

برنامه‌نویسی پیشرفته
زمستان 1402 - بهار 1403
فاز دوم پروژه

تیم تعریف و طراحی پروژه:
آرین همتی، سینا دانشگر

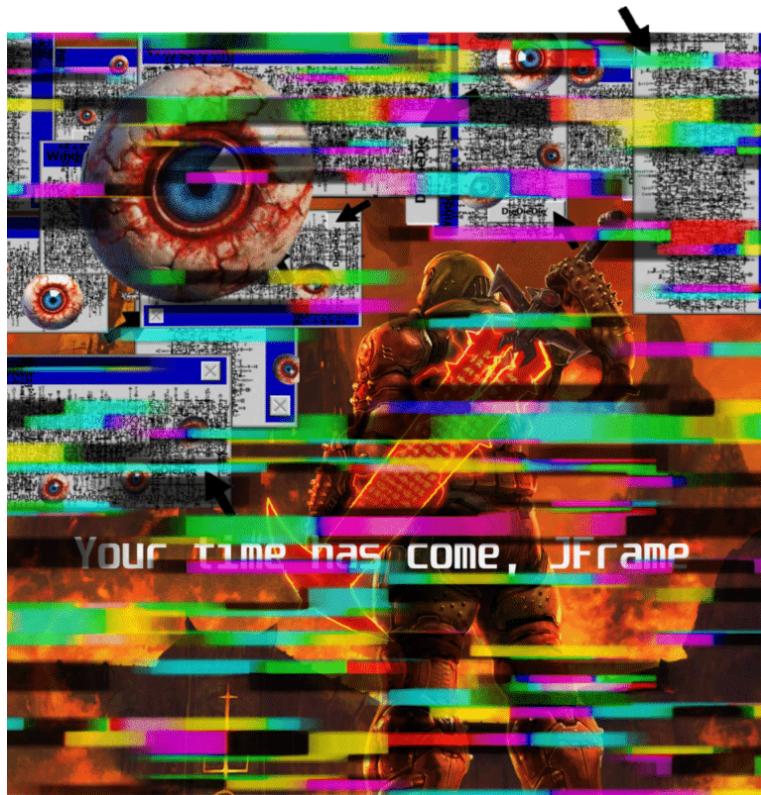
آیدی‌ها روی اسامی لینک شده‌اند و کافیست روی آنها کلیک کنید



مقدمه‌ای بر فاز دوم (حتماً بخوانید!)

به فاز دوم پرتوه درس برنامه‌نویسی پیشرفته خوش آمدید. 🎉 در فاز اول پرتوه شما یک مدل مینیمال از بازی‌ای که قرار است در طول این ترم پیاده‌سازی کنید را کامل کردید. در این فاز شما با پیاده‌سازی شالودهٔ فیزیک بازی از جمله حرکت شتاب دار (برای انتیتی‌ها و همچنین پنجره‌ها) و برخورد، و یک مدل کلی از قابلیت‌ها و Skill Tree، مقدمات پیاده‌سازی اجزای اصلی بازی در فاز جاری را به پایان رسانید. هدف اصلی این فاز، تکمیل پیاده‌سازی لاجیک بازی و مدل‌های مورد نیاز برای پیاده‌سازی بازی است و تمرکز کمتری بر گرافیک بازی وجود خواهد داشت. اهداف آموزشی اساسی این فاز شامل کلین‌کدینگ، استفاده صحیح از معماري و دیزاین‌پترن‌ها و همچنین پیروی از اصول بنیادی SOLID است. مشابه فاز قبل، اکیداً توصیه می‌شود همچنان رعایت مدل MVC را در سرتاسر کد خود رعایت کنید و اگر منوجه اشکالی در رابطه با این اصل در کدهای فاز قبلی شدید، آن را حتماً قبل از ادامه پیاده‌سازی برطرف کنید. پیاده‌سازی اصولی این دو فاز، در راستای **SOLID** و **MVC** کار شما در فاز سوم پرتوه را بسیار ساده‌تر خواهد کرد. برای اینکه شهود درستی از فلسفه رعایت مدل MVC داشته باشد توصیه می‌شود **این کارگاه** را ببینید. همچنین توصیه می‌شود قبل از شروع این فاز، مقداری درباره معماري‌های Event-Driven مطالعه داشته باشید. به این منظور مطالعه **این منبع** و همچنین مطالعه درس **Concurrency In Swing** توصیه می‌شود. در این لایه‌های بسیاری از لاجیک و گرافیک کد شما روی هم اضافه می‌شوند و لذا چالش اصلی شما در این فاز، پیاده‌سازی اصولی این لایه‌ها خواهد بود. شما با رعایت این اصول و انتخاب معماري‌های مناسب، می‌توانید در عین راحتی کد زدن و جلوگیری از وقوع مشکلات مکرر و زمان بر در میانه پرتوه، توسعه‌پذیری و تغییرپذیری آن را نیز حفظ کنید و در فاز سوم پرتوه نیز زمان کمتری برای رفع مشکلات بالقوه در معماري لایه‌های کد خود صرف کنید. این در حالیست که رعایت نکردن این اصول می‌تواند منجر به ایجاد یک کد پیچیده و غیرقابل فهم و غیرقابل توسعه شود که مشکلات فراوانی در توسعه فاز سوم پرتوه (شبکه، مولتی‌پلیر و ...) ایجاد خواهد کرد. فلذا اکیداً توصیه می‌شود قبل از دست به کد شدن، چندین بار (👉) داک را به طور کامل بخوانید و زمان کافی صرف کنید تا معماري‌ای قابل اطمینان و نسبتاً دقیق از لایه‌های کد خود طراحی کنید و بتوانید به راحتی هر قسمت از کد را تکمیل کنید. منابع تکمیلی برای مطالعه بیشتر درباره اصول SOLID و همچنین دیگر مباحثت مورد بحث در کلین‌کدینگ در مقدمه داک فاز اول آمده است. شما برای پیاده‌سازی این فاز به دانش کافی درباره طراحی، مولتی‌تریدینگ، ورک‌فلوی EDT، AWT و به خصوص **Design Pattern** ها و معماري نیاز خواهید داشت. در این فاز از شما انتظار می‌رود با رعایت اصول کلین‌کدینگ و معماري‌های مناسب، کمی توسعه‌پذیر پیاده‌سازی کنید و از coupling در ساختار کد خود دوری کنید.

نکته بسیار مهم: برخی موارد تعریف شده در داک فاز قبل (مانند دکمه‌های بازی، مکانیک‌های حرکت، برخورد، Impact و شلیک) برای پیاده‌سازی اجزای مختلف این فاز ضروری هستند و در نمره این فاز به طور مستقیم تاثیر دارند. در صورت عدم پیاده‌سازی آنها در فاز اول، طبیعتاً نمره بخش‌هایی که برای فعالیت به این موارد وابسته‌اند را دریافت نخواهید کرد. موارد مربوط به کلین‌کدینگ و رعایت درست معماري‌ها و دیزاین پترن‌ها در حدود 20 درصد نمره این فاز را تشکیل می‌دهد. رعایت بخش‌هایی اعم از **لين‌بخش** و **حرکت شتاب دار** و **گیت** که رعایت آنها در تمام فاز اول حائز نمره است، در این فاز و فاز بعدی نیز حائز نمره مجزا خواهند بود. بخش‌های دیگر فاز اول پرتوه در بارم‌بندی این فاز لاحظ نخواهند شد.



| | |
|---------|---|
| 2..... | مقدمه‌ای بر فاز دوم (حتماً بخوانید)..... |
| 4..... | محیط بازی (Game Env.) |
| 4..... | فاز جدید، HUI جدید! |
| 4..... | اپسیلون کجاست؟..... |
| 4..... | بیش از یک ترجان، بیش از یک فریم! |
| 4..... | مرگ، یک Negative Feedback Loop |
| 4..... | ریسک پیشروی (PR) |
| 4..... | چکپوینت‌ها |
| 5..... | بروتکل‌های رفتاری مربوط به فریم‌ها |
| 5..... | پروتکل نمایش فریم‌ها |
| 5..... | Layered Painting |
| 6..... | پروتکل‌های رفتاری فریم‌ها |
| 6..... | فریم موضعی |
| 7..... | پروتکل بنیادی رفتار فریم‌های موضعی. |
| 7..... | (Enemy) دشمن‌ها |
| 7..... | مکانیزم Global Routing |
| 7..... | حملات AoE |
| 7..... | Ranged Attack (RA) |
| 8..... | Normal Enemy Types |
| 9..... | Mini-Boss Enemy Types |
| 9..... | Final Boss |
| 9..... | انکهای باس |
| 10..... | بازگشت به سنت‌ها |
| 10..... | انکهای جدید |
| 10..... | مکانیک‌های باس. |
| 10..... | صحنه پایانی بازی |
| 11..... | ذخیره بازی. |
| 11..... | گریزی مجدد به SOLID |
| 12..... | Skill Tree v2.0 |
| 12..... | فروشگاه جدید |
| 13..... | Spawn |
| 13..... | مکانیک‌های Game Design |
| 14..... | باز هم بخش امتیازی! |
| 14..... | . Omenoct Wallrun |
| 14..... | . Necropick Alert |
| 14..... | . Archmire AoE Fade |
| 14..... | . Archmire Mitosis |
| 14..... | . Black Orb Avalanche |
| 14..... | . Annihilator |
| 14..... | . Save Validation |
| 14..... | . Save Synchronization |
| 14..... | . Writ of Athena |
| 14..... | . Log |
| 14..... | . Wyrm Horde |
| 14..... | استفاده از روش‌های مبتنی بر PCA |

تقریباً تمام متن‌های رنگی در سراسر این داک یا جملات و عبارات مهم در فرآیند پیاده‌سازی هستند و یا با کلیک روی آنها می‌توانید

به بخش‌هایی از داک این فاز یا فاز گذشته و یا لینک منابع مفید منتقل شوید فلذًا حتماً به این قسمت‌ها توجه ویژه داشته باشید.

محیط بازی (Game Env.)

فاز جدید، HUI جدید!

در این فاز از شما انتظار می‌رود علاوه بر XP, HP, Wave, Elapsed Time, قابلیت‌های در حال اثر و قابلیت فعال Epsilon را نیز در صفحه بازی به نمایش بگذارید. همچنین علامت مربوط به قابلیت فعال، باید در صورت فعال شدن این تغییر را برای بازیکن به نمایش بگذارد.

اپسیلون کجاست؟...

در صحنه پایانی فاز قبل اپسیلون کل فریم را پر می‌کند تا کدبیس را از شر تروجان موزی نجات دهد اما دیگر کار از کار گذشته است... تروجان کنترل کامل سیستم را به دست گرفته و خود را در همه جای root directory شما کپی کرده است! صفحه بازی بسته می‌شود و بلافاصله دقیقا مشابه **توضیحات فاز قبل** فریم جدیدی روی صفحه باز می‌شود. اما این بار، فقط یک تروجان در مقابل شما نیست...

بیش از یک تروجان، بیش از یک فریم!

در این فاز، هر یک از تروجان‌ها مستقل‌فرمی را اجرا و کنترل می‌کنند و در نتیجه شما در طول این بازی بعضی اوقات با چندین و چند فریم از دشمن‌های مختلف سر و کله خواهید زد... رفتارها و مکانیک‌های مربوط به فریم‌های مختلف در ادامه داک توضیح داده شده است. انتخاب معماری مناسب و درک مناسب از محیط گرافیکی جاوا برای هندل کردن صحیح و بدون لگ این فریم‌ها فرایند پیاده‌سازی را ساده‌تر خواهد کرد.

مرگ، یک Negative Feedback Loop

میزان درگیری کاربر در هر بازی به عواملی مختلفی وابسته است که در مطالعه Game Psychology به دو دسته کلی NFL, PFL تقسیم می‌شوند و شدت اثر هر یک، رفتار بازیکنان را در بازی تغییر می‌دهد. در این بازی شما چند مورد از این دست را پیاده‌سازی خواهید کرد که اولین آنها مکانیک آغاز تا پایان آن خواهد بود) نرخ پیشروی کلی (total progress rate) را برابر با مجموع نرخ پیشروی تمام Wave‌های بازی تعریف می‌کنیم. نهایتاً "ریسک پیشروی" در هر لحظه را برابر با $\frac{XP \times Total\ progress\ rate}{HP} \times 10^1$ تعریف می‌کنیم که در آن XP، مقدار HP کسب شده در بازی فعلی است.

ریسک پیشروی (PR)

در هر لحظه از بازی، نرخ پیشروی (progress rate) در یک Wave خاص را ($Wave\ Number \times Elapsed\ Wave\ Time$) تعريف می‌کنیم که در آن Elapsed Wave Time زمان صرف شده از ابتدای آغاز همان Wave است. (در صورتی که Wave مذکور پایان یافته باشد، این مقدار برابر با فاصله آغاز تا پایان آن خواهد بود) نرخ پیشروی کلی (total progress rate) را برابر با $\frac{XP \times Total\ progress\ rate}{HP}$ در آن قرار دارد باز می‌شود. در صورتی که "ریسک پیشروی" در هر لحظه را برابر با $\frac{XP \times Total\ progress\ rate}{HP} \times 10^1$ تعریف می‌کنیم که در آن XP، مقدار HP کسب شده در بازی فعلی است.

چکپوینت‌ها

در این بازی مثل بسیاری از بازی‌های دیگر، بازیکن می‌تواند پیشروی خود را در نقاطی از بازی ذخیره کند. شما باید بازی را در انتهای هر Wave برای بازیکن ذخیره کنید. (به انضمام تمام مشخصات مربوط به وضعیت بازی) برای این کار در روند بازی پورتال‌های چکپوینت طراحی می‌کنیم. در برخی از Wave‌های طراحی شده در این فاز، پس از گذشت مدتی از آغاز Wave یک پورتال در نقطه‌ای از فریمی که Epsilon در آن قرار دارد باز می‌شود. در صورتی که Epsilon وارد این پورتال شود، بازی موقتاً متوقف شده و یک پنل برای کاربر باز می‌شود و دو گزینه را پیش روی او قرار می‌دهد:

- کاربر می‌تواند با پرداخت PR تا از XP هایش، وضعیت بازی خود را در این نقطه سیو کند. در این صورت پنل بسته شده و پورتال از بین می‌رود. در نتیجه این انتخاب، 10 واحد به HP کاربر اضافه می‌شود و همچنین تنها در اولین باری که می‌میرد، با 10 واحد HP از نقطه سیو شده، دقیقا در همان شرایط وضعیت که بازی را در این پورتال سیو کرده بود به بازی ادامه می‌دهد. در صورتی که Epsilon قبل از مردن، بار دیگر بازی را در یک پورتال چکپوینت سیو کند، اثر چکپوینت قبلی او از بین خواهد رفت.
- بازیکن می‌تواند با رد کردن این پیشنهاد، به اندازه 10 درصد PR تا XP دریافت کند. در این صورت پنل بسته شده و دروازه چکپوینت نابود می‌شود و تا زمانی که بازیکن در نقطه‌ای بازی را سیو نکرده باشد، مرگ او باعث شروع مجدد بازی می‌شود.

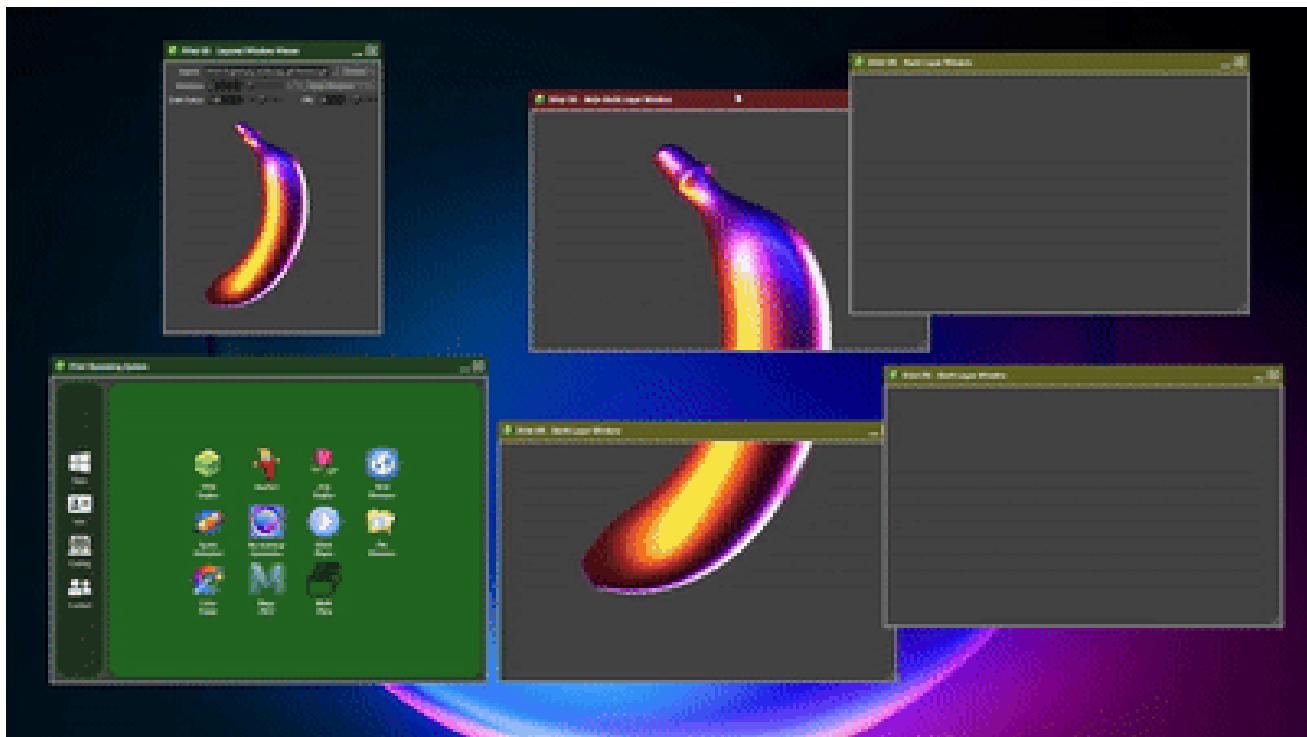
توضیح کامل فرایند سیو بازی در ادامه داک آمده است.

پروتکل‌های رفتاری مربوط به فریم‌ها

یکی از چالش‌های اصلی شما در این فاز، پیاده‌سازی یک محیط گرافیکی مناسب برای فریم‌هاست. از آنجا که در این فاز برخلاف فاز قبل، برخی انتیتی‌ها یا فریم‌ها ممکن است با یکدیگر هم‌پوشانی داشته باشند، چالش‌های شما در این فاز شما اعم از نمایش هم‌پوشانی، اولویت paint برای انتیتی‌های دارای هم‌پوشانی و همچنین تعریف رفتارهای فریم‌های بازی اعم از **shrinkage** و **حکمت** خواهند بود که در ادامه توضیح داده می‌شوند.

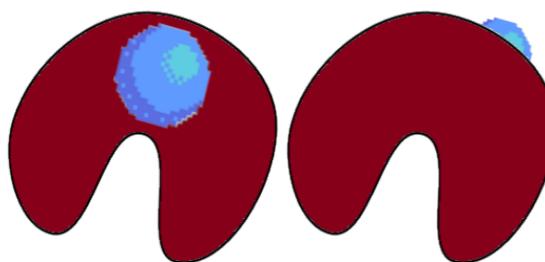
پروتکل نمایش فریم‌ها

در این بازی فریم‌های شما یک پروتکل سراسری را در قبال محتوایی که به نمایش می‌گذارند دنبال می‌کنند. به این معنا که اگر یک انتیتی در موقعیتی قرار داشته باشد که توسط چندین فریم پوشیده شده است، همه این فریم‌ها باید این انتیتی را به طور همزمان نمایش دهند. به بیان دیگر، محتوای نمایش داده شده توسط فریم‌های مختلف باید از لحاظ موقعیت مکانی با هم همخوانی داشته باشد. به این پروتکل نمایش، گفته می‌شود. مثال از این پروتکل نمایش در ویدیوی زیر آمده است:



Layered Painting

همانطور که در توضیحات این بخش اشاره شد، در این فاز برخلاف فاز قبل، برخی انتیتی‌ها و یا فریم‌ها ممکن است شامل هم‌پوشانی باشند. در این صورت یکی از مسائل مورد اهمیت برای نمایش اجزای مختلف بازی، ترتیب پینت کردن این عناصر گرافیکی است. به تصویر زیر توجه کنید:



در شکل سمت چپ ابتدا محیط زمینه و سپس انتیتی روی آن رسم شده است، اما در شکل سمت راست ابتدا انتیتی رسم شده و سپس محیط زمینه روی آن ترسیم شده است. طبیعتاً شکل سمت چپ برای بازیکن مطلوب است.

لذا شما باید در فرایند پینت عناصر خود ترتیب صحیحی برای پینت کردن این عناصر طراحی کنید و از آن در ساختار repaint خود بهره ببرید.

پروتکل‌های رفتاری فریم‌ها

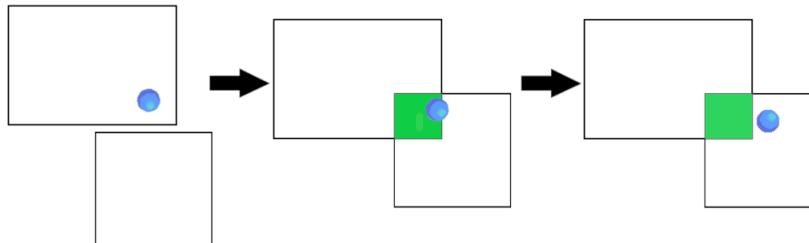
همانطور که توضیح داده شد، فریم‌های بازی در طول این فاز با یکدیگر همپوشانی خواهند داشت و شما باید عملکرد آنها در قبال عملیات‌های حرکت فریم و shrinkage را پیاده‌سازی کنید. در این فاز فریم‌ها طبق پیروی یا عدم پیروی از دو خاصیت اساسی زیر، پروتکل‌های رفتاری متفاوتی را دنبال خواهند کرد. در ادامه این دو خاصیت را شرح خواهیم داد:

- **خاصیت ایزومتری:** فریم‌هایی که خاصیت ایزومتری دارند تحت اثر **مکانیک‌های شلیک** تغییر اندازه نمی‌دهند. این فریم‌ها همچنین از پروتکل **shrinkage** (که در فاز گذشته تعریف شد) نیز پیروی نخواهند کرد. ابعاد این فریم‌ها تا زمانی که از بین بروند ثابت خواهد بود. دقت کنید فریم‌های این دسته با اینکه دچار تغییر در شکل فریم نمی‌شوند، اما از قابلیت جابجایی برخوردارند.
- **خاصیت صلب:** فریم‌هایی که از این خاصیت پیروی نمی‌کنند، می‌توانند آزادانه با تمام فریم‌های دیگر بازی همپوشانی داشته باشند. در صورتی که دو فریم که خاصیت صلب دارند به یکدیگر برخورد کنند، حرکت آنها متوقف می‌شود و قادر به همپوشانی با هم نخواهند بود.

برای مثال همانطور که در فاز قبل توضیح داده شد، فریم اولیه بازی از خاصیت ایزومتری پیروی نمی‌کند. دقت کنید همانطور که در ادامه توضیح داده خواهد شد، فریم‌ها می‌توانند در حین بازی دچار تغییراتی در این خاصیت‌ها شوند و این خاصیت‌ها برای هر فریم لزوماً ثابت نخواهند بود. به طور مشابه **انتیتی‌های صلب** را تعریف می‌کنیم: هر انتیتی که نتواند از دیواره‌های فریم‌ها عبور کند (مشابه Epsilon) را صلب می‌نامیم.

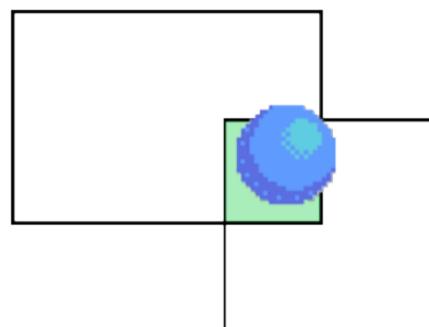
تمام انتیتی‌های دشمن به صورت پیش‌فرض (**مگر اینکه توضیح در این باره داده شود**) انتیتی‌های غیر صلب هستند.

توجه کنید فریم‌هایی که خاصیت ایزومتری ندارند، در حین بازی دائمًا تحت اثر **مکانیزم‌های shrinkage** و **مکانیک‌های شلیک** قرار خواهند داشت. چالش پیاده‌سازی در اینجاست که فریم‌های بازی به عنوان موجودات مستقل عمل نمی‌کنند! در واقع اگر دو یا چند فریم همپوشانی داشته باشند، دیواره ناحیه همپوشانی آنها موقتاً از بین می‌رود. بنابراین همه انتیتی‌ها (اعم از انتیتی‌های صلب) می‌توانند از ناحیه همپوشانی عبور کرده و بین این فریم‌ها جابجا شوند و دیواره فریم‌ها در این ناحیه جلوی حرکت انتیتی‌ها را نخواهند گرفت! مثالی از این پروتکل در تصویر زیر آمده است.



فریم موضعی

برای هر انتیتی X موجود در بازی، در هر لحظه از بازی، فریمی تحت عنوان فریم موضعی X تعیین می‌شود. این فریم در واقع فریمی است که این انتیتی به طور کامل در آن قرار دارد. برای مثال در زمان آغاز بازی، فریم موضعی Epsilon در واقع همان فریم اولیه بازی است که ظاهر می‌شود. همانطور که به نظر می‌رسد، فریم موضعی X در طول جابجایی این انتیتی ممکن است تغییر کند. دقت کنید ممکن است فریم موضعی X در هر لحظه null باشد. (X در هیچ یک از فریم‌های بازی به طور کامل قرار نداشته باشد) **همچنین توجه کنید تا وقتی X به طور کامل در فریم موضعی‌اش قرار داشته باشد، فریم موضعی‌اش تغییر نمی‌کند.** (حتی اگر به طور همزمان، کاملاً در فریم دیگری هم قرار گرفته باشد)

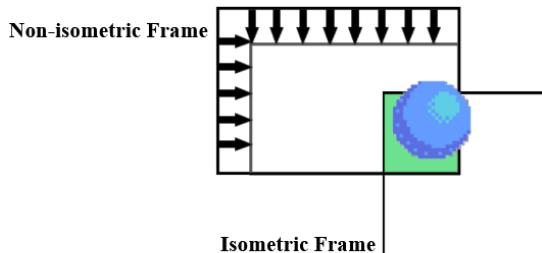


نمونه‌ای از وضعیتی که در آن Epsilon در هیچ یک از دو فریمی که می‌بینید به طور کامل قرار ندارد. فریم موضعی Epsilon در این شرایط null خواهد بود.

نکته مهم: مشابه **این بند**، هر انتیتی صلب باید در طول shrinkage فریم موضعی‌اش، همواره در درون این فریم قرار داشته باشد.

پروتکل بنیادی رفتار فریم‌های موضعی

اگر فریم موضعی یک انتیتی صلب null باشد، این انتیتی باید در ناحیه همپوشانی دو یا چند فریم قرار داشته باشد. در این صورت هیچ یک از این فریم‌ها نباید از سمت یال‌هایی که مجاور ناحیه همپوشانی هستند، کوچک شوند. علاوه بر این، هرگونه جابجایی این دو فریم نیز تا زمانی که این انتیتی از شرایط مذکور خارج شود، متوقف خواهد شد. این پروتکل از گیر کردن انتیتی‌ها در ناحیه مشترک بین دو فریم جلوگیری خواهد کرد.



توجه کنید تنها انتیتی‌های تحت اثر این پروتکل در بازی، **Epsilon** و تیرهای شلیک شده هستند. در نتیجه پیاده‌سازی این پروتکل پیچیدگی محاسباتی خاصی نخواهد داشت. قسمت‌هایی از دیوارهای فریم که روی محیط ناحیه همپوشانی نیستند رفتار عادی خود را حفظ می‌کنند.

دشمن‌ها (Enemy)

در این فاز علاوه بر دشمن‌هایی که در فاز قبل معرفی شد، تعدادی دشمن جدید نیز به بازی اضافه خواهد شد که در سه دسته اصلی دشمن‌های عادی، مینی‌باس‌ها و باس طبقه‌بندی می‌شوند. هر یک از این انتیتی‌ها همچنین پروتکل مسیریابی و Aggression مختص خود را دارند که در ادامه آنها را توضیح خواهیم داد. هر یک از آنها یک یا چند مکانیک حمله دارند که در ادامه به تفصیل (!) توضیح داده خواهند شد.

Global Routing مکانیزم

همانطور که در فاز قبل مشاهده کردید، این مکانیزم به حرکت بدون اشکال انتیتی‌هایی که نباید با یکدیگر همپوشانی داشته باشند کمک می‌کند. از آنجا که گریز از ایجاد همپوشانی به وضعیت موضعی انتیتی‌های اطراف هر انتیتی وابسته است، طبیعتاً به یک فرایند موضعی برای حل این مشکل نیازمندیم که در تعریف **Local Routing** آن را معرفی کردیم. اما از آنجا که برخی انتیتی‌های این فاز می‌توانند بدون توجه به همپوشانی، روی صفحه حرکت کنند، نیازمند یک رویه مسیریابی سراسری برای این دسته از انتیتی‌ها هستیم. همانطور که در بخش قبل توضیح دادیم، چون تیرهای از دسته انتیتی‌های صلب هستند، مسیر حرکتی که می‌توانند طی کنند وابسته به فریم‌های صفحه و موقعیت آنهاست. اما انتیتی‌هایی که از مکانیزم Global Routing پیروی می‌کنند مستقل از تمام اجزای دیگر بازی به حرکت خود در مسیری از پیش تعیین شده ادامه می‌دهند. تمام انتیتی‌های غیر صلب در این بازی (مگر اینکه توضیحات بیشتری داده شود) برای حرکت از مکانیزم Global Routing استفاده خواهند کرد

AoE حملات

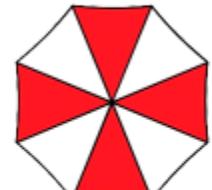
در این فاز، برخی از دشمنان شما قادر به حملاتی از نوع AoE هستند. حملات AoE اصطلاحی است که در طراحی بازی‌ها، برای توصیف حملاتی به کار می‌رود که بر یک منطقه (مساحت) مشخص، تاثیر می‌گذارند. اگر هر یک از انتیتی‌های بازی به طور کامل وارد این ناحیه شود، هر ثانیه مقدار مشخصی (مثلاً 2 واحد) آسیب متحمل می‌شود و این آسیب تا زمانی که از آن ناحیه خارج شود، ادامه خواهد داشت. هرگدام از ناحیه‌هایی که برای حمله AoE مشخص می‌شوند، به مدت معینی (مثلاً 5 ثانیه) پایدار خواهند بود و پس از آن اثر خود را از دست خواهند داد.

Ranged Attack (RA)

در فاز گذشته با دسته‌ای از حملات دشمن‌ها به نام **Melee Attack** آشنا شدید که همانطور که گفته شد، تمام دشمن‌ها، مگر در حالتی که ذکر شود، قادر به انجام آن خواهند بود. توضیحات این حمله در داک فاز قبل آمده است. در این فاز اما برخی از دشمن‌های شما همچنین قادر به انجام حملاتی به نام Ranged Attack یا حملات دوربرد نیز هستند. این حملات به دو دسته Laser, Projectile تقسیم می‌شوند. نحوه کار حملات Projectile دقیقاً مشابه تیر زدن Epsilon خواهد بود، با این تفاوت که برخلاف تیرهای Epsilon که به عنوان انتیتی صلب عمل می‌کردند، این پرتابه‌ها ممکن است انتیتی صلب نبوده و با مکانیزم Global Routing حرکت کنند و در نتیجه بتوانند از دیواره فریم‌ها عبور کنند تا زمانی که به Epsilon برخورد کنند یا از صفحه خارج شوند. (صلب بودن یا نبودن تیرها ذکر خواهد شد) توضیحات Laser نیز در ادامه داک ذکر خواهد شد.

- انتیتی‌هایی که می‌توانند با تمام انتیتی‌ها همپوشانی داشته باشند (و تحت اثر **Impact** و یا **برخورد** قرار نگیرند) را **Hovering** می‌نامیم.
- توجه داشته باشید مطابق تعریف **مکانیک‌های شلیک** در فاز قبل، تیرهای صلب در اولین برخورد با یک انتیتی (اعم از دیواره‌های فریم، کاراکترهای دیگر، دشمن‌ها، باس‌ها و یا حتی تیرهای دیگر) از بین می‌روند. (تیرهای غیر صلب طبق تعریف ارائه شده در برخورد با دیواره‌های فریم مستثنی هستند) در نتیجه در صورت برخورد تیرهای Hovering با یک انتیتی Epsilon، انتیتی متحمل آسیب می‌شود.
- به انتیتی‌هایی که قابلیت جابجایی ندارند انتیتی **ثقل** می‌گوییم. این انتیتی‌ها تحت اثر **Impact** و یا **برخورد** نیز جابجا نخواهند شد. وقتی کنید اگر یک انتیتی غیر ثقل به انتیتی ثقل برخورد کند، خودش می‌تواند تحت اثر **Impact** قرار بگیرد.

Normal Enemy Types

| Aggression Mechanics/Behaviour | Collectibles (Number/XP per each) | Damage Per Hit | HP | Character name |
|--|--|-----------------------|----|---|
| <p>- یک دشمن Non-Hovering و است که در ابتدا به سمت یک دیواره داخلی فریم موضعی Epsilon حرکت می‌کند و به آن می‌چسبد و شروع به شلیک تیرهای غیر صلب به سمت Omenocet می‌کند. در صورت تغییر فریم موضعی Epsilon، از دیواره جدا شده و مجدداً حرکت را آغاز می‌کند.</p> <p>- Omenocet می‌تواند با استفاده از شلیک تیر، حملات Melee و یا شلیک به دیواری که به آن چسبیده است، به او آسیب بزند.</p> <p>- امتحانی: با جابجایی Epsilon در فریم موضعی اش، Omenocet هم با حرکت روی دیواره داخلی این فریم، Epsilon را تعقیب می‌کند. سرعت این حرکت از سرعت عادی Omenocet بیشتر است.</p> | 8 Collectibles 4 XP per each | Melee: 8 Ranged: 4 | 20 |  Omenocet |
| <p>- هر 8 ثانیه یک بار از صفحه نمایش ناپدید شده و به زیر زمین می‌رود و تبدیل به یک انتیتی Hovering و غیر ثقل می‌شود. سپس پس از 4 ثانیه در شعاع معینی از Epsilon از زیر زمین خارج می‌شود و تعدادی تیر صلب (مثلاً 8 تا) در جهات مختلف شلیک می‌کند. (در این زمان مجدداً روی صفحه نمایش ظاهر می‌شود و در این مدت ثقل و Ranged خواهد بود)</p> <p>- این دشمن در زمان حضور در صفحه نمایش همواره ثقل است و Epsilon می‌تواند با استفاده از شلیک تیر و حملات Melee به او آسیب بزند. این دشمن از حملات Necropick بهره نمی‌برد.</p> <p>- امتحانی: شما باید لحظاتی قبل از ظاهر شدن مجدد Necropick روی صفحه، محل بیرون آمدن آن را روی صفحه نمایش با علامتی دلخواه نمایش دهید.</p> | 4 Collectibles 2 XP per each | Ranged: 5 | 10 |  Necropick |
| <p>- یک دشمن تماماً Hovering است که قادر به حمله Melee نیست و صرفاً از حملات AoE بهره می‌برد. اگر هر انتیتی دیگری کاملاً درون این دشمن قرار بگیرد، هر ثانیه 10 واحد از HP اش کاسته خواهد شد. به این حمله Drown می‌گوییم. همچنین این کاراکتر در تمام مسیر عبورش حملات AoE را فعل می‌کند. (به این معنا که ناحیه‌ای که توسط Archmire پوشیده شده است، در 5 ثانیه آلت، حتی در صورت که خود در این مکان حضور نداشته باشد، به هر انتیتی دیگری که به طور کامل وارد این ناحیه شود در هر ثانیه 2 واحد آسیب می‌زند) همچنین محل اثر حملات AoE باید در هر فریم بازی قابل دیدن باشد.</p> <p>- فقط می‌تواند با شلیک تیر به Epsilon آسیب بزند.</p> <p>- امتحانی: نقاط اثر حملات AoE این دشمن باید در گذر زمان اثر این حملات، کمرنگ‌تر شوند.</p> <p>- امتحانی: Archmire باید بعد از مرگ به دو قسمت کوچکتر تقسیم شود که هر یک نصف HP, Drown Damage, AoE Damage, Archmire اصلی،</p> | Archmire: 5 Collectibles 6 XP per each Mini-Archmire: 2 Collectibles 3 XP per each | Drown: 10 AOE: 2 | 30 |  Archmire |
| <p>- این دشمن در یک فریم ایزومنتریک و غیر صلب و تقریباً هماندازه با خودش در صفحه ظاهر می‌شود. ابتدا به سمت Epsilon حرکت می‌کند تا به فاصله مشخصی از او برسد. سپس با همین شعاع حول Epsilon شروع به حرکت می‌کند و به سمت او تیرهای غیر صلب شلیک می‌کند.</p> <p>- این دشمن یک انتیتی صلب و Non-Hovering است. مکانیک Impact روی این انتیتی اثر ندارد و صرفاً در صورت برخورد به هر یک از انتیتی‌های دیگر، جهت حرکت (پاد ساعتگرد یا ساعتگرد) را تغییر می‌دهد. (وقتی کنید در صورت برخورد با یک انتیتی دیگر، انتیتی دیگر تحت اثر Impact قرار می‌گیرد و صرفاً Wyrm از این اثر مستثنیست) این دشمن قابلیت Melee ندارد.</p> <p>- Epsilon فقط می‌تواند با شلیک تیر به Wyrm آسیب بزند.</p> | 2 Collectibles 8 XP each | Ranged: 8 | 12 |  Wyrm |

Mini-Boss Enemy Types

| Aggression Mechanics/Behaviour | Collectibles (Number/XP per each) | Damage Per Hit | Health Points | Character name |
|---|--|--|------------------|--|
| <p>این دشمن در دو گونه متفاوت در بازی ظاهر می‌شود: (به صورت تصادفی)</p> <ul style="list-style-type: none"> - این دشمن هم مانند Wyrm در یک فریم ایزو متريک و تقریباً هم اندازه با خودش در صفحه ظاهر می‌شود. این دشمن آسیب ناپذیر است و به مدت 2 دقیقه در بازی قرار خواهد داشت و بعد از آن به صورت خودکار از بین می‌رود. این دشمن در مقایسه با دشمن‌های دیگر بازی بزرگتر است و از حملات Melee نیز بهره نمی‌برد. این دشمن تا زمانی که در بازی حاضر است، یک انتیتی ثقل خواهد بود و در نتیجه مانع برای عبور انتیتی‌های Non-Hovering به حساب می‌آید. - در گونه دوم، نه تنها Barricados یک انتیتی ثقل است، بلکه فریم آن هم صلب خواهد بود و به عنوان مانع برای بزرگتر شدن فریم‌های دیگر توسط Epsilon نیز عمل خواهد کرد. | - | - | ∞ |  Barricados |
| <p>فرایند ظاهر شدن این دشمن در صفحه نمایش به این صورت است:</p> <ul style="list-style-type: none"> - ابتدا یک فریم توخالی، ایزو متريک و غیر صلب به اندازه یک گوی در صفحه نمایش ظاهر می‌شود. (می‌توانید با نمایش نمادی در این فریم، بازیکن را از تشکیل Black Orb مطلع کنید) - سپس هر بار فریم توخالی، ایزو متريک و غیر صلب دیگر هم اندازه با فریم قبلی، در راسی دیگر از یک پنج ضلعی منتظم مشخص (مانند تصویر) اضافه می‌شود تا هر پنج راس ایجاد شود. - در نهایت در هر فریم یک گوی سیاه (مانند تصویر) ظاهر شده و بین هر دو گوی سیاه یک اشعه لیزر با ضخامت مشخص ظاهر خواهد شد که به هر انتیتی دیگری که در این ناحیه قرار بگیرد در هر ثانیه 12 واحد آسیب می‌زند. این دشمن از حملات Melee برهه نمی‌برد. - پس از فعال شدن Black Orb، تمامی گوی‌ها انتیتی‌هایی ثقل خواهند بود. این دشمن از انجام Avalanche می‌تواند با استفاده از شلیک تیر و حملات Melee به آنها آسیب بزند. - با از بین رفتن هر یک از گوی‌های Black Orb تمام لیزرهای متصل به آن از بین خواهند رفت. - امتیازی: هر Black Orb تنها یک بار پس از فعال شدن، حمله‌ای به نام Avalanche انجام می‌دهد که در آن یکی از نواحی تشکیل شده توسط لیزرهای بین گوی‌های موجود (در صورت وجود) خالی می‌شود و هر انتیتی‌ای که کاملاً در این ناحیه قرار بگیرد، از صفحه بازی به پایین سقوط می‌کند. شما باید قبل از وقوع این حمله، هشدار وقوع را در ناحیه مربوطه نمایش دهید. | 5 Collectibles 30 XP each | Laser: 12 30 per each | |  Black Orb |

Final Boss

وقتی Wave پایانی بازی را به پایان رسانده و تمام دشمنان این Wave را از بین بردید، تمام تروجان‌های سیستم به همراه تمامی فریم‌ها به جز فریم موضعی Epsilon از بین می‌روند (بسته می‌شوند) و فریم موضعی Epsilon به وسط صفحه (مانند وضعیت آغازین بازی) منتقل می‌شود و عملکرد تمام کلیدها و موس غیرفعال می‌شود و شما بالاخره با تروجان اصلی مواجه می‌شوید! Smiley! (که نام آن از تروجان معروف گرفته شده است) که یک ایموچی در حال لبخند زدن است و در یک فریم غیر صلب و غیر ایزو متريک قرار گرفته است از بالای صفحه نمایش به آرامی به پایین حرکت می‌کند و در جایی که با فریم موضعی Epsilon هم پوشانی نداشته باشد، متوقف می‌شود. سپس دو فریم مشابه که به ترتیب حاوی تصویر ایموچی‌های ، هستند در سمت چپ و راست فریم Smiley ظاهر می‌شوند و نبرد نهایی بازی آغاز می‌شود... Smiley و دستهایش در این نبرد انتیتی‌های مستقل هستند و هر یک فقط در لحظات خاصی (در ادامه ذکر خواهد شد) آسیب پذیرند و در زمان‌های دیگر Epsilon نمی‌تواند به آنها آسیب بزند. میزان HP هر دست Smiley برابر با 100 واحد و HP خود این باس، 300 واحد است. اگر سر اثر Smiley از بین برود، دستهای او نیز همزمان از بین خواهند رفت و باس فایت به پایان می‌رسد. همچنین توجه کنید Smiley و دستهایش تحت Impact یا برخورد جابجایی و تغییر مسیر خواهند داشت. در صورت شکست دادن Smiley به طور خودکار 250 واحد XP دریافت می‌کند.

اتک‌های باس

- در این حمله، دو دست Smiley در دو طرف فریم موضعی Epsilon قرار می‌گیرند و مانع از بازتر شدن این فریم توسط اسپیلون می‌شوند. فریم‌های این دست‌ها در طول این حمله موقتاً صلب خواهند شد. طبیعتاً اجرای این حمله نیازمند وجود هر دو دست Smiley است. سر Smiley در این حمله آسیب پذیر است. (سر Smiley همیشه کمی بالاتر از بین دستانش قرار خواهد داشت)
- در این حمله Smiley به شعاع مشخص حول Epsilon به حرکت در می‌آید و از دستانش به او تیرهایی غیر صلب شلیک می‌کند. دستهای Smiley در این حمله آسیب پذیر خواهند بود.

به محض اینکه HP کل Smiley به کمتر از $\frac{2}{3}$ برسد، حملات او متوقف شده و **Epsilon** از حرکت می‌ایستد. سپس یک فریم جدید حاوی تصویر ایموجی 💣 در صفحه ظاهر می‌شود که Smiley را قادر به انجام حملات دسته Power Attack می‌کند. مشت Smiley آسیب ناپذیر است و تا زمانی که Smiley زنده باشد، در بازی حضور خواهد داشت. اکنون مرحله دوم نبرد آغاز می‌شود...

دقت کنید سر و دست‌ها و مشت Smiley تحت اثر مکانیک گشتاور دورانی قرار نخواهد گرفت و جهت آنها ثابت خواهد بود.

اتک‌های جدید

- **Vomit:** در این حمله Smiley از حرکت می‌ایستد و بعد از چند ثانیه، چندین نقطه از صفحه را برای حمله مشخص می‌کند. سپس در هر یک از این نقاط به شعاعی دلخواه (و طبیعتاً محدود) حملات AOE انجام می‌دهد. سر Smiley حین این حمله آسیب پذیر است.
- **Power Punch:** در این حمله Smiley مشت خود را حرکت می‌دهد و به یکی از دیواره‌های خارجی فریم موضعی **Epsilon** می‌کوبد. این کار باعث می‌شود این فریم از سمت انتخاب شده به داخل هل داده شده و فریم را کوچکتر کند که کار را برای **Epsilon** سخت تر می‌کند.
- در این حمله به طور موقت فریم دست Smiley به فریمی صلب تبدیل خواهد شد. در این حمله آسیب‌پذیر خواهد بود.
- **Quake:** طی این حمله، Smiley با کوبیدن مشت به پایین صفحه نمایش، یک موج **Impact** بزرگ از خود ساطع می‌کند. در نتیجه این اتفاق، به مدت 8 ثانیه تمام کلیدهای ورودی و موس بازی کارکرد نادرست خواهد داشت و عملکردهای آنها با هم جابجا می‌شود.
- **Rapid Fire:** در این حمله Smiley برای به مدت محدودی (مثلاً 30 ثانیه) شروع به شلیک متتمادی تیرهای غیر صلب در جهات مختلف (از سرش) می‌کند. طبیعتاً تیرهای Smiley نباید بر خودش اثر داشته باشند. سر Smiley در این حمله آسیب‌پذیر است.
- **Slap:** در این حمله Smiley با زدن یکی از دست‌ها یا مشتش به کاراکتر **Epsilon** به او آسیب می‌زند. دقت کنید فریم دست‌ها و مشت فریم‌هایی غیر صلب خواهد بود اما خود کاراکترهای دست یا مشت، کاراکترهایی Non-Hovering هستند که می‌توانند با بقیه انتیتی‌های بازی برخورد داشته باشند. در این حمله سر Smiley آسیب‌پذیر خواهد بود.
- **Annihilator:** در این حمله، Smiley چند ناحیه از زمین را مشخص کرده و بعد از هشدار به بازیکن، آنها را سوراخ می‌کند! عبور از این نواحی صرفاً برای **Epsilon** ناممکن خواهد بود. (حتی در صورتی که **Epsilon** کامل در این ناحیه قرار نگرفته باشد) همچنین قسمت زیر این نواحی باید قابل مشاهده باشد. (مثلاً اگر بازی روی صفحه دسکتاپ انجام می‌شود، باید والپیپر شما در قسمت‌های سوراخ شده قابل دیدن باشد!

مکانیک‌های باس

- در صورتی که **Epsilon** تقریباً زیر سر Smiley (و بین دست‌هایش) قرار گرفته باشد، Smiley اقدام به حمله **Squeeze** خواهد کرد.
- مکانیزم حمله Projectile می‌تواند مثل عملکرد Wyrm باشد. توجه کنید در صورت نداشتن دست، Smiley اقدام به این حمله نمی‌کند.
- در صورتی که **Smiley** فاصله زیادی با **Epsilon** داشته باشد، به طور تصادفی از حملات دوربرد خود استفاده خواهد کرد.
- تنها می‌تواند از طریق شلیک تیر و حملات دوربرد به **Smiley** آسیب بزند. همچنین به محض شلیک به سمت دست‌ها یا سر، **Smiley** جابجا خواهد شد تا خود را از اصابات تیر نجات دهد. (این مکانیک هم از عملکرد اصلی تروجان اقتباس شده است): از آنجا که دست‌ها، سر و مشت **Smiley** مستقل از هم هستند، او همچنین می‌تواند در آن واحد چندین حمله همزمان (در صورت امکان) انجام دهد. (مثلاً **Smiley** می‌تواند به طور همزمان Quake, Slap, Rapid Fire آغاز کند اما نمی‌تواند همزمان Squeeze, Slap, Rapid Fire را با هم اجرا کند!)

صحنه پایانی بازی

بعد از شکست دادن **Smiley**، دست‌ها و مشتش از بین می‌رونند و ایموجی‌اش به 💀 تبدیل می‌شود. سپس کوچکتر و کوچکتر می‌شود تا از بین برود. در نهایت **Epsilon** مانند صحنه پایانی فاز گذشته، بزرگتر و بزرگتر می‌شود تا کل فریم را پر کند. در این لحظه یک پیغام به کاربر نشان داده می‌شود که تعداد تیرهای شلیک شده، تعداد شلیک‌های موفق و تعداد دشمن‌های کشته شده، XP به دست آمده و مدت زمان بازی را به کاربر نشان می‌دهد. طبیعتاً گزینه بازگشت به منوی بازی باید در این پیغام موجود باشد. (دقت کنید مدت زمان صرف شده در منو نباید به عنوان مدت زمان بازی کاربر محاسبه شود و در این زمان‌ها باید تایмер را **متوقف** کنید)

ذخیره بازی

یکی از موارد مهم که همواره در پیاده‌سازی بازی‌ها (بالاخص بازی‌های fast-paced) ذخیره کردن وضعیت بازیست، تا بتوانیم در زمان‌های تعیین شده پیشروی بازی را ذخیره کنیم. البته از آنجا که همیشه زندگی بر وفق مراد نیست، ممکن است بر اثر کرش در بازی یا سیستم عامل، خاموش شدن اتفاقی سیستم و یا حتی وجود تروجانی مثل YouAreAnIdiot ۰۰۰، بازی در لحظه‌ای پیش‌بینی نشده متوقف شود! از این رو برای جلوگیری از از دست دادن پیشروی کاربر در این لحظات بفرزنج، به Real-time progress save یا فرایند سیو بازی در لحظه نیاز داریم. در نتیجه، یکی از موارد مورد انتظار از شما در این فاز، فرایندی است که بتواند در حین بازی، اطلاعات مهم آن را در یک فایل (با فرمت و روش دلخواه)، ثبت و ذخیره کند و بعداً آنها را مجدداً اجرا کند. استیت ذخیره شده باید شامل تمام اجزای وضعیت ذخیره شده بازی باشد (اعم از همه انتیتی‌ها، تیرها، فریم‌ها به انضمام وضعیت آنها، جهت و سرعت حرکت همه انتیتی‌ها و غیره) بدین معنا که لود شدن مجدد فایل سیو با ادامه بازی تفاوتی نکند. دقت کنید در این بازی مکانیک سیو شما شامل سیو در چکپوینت‌ها و سیو مرتب بازی در بازه‌های زمانی مختلف است.

امتیازی (Save Validation): مرگ همیشه یکی از ساختارهای تنبیه‌ی در بازی‌هاست. طبیعتاً شما هنگامی که با پیشروی بالایی در یک بازی می‌بازید، بعضاً مایلید به هر روشی پیشروی قبلی خود را به دست آورید. اگر شما به محل سیوهای چکپوینت دسترسی داشته باشید و سیو لحظه‌ای خود را نیز در اختیار داشته باشید، شاید بتوانید با جایگزین کردن این دو، فایل سیو لحظه آخر بازی خود را اجرا کنید و پیشروی خود را از دست ندهید! وظیفه شما در این بخش این است که ساز و کاری طراحی کنید که یا دسترسی به فایل سیو لحظه‌ای را غیر ممکن کند و یا در خوانش این فایل سیو به عنوان سیو چکپوینت مشکلی ایجاد کند تا بازیکن‌ها موفق به دور زدن مکانیزم سیو شما نشوند.

قبل از شروع یک بازی جدید، (در صورتی که فایل سیو لحظه‌ای از بازی قبل باقی مانده باشد) شما باید پیغامی به کاربر نشان داده و از او بپرسید که آیا مایل است بازی سیو شده قبلى (که قبل از پایان یا مردن کاراکتر بازی قطع شده است) را ادامه دهد یا خیر. در صورتی که کاربر این پیشنهاد را رد کند، این فایل حذف می‌شود و بازی جدیدی شروع می‌شود اما در صورتی که کاربر این پیشنهاد را قبول کند، شما باید ابتدا به مدت چند ثانیه وضعیت بازی را بی حركت به او نشان دهید و سپس بازی را ادامه دهید. دقت کنید در صورت خروج از بازی از طریق منوی تعییه شده در بازی، شما نباید این فایل سیو لحظه‌ای را ذخیره کنید و این فایل سیو صرفاً برای خروج‌های پیش‌بینی نشده در فرایند بازی خواهد بود. بنابراین شما باید در طول اجرای بازی، آن را با نرخ ثابتی (هر چند ثانیه یک بار) ذخیره کنید تا مانع از از دست رفتن اطلاعات بازی در این موقع شوید.

امتیازی (Save Synchronization): در فرایند سیو یک بازی fast-paced با تعداد زیادی فیلد و آبجکت، علاوه بر مشکلات Concurrency معمول که در فرایند سیو بازی رخ می‌دهد، یکی از مسائل مهم همزمانی داده‌های رونوشت شده است. به طور خلاصه ممکن است شما در لحظه‌ای در کد خود تصمیم به ذخیره‌سازی بازی بگیرید اما چون فرایند سیو به ترتیب روی آبجکت‌های مختلف در حال انجام است، در طول این فرایند تغییری در وضعیت بازی (حتی در حد حرکت یک انتیتی به اندازه یک پیکسل) رخ دهد و بعضی از آبجکت‌های شما قبل از این تغییر و مابقی پس از این تغییر ذخیره شده باشند. در این صورت دیتای ذخیره شده شما لزوماً با هم همخوانی نخواهد داشت. (برای مثال ممکن است در دیتای ذخیره شده شما دو فریم صلب با هم همپوشانی داشته باشند در صورتی که در فرایند عادی بازی این وضعیت هیچگاه رخ نخواهد داد) شما در این بخش باید با طراحی یک روند سیو مناسب، این مشکل را از سر راه بردارید و مطمئن شوید دیتای ذخیره‌سازی شده شما synchronized شده است.

گریزی مجدد به SOLID

یکی از مشکلات بنیادی در روند پیاده‌سازی، ایجاد Cyclic Dependency در ساختار کد است. برای اینکه شهود مناسبی از این مشکل داشته باشید، تصور کنید که سیستم برای لود یک فایل سیو، به ترتیب مشخصی serialization های تولید شده در فرایند سیو را به آبجکت‌هایی در کد شما تبدیل می‌کند. در نتیجه نیاز است قبل از ایجاد هر آبجکت، تمام فیلد‌های آن آبجکت‌هایی شناخته شده باشند. حال فرض کنید شما آبجکت‌های X_1, X_2, \dots, X_n را در ساختار کد خود دارید که در آن، X_1 به عنوان فیلدی از X_2 ... X_n و X_2 به عنوان فیلدی از X_3 ... X_n معرفی شده است. در این صورت سیستم برای لود فایل شما، مستقل از اینکه از کدام آبجکت آغاز به deserialization کند، با آبجکت ناشناخته‌ای برای معرفی به عنوان یک فیلد مواجه خواهد شد که باعث ایجاد Compilation Error در کد شما می‌شود. با اینکه حل این مشکل معمولاً به تغییرات بنیادی در کد نیاز خواهد داشت، یکی از راه‌های جلوگیری از این مشکل در ساختار کد، رعایت اصول SOLID است. در نتیجه توصیه می‌شود اصول پایه‌ای SOLID را به درستی در کلاس‌هایی که نیاز به ذخیره شدن اطلاعات دارند، رعایت کنید و همانطور که در مقدمه نیز اشاره شد، کلاس‌های خود را تا حد امکان به ساختار POJO نزدیک نگه دارید تا این ایجاد معماری مصون بماند!

Skill Tree v2.0

در این فاز شما به تکمیل کردن **Skill Tree** خواهید پرداخت. همانطور که در فاز قبل مشاهده کردید، قابلیت‌های Epsilon در سه دسته **حمله**، **دفاع و تغییر شکل** طبقه‌بندی می‌شوند. همچنین گفته شد برای باز کردن هر قابلیت، باید تمام قابلیت‌های قبلی آن دسته را باز کرده باشید. اما در فاز قبل صرفاً یک قابلیت از هر دسته معرفی شد و در این فاز به تعریف قابلیت‌های جدیدی برای Epsilon خواهیم پرداخت.

قابلیت‌های دسته حمله (به ترتیب)

- **Writ of Ares**: قابلیت اول دسته **حمله**. مطابق توضیحات **فاز قبل** عمل می‌کند.
- **Writ of Astra**: قابلیت دوم دسته **حمله**. با پرداخت 1000 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، تمام انتیتی‌ها در برخورد به **Epsilon** متحمل 2 واحد آسیب خواهند شد. (دققت کنید منظور از برخورد در این قابلیت، صرفاً **Melee Attack** های **Epsilon** نیست و هر گونه برخوردی مورد قبول خواهد بود.)
- **Writ of Cerberus**: قابلیت سوم دسته **حمله**. با پرداخت 1500 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، سه کاراکتر دایره‌ای شکل **Hovering**، غیر صلب و کوچکتر از **Epsilon** در مجاورتش ظاهر می‌شوند. در صورتی که هر انتیتی (غیر از **Epsilon**) کاملاً روی هر یک از آنها قرار بگیرد، متحمل 10 واحد آسیب خواهد شد. بعد از انجام یک حمله موفق، هیچ یک از این سه کاراکتر تا 15 ثانیه قادر به حمله مجدد نخواهد بود.

قابلیت‌های دسته دفاع (به ترتیب)

- **Writ of Aceso**: قابلیت اول دسته **دفاع**. مطابق توضیحات **فاز قبل** عمل می‌کند.
- **Writ of Melampus**: قابلیت دوم دسته **دفاع**. با پرداخت 750 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، به احتمال 5 درصد **Epsilon** در طی حملات **Melee** آسیب نمی‌بیند.
- **Writ of Chiron**: قابلیت سوم دسته **دفاع**. با پرداخت 900 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، با اعمال هرگونه آسیب به هر انتیتی دشمن 3 واحد به HP شما اضافه خواهد شد.
- **Writ of Athena**: قابلیت چهارم دسته **دفاع**. با پرداخت 1200 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، فریم موضعی **Epsilon** با 80% سرعت اولیه به **shrinkage** ادامه خواهد داد. همچنین این قابلیت صرفاً پس از باز کردن قابلیت دوم دسته **حمله** نیز می‌تواند باز شود.

قابلیت‌های دسته تغییر شکل (به ترتیب)

- **Writ of Proteus**: قابلیت اول دسته **تغییر شکل**. مطابق توضیحات **فاز قبل** عمل می‌کند.
- **Writ of Empusa**: قابلیت دوم دسته **تغییر شکل**. با پرداخت 750 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، اندازه **Epsilon** به اندازه 10% کوچکتر خواهد شد.
- **Writ of Dolus**: قابلیت سوم دسته **تغییر شکل**. با پرداخت 1500 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، دو تا از قابلیت‌هایی که قبلاً در Skill Tree باز شده‌اند به صورت تصادفی فعال خواهند شد. دقت کنید با اعمال دوباره این قابلیت بعد از گذشت **cooldown** آن، مجدداً همین دو قابلیت برای **Epsilon** فعال می‌شوند و در واقع هر یک از این دو قابلیت مجدداً فعال خواهد شد.

فروشگاه جدید

- در ادامه پیاده‌سازی فروشگاه در **فاز قبل**، در این فاز صرفاً سه قابلیت جدید به قابلیت‌های قابل خریداری اضافه خواهند شد:
- **O' Deimos, Dismay**: در این قابلیت بازیکن با پرداخت 120 واحد از XP هایش، همه دشمن‌های Non-Hovering را به مدت 10 ثانیه تا شعاع مشخص دور نگه می‌دارد. در این زمان هیچ دشمن Non-Hovering ای قادر نیست از این شعاع به **Epsilon** نزدیکتر شود.
 - **O'Hypnos, Slumber**: با پرداخت 150 واحد XP، این قابلیت اجرا می‌شود و تمام حرکات بازی (اعم از همه دشمن‌ها، تیرها و لیزرها و فریم‌ها) به استثنای **Epsilon** و تیرهایش را به مدت 10 ثانیه متوقف می‌کند و پس از گذشت این 10 ثانیه بازیکن مجدداً می‌تواند این قابلیت را فعال کند. همچنین در حین این 10 ثانیه بازیکن نباید قادر به خرید قابلیت جدیدی از طریق فروشگاه بازی باشد.
 - **O' Phono, Slaughter**: بازیکن باید قادر باشد با پرداخت 200 واحد از XP هایش، تنها یک تیر با 50 واحد آسیب شلیک کند. استفاده از این قابلیت همچنین دارای **cooldown** به مدت 120 ثانیه می‌باشد.

Spawn مکانیک‌های

در این فاز مشابه فاز قبل، شما باید ساز و کاری برای گسیل کردن (!) دشمن‌ها و مبنی‌بازندهای بازی به سمت بازیکن داشته باشید. با اینکه جزئیات پیاده‌سازی این مکانیک‌ها بر عهده خودتان است، لازم است این موارد را در این فرایند رعایت کنید:

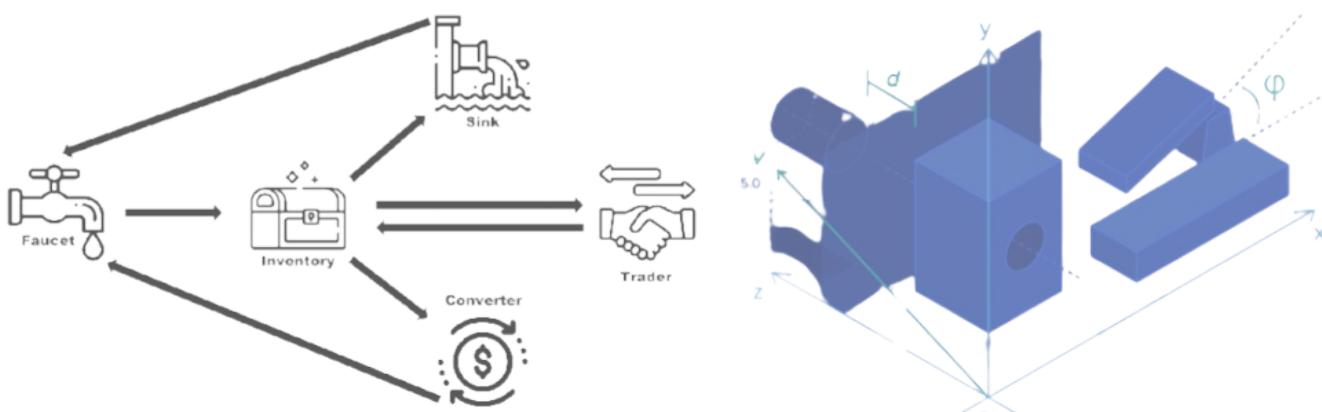
- فرایند Spawn دشمن‌ها مشابه فاز قبل در طی تعدادی Wave انجام می‌شود اما مکانیزم فعالیت Wave‌ها دچار تغییر خواهد شد. در این فاز برخلاف فاز قبل (که در هر Wave تعداد محدودی دشمن به سمت Epsilon فرستاده می‌شد) تعداد دشمن‌ها در هر Wave محدود نیست! در این فاز، دشمن‌های مختلف در نقاطی تصادفی از صفحه بازی Spawn می‌شوند و این فرایند زمانی که تعداد مشخصی از دشمن‌ها در آن Wave کشته شوند، متوقف می‌شود. سپس بازیکن باید تمام دشمنان باقی‌مانده در Wave را از بین Wave فعلی به پایان برسد و Wave جدید آغاز شود. از شما انتظار می‌رود در طی بازی حداقل 5 Wave مختلف (به غیر از آخر یا همان باس‌فایت) طراحی کنید. همچنین تمام دشمن‌ها باید در این Wave‌ها قرار داده شده باشند.
- در این فاز، شما باید مکانیزم "سختی افزاینده" را پیاده‌سازی کنید. در این مکانیزم، شما باید نرخ Spawn شدن دشمن‌ها در هر Wave را با گذشت زمان آن Wave افزایش دهید. در این صورت در گذر زمان هر Wave شکست دادن آن Wave برای بازیکن سخت‌تر و سخت‌تر خواهد شد که او را از بازی محناطانه دور کرده و به درگیر شدن بیشتر پلیر و فعال‌تر بودن او (Engaging Gameplay) منجر می‌شود.
- انواع دشمن‌هایی که در هر Wave برای Spawn برنامه‌ریزی شده‌اند باید مشخص باشد. طبیعتاً باس‌فایت تنها در آخر Wave های اول ظاهر نمی‌شوند.
- مشابه فاز قبل توجه کنید تعداد بیش از حد دشمن در یک صحنه، پروفورمنس بازی و همچنین تجربه پلیر را تحت الشاعع قرار خواهد داد.
- در نتیجه سعی کنید با آزمون و خطای مقادیر مناسبی را برای نرخ Spawn دشمن‌ها به دست آورید.

Game Design مختومه‌ای بر

همانطور که در این دو فاز دیدید، بازی تعریف شده تفاوت‌های بزرگی با مکانیک‌های بازی اصلی دارد که عمدۀ این تغییرات برای متواظن‌تر کردن مکانیزم‌های بازی هستند. در طی تولید و طراحی یک بازی، نکات تئوری زیادی وجود دارند که می‌توانند در عین ریز و کم اهمیت بودن، بازی‌ها را بسیار جذاب یا خیلی حوصله سر بر و تکراری کنند. چنین نکات ریزی در مکانیک‌ها، Economy و فیزیک بازی‌ها باعث می‌شوند بازیکن بدون اینکه حتی خودش متوجه باشد، مستقل از اینکه بازی در ژانر مورد علاقه‌اش باشد، درگیر مکانیک‌های بازی شود. از این رو در نهایت بازی‌هایی بسیار ساده مثل Stardew Valley، Minecraft و Demon's Souls موفق‌تر از بازی‌هایی با گرافیک بسیار پیچیده‌تر مثل Final Fantasy و GMTK: Design of games' economy

فاکتورهایی مثل Death Design, Game Economy, Combat System, Camera Design و مواردی از این دست تفاوت‌های شگرفی در درگیر شدن پلیرها در بازی ایجاد می‌کنند و در نتیجه بازی‌هایی که در این موارد به نحو احسن عمل می‌کنند، در نهایت هم بسیار موفق‌تر از دیگر بازی‌ها خواهند بود. لینک‌های زیر برای علاقمندان به این موضوعات قرار داده می‌شود:

1. [GMTK: Design of games' economy](#)
2. [GMTK: Design of combat systems](#)
3. [GMTK: How physics affect platformer games](#)
4. [GMTK: Analysis of video game designs](#)
5. [Designing better game economies](#)



باز هم بخش امتیازی!

[Omenoc Wallrun](#)

[Necropick Alert](#)

[Archmire AoE Fade](#)

[Archmire Mitosis](#)

[Black Orb Avalanche](#)

[Annihilator](#)

[Save Validation](#)

[Save Synchronization](#)

[Writ of Athena](#)

Log

در این بخش از شما انتظار می‌رود با استفاده از لاپبری‌هایی اعم از Log4j, Logback, Slf4j,... تمام وقایع کد خود را به صورت لگ در یک فایل قابل خواندن ذخیره کنید. لگ‌های شما باید شامل تمام حرکت‌ها، شلیک‌ها، آسیب‌ها، اتک‌ها، spawn شدن دشمن‌ها در صفحه بازی، خربدهای فروشگاه و پرداخت XP برای قابلیت‌ها، به دست آوردن XP از طریق گرفتن Collectible ها، تغییر اندازه فریم‌ها و غیره باشد. برای گرفتن نمره این بخش شما باید دسته بندی مناسبی از لگ‌های خود ارائه دهید و آنها را به درستی در سطوح SEVERE, WARNING, INFO, DEBUG طبقه‌بندی کنید. نوشتن یک لگ کامل همچنین می‌تواند برای دیباگ کد، بررسی کرش‌ها و (در سطوح بالاتر) حتی مطالعه رفتار پلیرها به شما کمک کند.

Wyrm Horde

در این بخش شما بجای پیاده‌سازی Wyrm و استفاده از آن در دسته‌های چندتایی، یک دشمن به نام Wyrm Horde را پیاده‌سازی می‌کنید. این دشمن در واقع تعدادی Wyrm در فریم‌های مختلف است که از راس‌های دو طرف خود به یکدیگر متصل شده‌اند. دقت کنید چالش پیاده‌سازی شما برای این دشمن این است که با اینکه هر Wyrm به عنوان یک دشمن مستقل عمل می‌کند، اما حرکات هر Wyrm روی بقیه اعضای این دشمن تاثیرگذار است! به این معنا که اگر برای مثال مطابق **پروتکل بنیادی رفتار فریم‌های موضعی** فریم یکی از Wyrm ها متوقف شود، بقیه Wyrm ها نمی‌توانند به اندازه‌ای حرکت کنند که اتصال بین Wyrm ها به هم بخورد. به عنوان مثال دیگر در صورتی که مسیر حرکت یکی از Wyrm ها به علت حرکت اپسیلون و یا یک **برخورد** دستخوش تغییر شود، بقیه Wyrm ها نیز تغییر در مسیر خود خواهند داشت. همچنین توجه کنید Wyrm های Wyrm Horde می‌توانند **صرفاً با دیگر Wyrm های همان Horde** همپوشانی داشته باشند. دقت کنید به این تعبیر، هر Wyrm خود یک انتیتی Wyrm Horde یک عددیست. در نهایت دقت کنید در صورتی که یکی از Wyrm های میانی یک Wyrm Horde کشته شود، Wyrm Horde اولیه به دو جزا تقسیم خواهد شد و هر یه به صورت مجزا به فعالیت ادامه خواهد داد. مکانیزم حمله Wyrm مشابه تعریف قبل است.

استفاده از روش‌های مبتنی بر PCA

همانطور که تا در این داک دیدید، در این بازی بعضًا به بررسی برخورد بین انتیتی‌هایی که لزوماً چندضلعی نیستند، نیازمندیم. به طور کلی در این بازی برای انتیتی‌های غیر چندضلعی تنها بررسی برخورد با هیبتاکس‌های مستطیلی از شما انتظار خواهد رفت. اما دقیق کردن این برخوردها، علاوه بر جلوه بصری معقول‌تر، جنبه استراتژیک بازی را پررنگ‌تر خواهد کرد. در این بخش، دقیق کردن برخوردها **برای انتیتی‌های دلخواه** با روش‌های Polygonization نمره امتیازی به همراه خواهد داشت. مطالعه محتوای **این کارگاه** می‌تواند مفید باشد.



پیاده‌سازی جعبه‌ای
فاقد نمره امتیازی جزئی



پیاده‌سازی با تقریب چندضلعی
نمود نمره امتیازی جزئی



پیاده‌سازی دقیق
نمود نمره امتیازی کامل

To be continued...

با آرزوی موفقیت و سر بلندی روز افزون
تیم درس برنامه نویسی پیشرفته

