ایونت به خودی خود مستقل از فریم‌ریت است. این ایونت بر اساس زمان کار می‌کند، نه بر اساس فریم‌ها. اتّفاقاً اگر ایونتی مثل Every 60 * dt seconds بنویسید آن را وابسته به فریم‌ریت کرده‌اید، دقیقاً برعکس آن چیزی که میخواستید! چنین ایونتی در 10 FPS (یعنی dt = 0.1) هر ۶ ثانیه یکبار اجرا می‌شود، و در 100 FPS (یعنی dt = 0.01) هر ۰٫۶ ثانیه یکبار اجرا می‌شود؛ ولی اگر فقط می‌نوشتید Every 6 seconds هر ۶ ثانیه یکبار اجرا می‌شد و هیچ اهمّیّتی نداشت که فریم‌ریت چه قدر است.

نکات حرفه‌ای

فریم‌ریت مینیمم

در فریم‌ریت‌های خیلی پایین dt خیلی بزرگ می‌شود. مثلاً در 5 FPS مقدار dt می‌شود 0.2. بنابراین شیئی که داشت با سرعت ۵۰۰ پیکسل بر ثانیه حرکت می‌کرد، حالا ۱۰۰ پیکسل بر تیک حرکت می‌کند. این قضیه باعث می‌شود که شیءمان یکهویی غیب شود و جای دیگری ظاهر شود یا حتّی از درون دیوار و سایر موانع عبور کند و برخوردش با آن اشیاء نادیده گرفته شود.

معمولاً در چنین فریم‌ریت‌های پایینی عملاً نمی‌شود بازی کرد، ولی با اوضاع ناپایداری مثل این، وضعیّت خیلی بدتر هم می‌شود. برای اینکه حتّی در فریم‌ریت‌های خیلی پایین هم بازی‌مان قابل اتّکا باشد کانستراکت اجازه نمی‌دهد مقدار dt از 0.1 کمتر شود. به عبارتی اگر فریم‌ریت کمتر از 10 FPS باشد، مقدار dt قطعاً 0.1 است. این قضیه یک معنای دیگر هم دارد و آن این است که در فریم‌ریت‌های کمتر از 10 FPS سرعت بازی کند می‌شود و به حالت اسلوموشن در می‌آید (که قبلاً این را به عنوان یکی از ایرادات بازی‌های وابسته به فریم‌ریت معرّفی کرده بودیم)، با این حال معمولاً کُندشدن بازی بهتر است از مشکل «یکهویی غیب و ظاهر شدن اشیاء».

بی‌ثباتی

همان طور که قبلاً در مورد فیزیک گفته بودیم، dt معمولاً تغییرات تصادفی کوچکی دارد. این تغییرات معمولاً به خاطر تایمرهای نه‌کاملاً دقیق کامپیوترهاست. این موضوع می‌تواند موجب بی‌ثباتی‌هایی در بازی شود. با این حال معمولاً این بی‌ثباتی‌ها ناچیز و بی‌اهمّیّت است و آن طور مثل رفتار فیزیک خودش را نشان نمی‌دهد. برای همین توصیه می‌کنیم که همیشه در بازی‌هایتان از dt استفاده کنید، مگر اینکه واقعاً دقّت خیلی زیادی لازم باشد (که تقریباً هیچگاه چنین دقّتی را لازم نداریم).

تناسب زمانی شیء

شما به اشیاء مختلف به صورت مجزّا می‌توانید تناسب زمانی متفاوتی بدهید، این کار توسّط اکشن Set object time scale سیستم انجام می‌شود. بدین وسیله شما می‌توانید مثلاً تناسب زمانی بازی‌تان را روی 0.3 قرار دهید تا در حالت اسلوموشن قرار بگیرد، ولی تناسب زمانی پلیر خود را روی 1 قرار دهید تا با سرعت کامل و معمولی خود حرکت کند. برای انجام واقعی این کار ابتدا توسّط اکشن Set Time Scale تناسب زمانی کل بازی را روی 0.3 تنظیم کنید و سپس توسّط اکشن Set object time scale تناسب زمانی پلیر خود را روی 1 تنظیم کنید.

اکسپرشن سیستمی dt فقط تحت تأثیر تناسب زمانی بازی است (نه تناسب زمانی اشیاء به صورت مجزّا). هر یک از اشیاء، dt خودش را دارد (مثلا Player.dt) که برای حرکت‌های مربوط به این شیء باید از این اکسپرشن استفاده شود، نه اینکه از اکسپرشن سیستمی dt استفاده کنیم و همین طور خالی بنویسیم dt. لذا الآن دو مقدار برای dt وجود دارد: یکی برای خود بازی، و یکی برای پلیر. چون این مقادیر متفاوت هستند، قسمت‌های مختلف بازی می‌توانند با سرعت‌های مختلفی به پیش روند.

در این مثال، برای برگرداندن پلیر به زمان اصلی بازی، از اکشن Restore object time scale سیستم استفاده کنید.

جمع‌بندی

خیلی مهم است که **از همان ابتدای کار** از dt استفاده کنید. این کار باعث می‌شود گیم‌پلی بازی بهتر شود، و تضمین می‌کند سرعت بازی همیشه یکنواخت باشد و مثلاً در قسمت‌هایی از بازی که بار پردازشی زیاد است بازی‌تان کُند نشود. از مهمترین مزایای دیگر این کار می‌توان به قابلیّت «تناسب زمانی»، پیاده‌سازی راحت Pause، و حتّی کنترل تناسب زمانی اشیاء به صورت مجزّا اشاره کرد.

فراموش نکنید که در پیاده‌سازی رفتارهای خود کانستراکت از قبل dt به کار رفته است (غیر از رفتار Physics که در صورت نیاز می‌توانید این قابلیّتش را فعّال کنید). اگر در بازی‌تان برای حرکت اشیاء فقط از رفتارها استفاده کرده‌اید و در سیستم ایونت حرکتی را پیاده‌سازی نکرده‌اید اصلاً لازم نیست نگران dt باشید! ولی در بیشتر بازی‌ها حرکت‌هایی وجود دارند که توسّط ایونت‌ها کنترل می‌شوند، و در این موارد خیلی مهم است که حواستان باشد که بازی‌تان مستقل از فریم‌ریت باشد.