

برنامه‌نویسی پیشرفته  
زمستان ۱۴۰۲ - بهار ۱۴۰۳  
فاز اول پروژه

تیم تعریف و طراحی پروژه:  
آرین همتی، سینا دانشگر

آیدی‌ها روی اسامی لینک شده‌اند و کافیست روی آنها کلیک کنید



# مقدمه‌ای بر فاز اول پروژه (حتماً بخوانید!)

به پروژه درس برنامه‌نویسی پیش‌رفته خوش آمدید. 🎉 این پروژه فرصتیست که در طی آن با اصول برنامه‌نویسی شی‌گرا **عمیقاً** آشنا شوید. در این فاز شما شالوده‌گرافیکی و مقدار قابل توجهی از لاجیک یک بازی که برای شما طراحی شده است را پیاده‌سازی خواهید کرد. اهداف آموزشی این فاز شامل گرافیک پیشرفته، (تشخیص برخورد و همپوشانی، پیاده‌سازی الگانه‌های گرافیکی و ...)، کار با جریان‌ها و جنریک است. به همین‌طور از اهداف عملی و پیاده‌سازی که در این فاز اهمیت دارد، استفاده صحیح از پارادایم OOP است. با اینکه Design Pattern ها از اهداف آموزشی این فاز پروژه نیستند، اما **اکیداً** توصیه می‌شود درباره معماری MVC مطالعه کنید و آن را **حتماً** در پیاده‌سازی این فاز، و فازهای آتی اعمال کنید. در این فاز لایه‌هایی از لاجیک و گرافیک بازی روی یکدیگر اضافه خواهند شد و با رعایت کردن معماری MVC، شما می‌توانید در عین راحتی کد زدن، توسعه‌پذیری و تغییر پذیری آن را حفظ کنید و در فازهای بعدی پروژه نیز زمان کمتری برای رفع مشکلات بالقوه در عملکرد لایه‌های مختلف از کد خود صرف کنید. (**Spoiler Alert**: در فاز آتی احتمالاً به مقدار عظیمی دیباگ نیاز پیدا خواهید کرد) 🤪 فلذا اکیداً توصیه می‌شود درباره این اصول مقداری مطالعه کنید و قبل از دست به کد شدن، چندین بار داک را به طور کامل بخوانید و زمان کافی صرف کنید تا راهبردی دقیق برای رعایت MVC طراحی کنید. همچنین توصیه می‌شود کلاس‌های مربوط به مدل‌های پایه خود (که بعداً باید در فرآیند سیو در فایل ذخیره شوند) را طوری طراحی کنید که حتی‌الامکان به مدل (Plain Old Java Object) POJO باشند. (تا در فرآیند **serialization** به مشکلات بنیادی که نیاز به تغییر اساسی در کد و معماری دارند، بر نخورید) کلیتی از محتوایی که برای یادگیری اصول SOLID و معماری نیازمندید در محتوای آموزشی آماده شده توسط تیم در [این لینک](#) فهرست شده است. منابعی ابتدایی برای یادگیری این مطالب در کنار روند درس در [این لینک](#) ضمیمه شده است. همچنین در تکمیل منابع قبل، منبعی بسیار سودمند و کامل برای یادگیری این مباحث در [این لینک](#) آمده است.

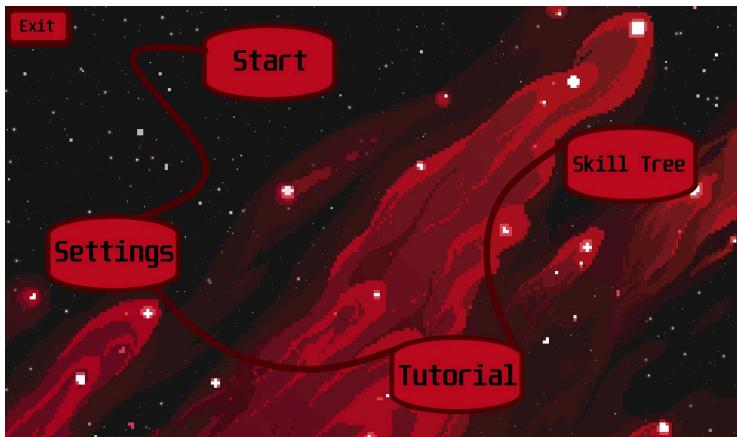
نکته بسیار مهم: تعیین مقادیر ثابت مناسب، به نحوی که تجربه معقولی از بازی به کاربر افزا کند، در سرتاسر پروژه بر عهده شماست و معقول بودن این مقادیر ثابت، حائز نمره می‌باشد. لذا از شما انتظار می‌رود از انتخاب مقادیر نامعقول برای ثوابتی اعم از اندازه کارکتر، اندازه فریم، مقادیر تغییر اندازه فریم، **collectible** ها و سایر موارد پرهیز کنید تا بازی شما در فرآیند تحويل و ارزیابی با مشکلات اساسی مواجه نشود.



1.....	مقدمه‌ای بر فاز اول پروژه (حتماً بخوانید!)
3.....	صفحه ورود بازی
3.....	همه چیز از اینجا آغاز می‌شود! (شروع بازی)
3.....	دکمه‌های بازی
4.....	محیط بازی (Game Env.)
4.....	اطلاعات بازی (HUI)
4.....	صدا
4.....	ولی من کیستم؟... (تعریف کاراکتر بازی)
4.....	من کجا هستم؟... (تعریف فریم بازی)
5.....	روغن‌کاری محیط گرافیکی
5.....	مرگ (Death Mechanics)
5.....	مکانیک‌های ضربه، برخورد و آسیب!
6.....	GameOver و اما
6.....	مکانیک‌های شلیک
6.....	دشمن‌ها (Enemy)
7.....	Melee attack (MA)
7.....	Local Routing
7.....	Progress Rewards, a PFL (Positive Feedback loop)
7.....	Enemy Types
7.....	مکانیک‌های Aggression
8.....	به منو باز می‌گردیم!
8.....	فروشگاه
8.....	Skill Tree
8.....	Settings
9.....	مکانیک‌های spawn و wave های مختلف حملات
9.....	سختی بازی
9.....	پایان بازی
9.....	ابزارهایی که از شما انتظار می‌رود در توسعه کدتان از آنها استفاده کنید
9.....	استفاده از ابزار گیت.
9.....	استفاده از Build Tool (میون و گریدل)
10.....	بخش‌های امتیازی!
10.....	دوران اپسیلونی!
10.....	تنظیمات مربوط به دکمه‌های بازی
10.....	پیاده‌سازی گشتاور دورانی
10.....	Broad/Narrow Search
10.....	استفاده از ابزار JLINK
10.....	سخن پایان

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 یا لینک منابع مفید منتقل شوید فلذاً حتماً به این قسمت‌ها توجه ویژه داشته باشید.

## صفحه ورود بازی



در ابتدای اجرای برنامه، برنامه شما باید یک منوی اولیه به کاربر نشان دهد. در این منو باید این دکمه‌ها موجود باشد:

- شروع بازی
- Skill Tree
- راهنمای بازی
- تنظیمات بازی
- خروج از بازی

کارکرد همه این گزینه‌ها به تفضیل در ادامه داک توضیح داده شده است. بازی در اینجا آغاز می‌شود...

### همه چیز از اینجا آغاز می‌شود! (شروع بازی)

بعد از کلیک روی گزینه شروع بازی، منو بسته شده و تمام برنامه‌های باز باید وارد حالت minimized شوند. سپس یک فریم مربعی بزرگ (مثلاً 700\*700 پیکسل) باز می‌شود و شخصیت اصلی بازی که یک دایره کوچک (مثلاً به قطر 25 پیکسل) است، در این فریم ظاهر می‌شود. سپس فریم به سرعت کوچک می‌شود تا به اندازه‌ای در حدود 200\*200 پیکسل برسد. وقت کنید تمام حرکات پیاده‌سازی شده در این بازی باید به صورت پیوسته (smooth) انجام بگیرند. فریم بازی نباید قابل جابجایی و تغییر سایز (همچنین minimize شدن) و بسته شدن باشد.

### دکمه‌های بازی

شما باید دکمه‌هایی برای حرکت و اعمال دیگر پیاده‌سازی کنید. انتخاب دکمه مورد نظر به انتخاب خودتان است:

**حرکت در چهار جهت اصلی:** شما باید چهار دکمه برای حرکت به جهت‌های چپ و راست و بالا و پایین اختصاص دهید.  
• (استفاده از کلیدهای WASD و یا Arrow Key) حرکت در جهت‌های فرعی (شمال شرقی و غربی، جنوب شرقی و غربی) باید با فشار دادن همزمان دکمه‌های مربوطه امکان‌پذیر باشد. فشار دادن همزمان دکمه‌های مربوط به حرکت به بالا و پایین و یا دکمه‌های مربوط به حرکت به چپ یا راست نباید منجر به حرکت کاراکتر شود. مکانیک‌های مربوط به نحوه حرکت کاراکتر در ادامه داک توضیح داده خواهد شد.

**فروشگاه:** در این بازی شما باید دکمه‌ای مربوط به فروشگاه تعییه کنید تا با فشردن آن، تمام اتفاقات و حرکات بازی متوقف شوند و فروشگاه بازی نمایش داده شود. تعریف و طراحی فروشگاه در ادامه توضیح داده شده است.

**هدفگیری و شلیک:** کاراکتر اصلی شما در این بازی باید قادر باشد از خودش در برابر حملات احتمالی (LMAO) محافظت کند. به همین دلیل مکانیزم‌هایی برای هدفگیری و شلیک تیر در بازی تعییه شده است. شما باید دکمه‌ای برای شلیک تیر در بازی خود در نظر گرفته باشید. (استفاده از Left mouse button پیشنهاد می‌شود) هدفگیری و جهت شلیک‌تیر، با استفاده از ماوس کنترل می‌شود. در هر لحظه که دکمه مربوط به شلیک تیر فشرده شود، باید تیر به سمت نشانگر (cursor) ماوس شلیک شود. مکانیک‌های مربوط به شلیک و آسیب در ادامه داک توضیح داده خواهند شد. همچنین شما می‌توانید قسمت امتیازی تعریف شده مربوط به این بخش را نیز پیاده‌سازی کنید.

**قابلیت:** نهایتاً باید دکمه‌ای در بازی شما تخصیص داده شده باشد تا با فشردن آن، قابلیت انتخاب شده بازیکن تا پایان آن بازی فعال شود. تعریف دقیق جزئیات مربوطه، در ادامه داک توضیح داده شده است.

## محیط بازی (Game Env.)

### اطلاعات بازی (HUI)

شما باید در مکانی از صفحه (عموماً بالای صفحه یا گوشه‌های پایینی صفحه) اطلاعات بازی را به کاربر نشان دهید. همچنین این اطلاعات می‌توانند مخفی باشند و با زدن دکمه‌ای خاص، به مدت محدودی (مثلًا 3 ثانیه) در محل تعریف شده نمایش داده شوند و دوباره مخفی شوند. (تصمیم برای نحوه پیاده‌سازی به عهده خودتان خواهد بود) این اطلاعات باید شامل موارد زیر باشند:



Wave	•
Elapsed Time	•
XP (Experience points)	•
HP (Health points)	•

تصویر تزئینی است

### صدا

بازی شما باید دارای صدای پس زمینه باشد. همچنین قسمت‌های پایان هر wave، پایان بازی، آسیب دیدن هر Entity، ورود هر دشمن به صفحه بازی، کشتن هر دشمن، باید دارای صدایها و آهنگ‌های پس زمینه منحصر به فرد باشند.

### ولی من کیستم؟... (معرفی کاراکتر بازی)

شما در این بازی نقش یک شکل هندسی به نام **Epsilon** را ایفا می‌کنید که در ابتدای بازی صرفا یک دایره است. یک تروجان به کدبیس شما حمله کرده و شما احتمالاً تنها برنامه‌نویسی بوده‌اید که فیلدهایش را private می‌کرده است... از این رو، تمام آشکال هندسی دیگر به شما حمله‌ور شده‌اند تا تنها بازمانده را از بین ببرند و کنترل کامل را به دست بگیرند. تروجان موذی، از سوی دیگر کنترل فریم‌های ویندوز را به دست گرفته و قصد دارد شما را با کوچکتر کردن فریم بازی محبوس کند تا سریعتر کنترل فیلدهای شما را به دست بگیرد. شما به کمک یک سلاح باید تا در توان دارید، از خود دفاع کنید تا شاید اگر زنده ماندید، بقیه را قانع کنید تا Encapsulation را در کدهایی که می‌زنند رعایت کنند... هدف شما در این بازی این است که در برابر چندین موج (Wave) از حملات اشکال هندسی شیطانی دیگر جان سالم به در ببرید و آنها را نابود کنید. هر wave از این حملات، تنها هنگامی به پایان می‌رسد که تمام دشمنان موجود در صفحه بازی کشته شوند. مکانیک آسیب و ضربه در **این قسمت** به طور کامل توضیح داده شده است.

### من کجا هستم؟... (معرفی فریم بازی)

کاراکتر **Epsilon** در این بازی در یک کانتینر گرافیکی (JFrame یا امثال‌هم) قرار دارد. همانطور که گفته شد تروجان مگار در تلاش است با کوچک کردن صفحه بازی، شما را گیر بیندازد. بعد از گذشت ده ثانیه از آغاز بازی، چهار دیوار بازی با سرعتی محدود، دائمًا (به استثنای زمان‌های بین دو wave) شروع به کوچکتر شدن می‌کنند. دقت کنید شما باید یک مینیمم اندازه برای فریم بازی قائل شوید و فریم در هیچ لحظه‌ای از بازی نباید از این اندازه ثابت تعیین شده کوچکتر شود. (در این اندازه shrinkage فریم هم به طور موقت متوقف می‌شود تا فریم بزرگ‌تر شود) شما در طول حرکت **Epsilon** باید قادر به عبور از دیوارهای فریم فعلی خود باشید.

توجه کنید در طول فرایند **shrinkage** توسط فریم بازی، کاراکتر **Epsilon** باید همواره در داخل فریم قرار داشته باشد

## روغنکاری محیط گرافیکی...

دریافت FPS بالا و روان‌تر به نظر رساندن بازی، معضلی به قدمت Tennis for Two (در سال 1958) بوده است. از این‌رو، چون ذات محاسبات کامپیوترا گستته است، بازی‌سازها و برنامه‌نویس‌ها دائمًا در طول تاریخ در پی حربه‌هایی بر روان و پیوسته به نظر رساندن این فرآیندهای گستته بوده‌اند. این حربه‌ها؛ ایده‌های هوشمندانه‌ای مثل Ray-Tracing، Anti-Aliasing تا ایده‌های بدیهی‌ای مثل **کنترل حرکت** را شامل می‌شود. از سوی دیگر شما به عنوان بازی‌ساز وظیفه دارید به بازیکن حس مثبتی از بازی القا کنید تا بازیکن بازی شما را فردا هم باز کند. (مگر اینکه در FromSoft مشغول به کار باشید) در این باره عبارتی معروف در میان جامعه بازی‌سازان وجود دارد که تمام تکنیک‌الیتی‌های کنترل حرکت را در یک جمله خلاصه می‌کند: **Make movement feel good** اما این مهم، چگونه قابل حصول است؟! در این فاز از شما انتظار می‌رود در این بازی از Acceleration، Deceleration (حرکت شتابدار) در حرکت دادن Epsilon و همچنین همه Entity‌های مختلف، جابجا کردن فریم‌ها و تغییر اندازه آنها و ... بهره ببرید. منبعی بسیار سودمند، آموزنده و ملموس برای یادگیری این موضوع (و موضوعاتی جامع‌تر در کنترل حرکت) در [این منبع](#) آمده است.

## مرگ (Death Mechanics)

یکی از مهم‌ترین اجزای تعیین کننده در ساختار مکانیک هر بازی، مکانیک‌های مربوط به مرگ است. در این بخش، تعریف آسیب دیدن هر کاراکتر و مکانیک‌های مرگ مربوط به بازی را توضیح خواهیم داد:

### مکانیک‌های ضربه، برخورد و آسیب!

در این بخش آسیب دیدن کاراکتر را به طور دقیق تعریف می‌کنیم. ابتدا تعدادی تعریف انجام می‌دهیم: (در این بخش و در طی فرآیند تشخیص برخوردها می‌توانید قسمت امتیازی این بخش را پیاده‌سازی کنید)

- **بردار حرکت:** به جهت حرکت Epsilon و یا هر دشمن، بردار حرکت مربوط به آن Entity می‌گوییم.
- **ضربه:** به معنای حمله کردن به یک دشمن از روش‌های معتبر و تعریف شده در ادامه داک است.
- **برخورد:** به تماس هر Entity (اعم از Epsilon، دشمن‌ها) به دیواره‌های فریم یا به یک دیگر اطلاق می‌شود. توجه کنید که هرگونه تماس Epsilon با یک دشمن، به نحوه‌ای که ضربه تلقی نشود نیز برخورد محسوب خواهد شد.
- **آسیب:** به منزله هرگونه برخورد Epsilon با دشمن (تحت شرایط تعریف شده در ادامه داک) و یا قرار گرفتن او در ناحیه اثر هر یک از حملات دوربرد آنها خواهد بود. توضیحات دقیق‌تر درباره تعریف شرایط برخورد در ادامه آمده است.

در ادامه داک هرگونه استفاده از این کلمات به تعاریف ارائه شده در این بخش اشاره خواهند داشت.

هنگامی که Entity‌ها به یکدیگر برخورد کنند، یک **مکانیک Impact** فعال می‌شود که از نقطه برخورد، به همه جهت‌ها، موجی از ضربه ساطع می‌کند. طبیعتاً هر چه Entity‌های دیگر، از نقطه برخورد دورتر باشند، اثر این موج ضربه روی آنها کمتر و یا بی‌اثر خواهد بود. به ازای هر Entity در نزدیکی این نقطه برخورد، شما یک بردار اثر به آن نسبت می‌دهید که نهایتاً با بردار حرکت فعلی آن Entity برآیند گرفته می‌شود. توجه کنید اثر دادن این بردار روی بردار حرکت **باید با کنترل حرکت و به صورت smooth** صورت بگیرد.

توجه کنید نیاز نیست موج ضربه را به صورت گرافیکی پیاده‌سازی کنید و این توضیحات صرفا برای توضیح اثر برخورد موج ضربه روی Entity‌های منحرک هستند. همچنین توجه کنید موج ضربه روی فریم‌ها اثرگذار نیست و آنها را حرکت نمی‌دهد.

wasted

## و اما GameOver

شما در ابتدای این بازی 100 واحد HP دارید و هر زمانی و به هر نحوی که این عدد به صفر برسد (HP شما نمی‌تواند منفی شود) شما می‌میرید. (مگر در حالت داشتن Skill های خاص که در قسمت Skill Tree توضیح داده شده‌اند) در این حالت، پنجره‌ای برای کاربر باز می‌شود که Game Over را برای او نمایش می‌دهد. در این پنجره، شما باید XP کسب شده را به کاربر نشان دهید (می‌توانید به سلیقه خود ضریب‌هایی برای این امتیاز کسب شده قائل شوید) و بازگشت به منوی بازی را به کاربر نشان دهید. پس از این، بازی تماماً ریست می‌شود (به استثنای وضعیت Skill Tree) و از ابتدا بازی را آغاز می‌کنید.

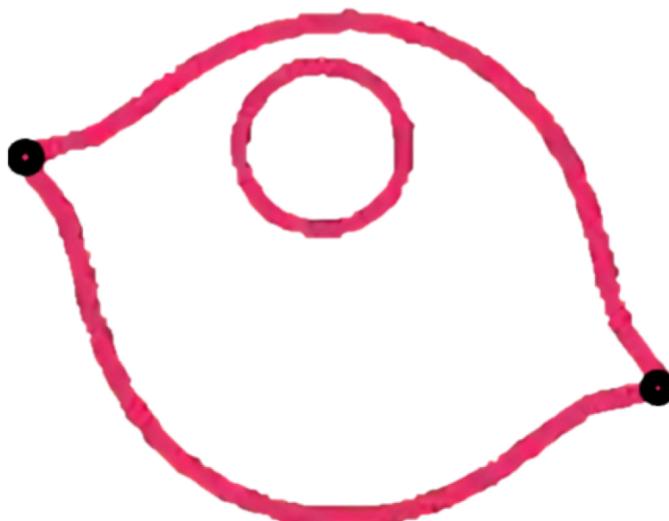
## مکانیک‌های شلیک

همانطور که گفته شد، شما در این بازی به طور ناگهانی مورد حمله قرار گرفته‌اید و به طور کاملاً اتفاقی یک Laser Gun همراه خود دارید (🇺🇸🦅🇺🇸🦅🇺🇸🦅) و می‌توانید با این سلاح از خود در برابر دشمنان و تروجان دفاع کنید. هر تیری که با موفقیت به دشمنان شما اصابت کند، 5 واحد از HP آنها می‌کاهد. اما در دفاع در مقابل تروجان، شما می‌توانید با شلیک به دیواره‌های فریم، فریم را بزرگ‌تر کنید تا فضای بیشتری برای دفاع از خودتان داشته باشید. با اصابت هر تیر به یک دیواره از فریم، فریم باید از سمت همان دیواره به مقداری نسبتاً کم بزرگ‌تر شود و هم‌زمان تمام فریم به مقدار کمی به همان سمت حرکت کند. (توجه کنید این تغییر اندازه باید به صورت smooth صورت بگیرد) همچنین توجه کنید با برخورد تیر با هر Entity و یا دیواره فریم، **مکانیک Impact** فعال می‌شود و تیر از بین می‌رود، اما خود تیرها تحت اثر مکانیک Impact قرار نمی‌گیرند و مسیر حرکت آنها تغییر نخواهد کرد.

## دشمن‌ها (Enemy)

این موجودات که توسط تروجان کنترل می‌شوند در تلاشند تا زنده ماندن را برای شما تا حد ممکن سخت کنند. اکثربیت دشمن‌ها (به انضمام باسفایتها) و حملات زمان‌دار، در فاز بعدی تعریف می‌شوند و در این فاز به یک مدل کلی از دشمن‌ها اکتفا می‌کنیم.

در این بازی هر دشمن یک شکل هندسی است که از تعدادی رأس و تعدادی یال (نه لزوماً پاره خط‌های صاف) تشکیل شده است. اکثر این دشمن‌ها مثل شما مجهز به سلاح‌های دوربرد هستند اما همچنین می‌توانند با کوبیدن راس‌هایشان به شما، آسیب بزنند. در این بازی **راس‌ها نقاط خطرناک هر دشمن و یال‌ها نقاط آسیب‌پذیر آن هستند** **فلذا کلراکتر هرچه ۶۰ تراشند آسیب‌پذیرتر است.** Epsilon بدیهتاً تا وقتی یک دایره است نمی‌تواند به دشمن‌ها از طریق برخورد، آسیب بزند و آسیب پذیرترین کاراکتر ممکن است.



مثالی از پیکربندی دشمن‌ها: نقاط مشکی راس‌های دشمن و خطوط سرخابی یال‌ها هستند

## Melee attack (MA)

به معنای برخورد یک راس از یک Epsilon (در صورت وجود) به یک یال از کاراکتر دشمن (و بالعکس) است. هر Melee attack از طرف Epsilon به اندازه 10 واحد از HP کاراکتری که حمله را متحمل می‌کاهد. قدرت حمله هر یک از دشمن‌های بازی در تعریف آنها ذکر خواهد شد. همه دشمن‌ها به صورت دیفالت (مگر اینکه توضیح جدیدی در این باره داده شود) قادر به انجام این حمله در هر زمانی از بازی خواهند بود.

## Local Routing

این استراتژی، نحوه حرکت دشمن‌ها را بدون اینکه نیازمند پیاده‌سازی یک AI پیچیده باشید با یک فرآیند ساده توضیح می‌دهد:

- دشمن‌ها در برخورد به یکدیگر به هم آسیبی وارد نمی‌کنند اما این برخورددها **مکانیک Impact** را فعال می‌کنند.
- دشمن‌های عادی (تمام دشمن‌های این فاز) در هر لحظه **نزدیک‌ترین مسیر را به سوی Epsilon** طی می‌کنند.
- دشمن‌ها نباید بتوانند از روی یکدیگر عبور کنند. همچنین Overlap برای هیچ کدام از کاراکترهای این فاز **مجاز نیست**.
- **مکانیک Impact** باید در برخورد دشمن‌ها به جلوگیری از گیر کردن آنها در حرکت به سوی شما کمک کنند...

این مکانیک را در طول این فاز و فازهای آتی **Local Routing** می‌نامیم.

## Progress Rewards, a PFL (Positive Feedback loop)

با کشتن هر دشمن تعدادی collectible به صورت نقاطی در محل مرگ آن کاراکتر پخش می‌شود. هر یک از این نقاط در زمان محدودی (مثلًا 10 ثانیه) از بین میروند و Epsilon باید بتواند با عبور از روی آنها، این collectible را بردارد. برداشتن هر یک از این آیتم‌ها مقداری به XP شما اضافه خواهد کرد. مقادیر دقیق موارد گفته شده در مشخصات هر دشمن ذکر خواهد شد.

## Enemy Types

حرکت با کمک Local Routing	collectible 1 دارد که برداشتن آن 5 واحد به XP کاراکتر شما اضافه می‌کند	با هر بار آسیب زدن 6 واحد از Epsilon از HP	10 واحد HP	
حرکت با کمک Local Routing	collectible 2 دارد. برداشتن هر یک 5 واحد به XP کاراکتر شما اضافه می‌کند	با هر بار آسیب زدن 10 واحد از Epsilon از HP	15 واحد HP	

## مکانیک‌های Aggression

در اینجا مکانیک‌های مربوط به رفتار هر یک از دشمن‌ها را توضیح می‌دهیم. دقت کنید اولویت اجرا همواره با خواهد بود و تمام دستورالعمل‌های زیر تنها زمانی اجرا می‌شوند که نیازی به اجرای فرآیند Local Routing نیست.

- **مکانیک‌های مربوط به Quarantine (مربع)**: این کاراکتر در طول حرکت خود به صورت تصادفی گاهاً حرکتی ناگهانی به سمت Epsilon انجام می‌دهد که به آن Dash می‌گوییم. اگر در هنگام Dash، برخوردی با هر Entity دیگری صورت گیرد، همانطور که تاکید شد Dash متوقف شده و اولویت با اجرای Local Routing خواهد بود.
- **مکانیک‌های مربوط به Trigorath ( مثلث )**: این کاراکتر در طول حرکت خود دو وضعیت خواهد داشت. اگر فاصله‌اش از Epsilon زیاد باشد، با سرعت زیاد به سمت او حرکت می‌کند و وقتی فاصله‌اش با او از حد معینی کمتر شد، سرعت او به مقدار کمتری کاهش می‌یابد. جزئیات پیاده‌سازی به سلیقه خودتان خواهد بود.

## به منو باز می‌گردیم!

حال که تعریف اجزای سازنده بازی را به پایان رساندیم، وقت آن است که به بحث منوهای بازی برگردیم و به تکمیل آنها بپردازیم.

### فروشگاه

همانطور که اشاره شد، شما باید دکمه‌ای را به فروشگاه بازی اختصاص دهید تا **تنها وقتی که در حین بازی فشرده شد** بازی را متوقف کرده و پنجره‌ای برای کاربر باز کند. در این پنجره کاربر باید قادر باشد در ازای مقداری از XP جمع شده‌اش، قابلیت‌هایی را **تنها برای همان دست بازی** خریداری کند تا همان لحظه شروع به اثر کنند. (در صورت Game Over شدن تمام این قابلیت‌ها نیز ریست می‌شوند) در این فاز باید قابلیت‌های زیر در این فروشگاه قابل خریدن باشند:

- **O' Hephaestus, Banish**: بازیکن باید بتواند در ازای پرداخت 100 واحد از XP هایش، یک موج ضربه قوی از خود ساطع کند که تمام دشمن‌های اطرافش تا شعاع خاصی را به عقب براند. تعریف دقیق موج ضربه در **مکانیک Impact** آمده است.
- **O' Athena, Empower**: بازیکن با انتخاب این گزینه می‌تواند با پرداخت 75 واحد از XP هایش، موقعتاً به مدت 10 ثانیه با هر شلیک سه تیر به جهت هدف‌گیری شده شلیک کند. جزئیات پیاده‌سازی به سلیقه خودتان خواهد بود.
- **O' Apollo, Heal**: بازیکن باید بتواند در ازای پرداخت 50 واحد از XP هایش، 10 واحد به HP خود اضافه کند.

### Skill Tree

اینجا جایی از منوی اصلی بازی است که شما قابلیت‌های Epsilon را بهبود می‌بخشید. مثل Skill Tree هر بازی دیگری، در اینجا شما XP های خود را در ازای یک قابلیت خاص معاوضه می‌کنید. قابلیت‌های شما در سه دسته **حمله**، **دفاع** و **تغییر شکل** طبقه‌بندی می‌شود و برای باز شدن هر قابلیت باید تمام قابلیت‌های قبلی آن دسته را باز کرده باشید. (چون بهش می‌گیم Tree نه K) اما تنها یک قابلیت را می‌تواند از تمامی دسته‌ها فعال کنید. در این فاز شما تنها سه قابلیت از این درخت را پیاده‌سازی می‌کنید تا موارد دیگر را در فاز بعدی به آنها اضافه کنید. همچنین بازیکن باید بتواند با کلیک روی هر یک از قابلیت‌ها، آنها را به عنوان قابلیت فعال خود انتخاب کند تا با فشردن **کلید مربوط به قابلیت**، آن را در حین بازی فعال کند. هر بار استفاده از این قابلیت 100 واحد XP هزینه خواهد داشت (که باید از XP های همان دست بازی کسر شود) و به مدت 5 دقیقه cooldown به همراه دارد. (به این معنا که تا 5 دقیقه، کاربر نمی‌تواند حتی در صورت داشتن XP کافی، این قابلیت را مجدداً فعال کند)

- **Writ of Ares**: قابلیت اول دسته **حمله**. 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، حملات Epsilon دو واحد بیشتر به دشمن آسیب می‌زنند.
- **Writ of Aceso**: قابلیت اول دسته **دفاع**. با پرداخت 500 واحد XP می‌توانید این قابلیت را باز کنید. در صورت انتخاب این قابلیت به عنوان قابلیت فعال و فعال کردن آن در حین بازی، Epsilon هر ثانیه 1 واحد از HP خود را بازیابی می‌کند.
- **Writ of Proteus**: قابلیت اول دسته **تغییر شکل**. 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، یک راس به Epsilon اضافه می‌شود. تا زمانی که شکل Epsilon دایره است، این راس‌ها روی محیط دایره و با فاصله برابر از هم اضافه خواهند شد.

### Settings

در این بخش از منو شما باید بتوانید حساسیت دکمه‌های مربوط به حرکت، سختی بازی و صدای بازی را در قالب سه slider مختلف تنظیم کنید. کاربر باید بتواند با جلو یا عقب بردن هر یک از این slider ها حساسیت و صدا را کم و زیاد کند. توضیحات مربوط به سختی بازی در بخش بعدی آمده است. می‌توانید در این منو این بخش امتیازی را پیاده‌سازی کنید.

## مکانیک‌های wave و spawn مختلف حملات

در نهایت وقتی تمام بیس گرافیکی و لاجیک فاز اول را پیاده‌سازی کردید، نیاز است به طور خودکار دشمن‌هایی را در صفحه اضافه کنید تا چالش اصلی بازی را برای بازیکن فراهم کنند. در فازهای بعدی کامل‌تر به این موضوع خواهیم پرداخت اما فعلًا در این فاز، از شما انتظار می‌رود حملات بازی را در سه برنامه‌ریزی کنید. جزئیات پیاده‌سازی کاملاً به سلیقه شما واگذار می‌شود اما طبیعتاً منطقیست که wave های بعدی تعداد دشمن‌های بیشتری را به سمت **Epsilon** بفرستد تا کار را برای او سخت‌تر کند. اما از طرف دیگر توجه داشته باشید تعداد بیش از حد دشمن در یک صحنه، هم از سویی پروفورمنس بازی را تحت الشعاع قرار می‌دهد و هم احتمالاً تجربه مفرح بازی شما را از بازیکن خواهد گرفت بنابراین سعی کنید با آزمون و خطای مقادیر مناسبی را به دست آورید.

### سختی بازی

در این قسمت (که از طریق منوی Settings کنترل می‌شود) شما باید سطح سختی بازی خود را کنترل کنید. سطح سختی می‌تواند با افزایش تعداد دشمن‌های هر wave، بالا بردن سرعت و یا قدرت دشمن‌ها و یا کوچکتر کردن آنها (به هدف سخت‌تر کردن مورد اصابت قرار دادن آنها) باشد. جزئیات پیاده‌سازی مجدداً به عهده خودتان خواهد بود.

### پایان بازی

در نهایت با موفقیت از تمام سه wave حملات دشمن جان سالم به در می‌برد و کنترل **این فریم** را به دست می‌گیرد. در این لحظه شما باید صحنه پایانی بازی را طراحی کنید که در آن **Epsilon** در همان نقطه‌ای که هست بزرگ و بزرگتر می‌شود تا تمام فریم را فرا بگیرد. سپس فریم کوچک می‌شود تا اندازه آن به صفر برسد. اما **Epsilon** و بقیه شکل‌ها به کجا می‌روند؟ فاز بعدی منتظر شماست... پس از این صحنه پنجه پایانی که در **این قسمت** تعریف شده را به کاربر نشان می‌دهید و بازی به پایان می‌رسد.

### ابزارهایی که از شما انتظار می‌رود در توسعه کدتان از آنها استفاده کنید

#### استفاده از ابزار گیت

برای توسعه پروژه خود شما باید از ابزارهای version control مثل Git استفاده صحیح از Git برای توسعه این پروژه در هر سه فاز، اجباری خواهد بود. برای کسب نمره این بخش لازم است موارد زیر را رعایت کنید:

- یک ریپازیتوری پرایوت داشته باشید و نسخه اولیه پوش شده مربوط به نسخه فاز یک پروژه باشد
- برای هر فاز پروژه یک Branch و یک Development Branch داشته باشید
- در طول توسعه پروژه در Git، حتماً کامیت‌های منظم داشته باشید و توضیحات کامیت را بنویسید

#### استفاده از Build Tool (میون و گریدل)

پروژه شما باید شامل یک کانفیگ کامل از ابزارهای Maven، Gradle باشد به طوری که فقط با استفاده از سورس پروژه بتوان آن را بیلد و اجرا کرد. به این معنا که کتابخانه‌های مورد استفاده در پروژه به طور خودکار دانلود و استفاده شوند. استفاده از ابزارهای مشابه مثل Apache Ant بلامانع است.

همچنین در اینجا شما می‌توانید قسمت امتیازی این بخش را پیاده‌سازی کنید.

## بخش‌های امتیازی!

### دوران اپسیلون!

به طور خلاصه، در این بخش شما باید همزمان با حرکت موس، Epsilon را هم به سمت او بچرخانید! این فیچر به خصوص هنگامی که قابلیت **Writ of Proteus** را فعال کرده باشید و همچنین در فازهای آتی کاربرد ویژه‌ای دارد و قابل تشخیص خواهد بود.

### تنظیمات مربوط به دکمه‌های بازی

در این بخش شما باید یک گزینه برای تغییر Key Binding در منوی تنظیمات داشته باشید. با کلیک روی این گزینه، یک صفحه باز خواهد شد که دکمه‌های تشخیص داده شده به هر یک از عملیات‌ها را نمایش دهد. علاوه بر این، کاربر باید قادر باشد در این صفحه کلیدهای مربوط به هر یک از **اعمال ذکر شده در ابتدای داک** را تغییر دهد. کاربر برای تغییر هر یک از این دکمه‌ها، روی عملیات موردنظر کلیک می‌کند و سپس برنامه شما باید منتظر فشرده شدن یک کلید باشد تا عملیات را به آن کلید نسبت دهد. همچنین در این بین باید دکمه‌ای برای لغو این عملیات موجود باشد. (ترجیحا Esc) در نهایت دقت کنید باید تکراری نبودن دکمه جدید را قبل از ثبت چک کنید و در صورت تکراری بودن اوری را به کاربر نمایش دهید.

### پیاده‌سازی گشتاور دورانی

در این بخش، شما باید گونه‌ای از گشتاور دورانی را پیاده‌سازی کنید. وضعیتی را متصور شوید که کاراکتر شما به یک یال از یک **Trigorath** برخورد می‌کند. طبق مکانیک Impact شما باید یک موج ضربه دقیقاً از این نقطه برخورد ساطع کنید که همه Entity‌ها اعم از **Epsilon** را به عقب براند. اما در این بخش شما همچنین باید گشتاور دورانی حاصل از این ضربه را نیز اعمال کنید که باعث می‌شود **Trigorath** تا مدت زمان معینی در حین حرکات خود به سمت **Epsilon**، حول مرکز نقل خود در دوران باشد. مثال و توضیحات کامل در این زمینه در [این لینک](#) آمده است.

### Broad/Narrow Search

مسئله Collision Detection یکی از مسائل بنیادی در پیاده‌سازی تمام بازی‌های ویدیوییست. این مسئله نحوه تشخیص برخورد و آجکت گرافیکی را با روش‌هایی بهینه حل می‌کند. با اینکه شما در این فاز نیازمند رویه‌های پیشرفته Collision Detection نیستید (جون تمام اشکال شما دایره، مربع و مثلث هستند) اما مسئله دیگری که در اینجا حائز اهمیت است، لود محاسباتی شماماست. به طور خلاصه در طول یک پیاده‌سازی ساده، اگر شما در آن واحد، 20 شکل مختلف در فریم بازی داشته باشید باید هر لحظه  $\frac{1}{2}$  احتمال برخورد را بررسی کنید. با توجه به اینکه برای بررسی دقیق هر برخورد هم تا 16 برخورد مختلف از پاره خط‌ها را باید چک کنید، این اعداد با پیچیده‌تر شدن اشکال هندسی و بالاتر رفتن تعداد شکل‌های موجود در صفحه سر به فلک خواهد کشید که ممکن است باعث وقوع مشکلات پرفورمنس از قبیل لگ و FPS پایین شوند. برای حل این مشکل، یکی از حربه‌های رایج، تقسیم این لود گرافیکی و سرشکن کردن هزینه زمانی این محاسبات است و یکی از رویه‌های انجام این فرآیند استفاده از Broad/Narrow Search است که "مسئله بررسی برخوردها" را به مسئله "بررسی برخوردهای آجکت‌های نزدیک به هم" تقلیل می‌دهد. در این فرآیند مقداری زیادی از لود محاسباتی که ناشی از بررسی برخورد آجکت‌هایی بود که به اندازه کافی از هم دور بوده‌اند (و مطمئناً نمی‌توانستند با هم برخوردی داشته باشند) از بین می‌روند. برای یادگیری کامل درباره این فرآیند به [این منبع](#) مراجعه کنید.

### استفاده از ابزار JLINK

برای دریافت امتیاز این بخش شما باید با استفاده از Build Tool JLINK مورد نظر خود و ابزار JLINK بتوانید از پروژه، یک فایل قابل اجرا دریافت کنید که روی هر سیستم عامل مشابهی بدون نیاز به نصب جاوا یا آماده‌سازی دیگری اجرا شود. توضیحی کامل از این ابزار در [این لینک](#) مورد بحث قرار داده شده است. همچنین برای یادگیری کامل در مورد این ابزار به [این لینک](#) مراجعه کنید.

### سخن پایانی

همانطور که احتمالاً می‌دانید، فرآیند تحویل تمام تمرینات و بالاخص پروژه با رویه‌ای به نام "ارزیابی" همراه است که در آن از شما انتظار می‌رود به تمام قسمت‌های کد خود نسلط کامل داشته باشید و بتوانید در حین تحویل، قسمت‌هایی از کد خود را به صورت جزئی یا کلی تغییر دهید. در غیر این صورت، تشخیص بر تقلب خواهد بود و مستقل از پیاده‌سازی شما و کامل بودن مابقی کد، **نمره بخش مربوطه صفر منظور خواهد شد**. لذا خواهشمندیم از هرگونه تقلب پرهیز کنید. استفاده **جزئی** از ابزارهای هوش مصنوعی و کمی کردن کد از منابع اینترنتی بلامانع است اما همچنان از شما انتظار می‌رود به عملکرد هر قسمی از کد خود کاملاً تسلط داشته باشید تا بتوانید فرآیند ارزشیابی را با موفقیت پشت سر بگذارند.

Next JFrames await  
To Be Continued...

با آرزوی موفقیت و سر بلندی شما  
تیم درس برنامه نویسی پیشرفته

