# برنامه‌نویسی پیشرفته

## زمستان 1403 - بهار 1404

## فاز اول پروژه


تیم تعریف و طراحی پروژه:

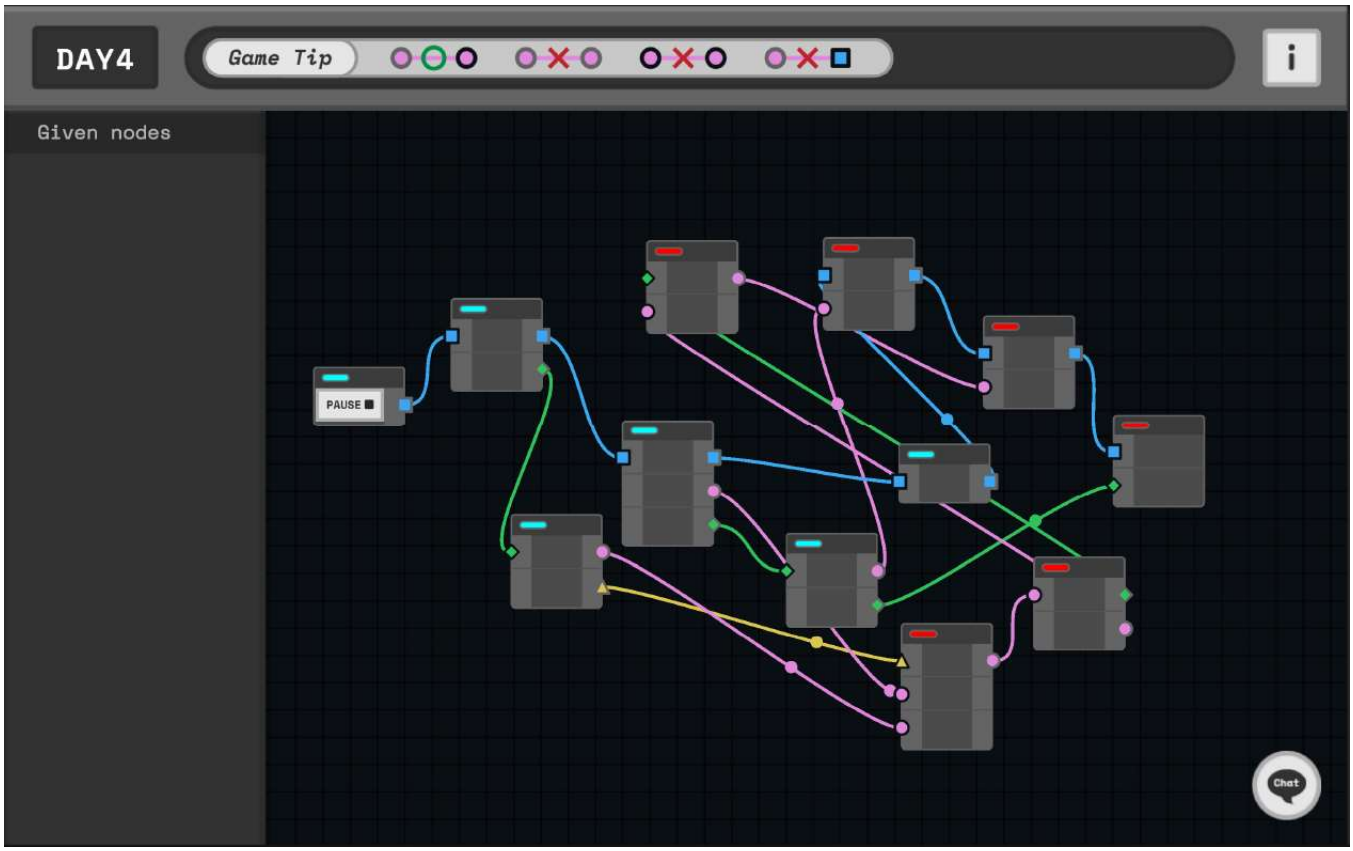
آرین همتی

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 شالودهٔ گرافیکی 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، پیاده‌سازی حرکت و مدل‌سازی اِلمان‌های 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 مختلفی 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 POJO (Plain Old Java Object) نزدیک 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 درس در این لینک فهرست شده و منابعی تکمیلی برای یادگیری این مطالب در کنار روند درسی در این لینک ضمیمه شده است.

**نکته بسیار مهم: تعیین مقادیر ثابت مناسب، به نحوی که تجربه معقولی از بازی به کاربر القا 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 کاراکتر، اندازه فریم، مقادیر تغییر اندازه فریم، collectible ها و سایر موارد پرهیز کنید تا بازی شما در فرآیند تحویل و ارزیابی با مشکلات اساسی مواجه نشود.**



**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 فلذا حتما 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 (ادامه از آخرین مرحله)
- مراحل بازی
- تنظیمات بازی
- خروج از بازی

کارکرد همهٔ این گزینه‌ها به تفضیل در ادامهٔ 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 و یا انتخاب مرحله بازی از طریق گزینه مراحل، منو بسته شده و **تمام برنامه‌های باز باید وارد حالت minimized شوند.** سپس تمام سیستم‌های مرحله به همراه پورت‌های ورودی و خروجی‌شان به صورت اشکال مستقل از هم در صفحه بازی قرار گرفته و بازی آغاز می‌شود. دقت کنید **تمام حرکات پیاده‌سازی شده در این بازی باید به صورت پیوسته (smooth) انجام بگیرند.** فریم بازی نباید قابل جابجایی و تغییر سایز (همچنین maximize، minimize شدن) و بسته شدن باشد.

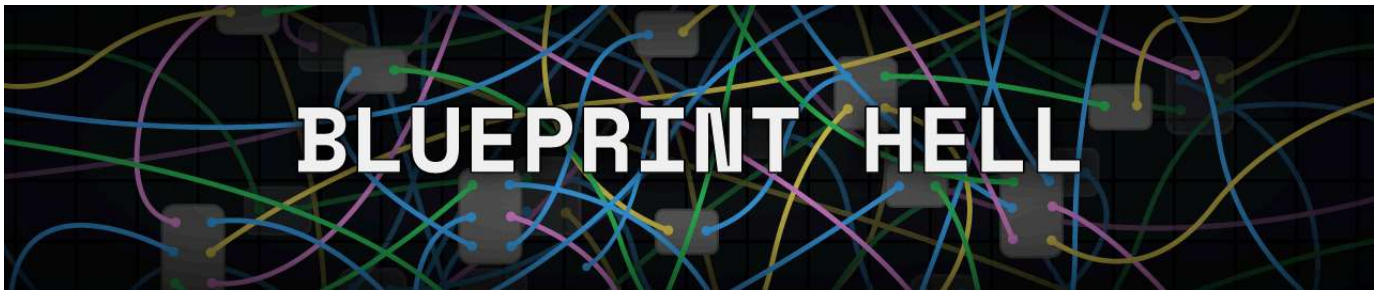
### دکمه‌های بازی

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

- **پیش‌روی زمان:** شما باید دو دکمه برای حرکت زمان به جلو و عقب اختصاص دهید. (استفاده از Arrow Key ها پیشنهاد می‌شود) فشار دادن همزمان دکمه‌های مربوط به حرکت زمان به جلو و عقب تغییرات زمانی شود. مکانیک‌های مربوط به نحوه و اثر پیش‌روی زمان در ادامهٔ داک توضیح داده خواهد شد.

- **سیم‌کشی:** شما باید با بردن نشانگر ماوس روی یکی از پورت های خروجی و بردن آن روی پورت ورودی یک سیستم دیگر در حال فشردن این کلید، یک سیم بین این دو پورت ایجاد کنید. همچنین می‌توانید **این قسمت امتیازی** را پیاده‌سازی کنید.

- **فروشگاه:** در این بازی شما باید دکمه‌ای مربوط به فروشگاه تعبیه 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.




## محیط بازی (Game Env.)

### اطلاعات بازی (HUD)

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 دکمه ای خاص، به مدت محدودی (مثلاً 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:

اصطلاح HUD یا Heads-up display جزوی از رابط کاربری (UI) یک بازی است که حاوی اطلاعاتی درباره وضعیت و داده‌های بازی‌ست. طراحی یک HUD مناسب برای بازی می‌تواند در ایجاد یک تجربه منحصر به فرد برای بازیکن نقش بزرگی ایفا کند. برای مثال می‌توانید HUD بازی‌های سری Witcher را در طی تکامل سری مقایسه کنید و به اثر آنها بر روی تجربه کاربری دقت کنید. برای یادگیری بیشتر درباره طراحی‌های کارا برای HUD بازی (فارغ از بحث پروژه و صرفاً در جهت علاقه به موضوعات بازی‌سازی) می‌توانید صرفا روی این باکس کلیک کنید!

- Remaining Wire Length
- Temporal Progress
- Packet Loss
- Coins

### صدا

بازی شما باید دارای صدای پس زمینه باشد. همچنین قسمت‌های اتصال موفق دو دستگاه، پایان هر مرحله، آسیب دیدن هر پکت، باید دارای صداها و آهنگ‌های پس زمینه منحصر به فرد باشند. (تعاریف مربوط به هر یک از این کلمات در ادامه داک آمده است)

### ولی من کیستم؟...

شما نقش یک اپراتور شبکه را بازی می‌کنید که باید تعدادی پکت را بین دستگاه‌های موجود در شبکه انتقال دهید! شاید این توصیف کمی ساده به نظر برسد ولی توزیع و پردازش پکت بین تعداد زیادی سیستم وقتی که فقط مقدار محدودی سیم برای ایجاد اتصالات دارید کار دشواری خواهد بود. وظیفه شما این است که شبکه‌ای طراحی کنید که داده‌ها به سلامت به سیستم‌های شبکه برسند.

### روغن‌کاری محیط گرافیکی...

دریافت FPS بالا و روان‌تر به نظر رساندن بازی، معضلی به قدمت Tennis for Two (در سال 1958) بوده است. از این رو، چون ذات محاسبات کامپیوتری گسسته است، بازی‌سازها و برنامه‌نویس‌ها دائماً در طول تاریخ در پی حربه‌هایی بر روان و پیوسته به نظر رساندن این فرآیندهای گسسته بوده‌اند. این حربه‌ها؛ ایده‌های هوشمندانه‌ای مثل Anti-Aliasing، Ray-Tracing 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) همچنین نقلی معروف از توسعه‌کننده‌های بازی وجود دارد که 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 پکت‌ها روی اتصالات شبکه استفاده کنید و همچنین راهکاری برای یکنواخت نشان دادن حرکت پکت‌ها روی اتصالات بین سیستم‌های مختلف طراحی کنید. (به دلیل اینکه همان‌طور که در فاز بعد خواهید دید، اتصالات بین دستگاه‌ها لزوماً خطوط صاف نیستند) منبعی سودمند برای یادگیری این موضوع (و موضوعاتی جامع‌تر در کنترل حرکت) در اینجا آمده است.

## Packet Loss (Death Mechanics)

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

### مکانیک‌های 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 یک پکت روی یکی از اتصالات بردار حرکت مربوط به آن پکت می‌گوییم.
- برخورد: به هرگونه تماس دو پکت با هم اطلاق می‌شود و هر برخورد باعث ایجاد مقدار مشخصی نویز در هر پکت می‌شود.

- **آسیب:** اگر یک پکت در حال عبور از اتصالات، از سیم منحرف شده و از مسیر سیم خارج شود و یا میزان نویز آن از اندازه‌اش بیشتر شود، آن پکت از دست خواهد رفت و به اندازه آن پکت به مقدار Packet Loss افزوده خواهد شد.

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

هنگامی که پکت‌ها به یکدیگر برخورد کنند، یک **مکانیک Impact** فعال می‌شود که از نقطهٔ برخورد، به همه جهت‌ها، موجی از ضربه ساطع می‌کند. طبیعتاً هر قدر پکت‌های 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.** اگر این موج ضربه به اندازه کافی به یک پکت نزدیک باشد، آن پکت از روی سیم در حال گذر منحرف خواهد شد و از دست خواهد رفت.

**توجه کنید نیاز نیست موج ضربه را به صورت گرافیکی پیاده‌سازی کنید و این توضیحات صرفاً برای توضیح اثر برخورد موج ضربه هستند. همچنین توجه کنید موج ضربه روی دستگاه‌ها و یا پکت‌های درون یک دستگاه اثرگذار نیست و آنها را حرکت نمی‌دهد.**

### و اما GameOver

**شما در ابتدای هر مرحله باید تعداد مشخصی پکت را برای زمان مشخصی در شبکه به جریان در آورید** و هر زمانی و به هر نحوی که میزان Packet Loss شما از 50 درصد بیشتر شود، شبکه طراحی شده شما معیوب است و مرحله را خواهید باخت. (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 تعداد پکت‌های سالم موجود در شبکه و میزان Packet Loss 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، مرحله تماماً ریست می‌شود (به استثنای وضعیت Skill Tree) و از ابتدا بازی را آغاز می‌کنید.

## مکانیک‌های مربوط به پکت‌ها (Packet Mechanics)

این موجودات که توسط بازی کنترل می‌شوند دائماً در شبکه شما در حال حرکت خواهند بود. اکثریت انواع پکت‌ها و حرکات و پروتکل‌های خاص مربوط به آنها در فاز بعدی تعریف می‌شوند و در این فاز به یک مدل کلی از پکت‌ها اکتفا می‌کنیم. در این بازی هر پکت یک شکل هندسی است که مکانیک‌های حرکتی مخصوص به خودش را دارد. در این فاز تنها دو نوع پکت به شکل‌های مربع و مثلث وجود دارند. همچنین در این فاز اتصالات تنها به صورت خطوط صاف بین پورت‌های مدنظر هستند. نحوه کار یک سیستم به این صورت است که هر پکت را از یک پورت ورودی دریافت کرده و بلافاصله از یک پورت خروجی خارج می‌کند. برای انتخاب پورت خروجی یک پکت، در صورتی که یک پورت سازگار وجود داشته باشد که هیچ پکتی روی سیم متصل به آن قرار نداشته باشد، (اصطلاحاً می‌گوییم این پورت خالی‌ست) اولویت با این پورت خواهد بود. در غیر این صورت به طور تصادفی یک پورت خالی برای خارج کردن پکت انتخاب می‌شود. در صورتی که هیچ یک از پورت‌های خروجی دستگاه خالی نباشند، پکت در دستگاه ذخیره می‌شود تا یک پورت خالی در دسترس قرار بگیرد. ظرفیت ذخیره‌سازی یک سیستم 5 پکت است.

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

با ورود هر پکت به یک سیستم تعدادی سکه به شبکه داده می‌شود که می‌توان آنها را برای خرید Skill ها و آیتم‌های فروشگاه استفاده کرد. همچنین در فاز بعدی برای مکان‌های قرارگیری اتصالات دستگاه‌ها امتیازاتی در قالب پاورآپ منظور خواهد شد. همچنین توجه کنید منظور از سازگاری یک پکت با یک پورت، هم شکل بودن آنها خواهد بود. به این معنا، پورت‌های مربعی با پکت‌های مربعی سازگار و با پکت‌های مثلثی ناسازگار خواهند بود. (مشابهاً برای پورت‌هایی که در فاز بعد معرفی خواهند شد)

| | | | |
|---|---|---|---|
|  | اندازه: 2 واحد | با هر بار ورود به سیستم 1 سکه به شبکه اضافه می‌کند | سرعت حرکت با شروع از یک پورت سازگار نصف سرعت حرکت آن با شروع حرکت از یک پورت ناسازگار است. حرکت آن در هر یک از پورت‌ها با سرعت ثابت انجام می‌گیرد. (مگر در اثر Impact) |
|  | اندازه: 3 واحد | با هر بار ورود به سیستم 2 سکه به شبکه اضافه می‌کند | حرکت آن با شروع حرکت از پورت‌های سازگار با سرعت ثابت است (مگر در اثر Impact) اما حرکت آن در عبور از یک پورت ناسازگار شتاب‌دار خواهد بود |

### سیستم‌ها (Network Systems)

هر سیستم یک واحد شبکه است که به شکل یک مستطیل در فریم بازی نمایش می‌دهد. هر سیستم یک اندیکاتور در بالای آن دارد که در ابتدای مرحله خاموش است. همچنین هر سیستم تعدادی پورت ورودی در سمت چپ و تعدادی پورت خروجی در سمت راست خود دارد. (مثالی از طراحی یک سیستم در بازی در تصویر ابتدای داک آمده است) شما باید بتوانید با بردن نشانگر ماوس روی یک پورت خروجی، نگه داشتن کلید مربوط به سیم‌کشی و بردن نشانگر به پورت ورودی یک سیستم دیگر، به شرطی که طول سیم به اندازه کافی موجود باشد یک خط صاف بین این دو پورت رسم کند. این سیم یک مسیر برای انتقال پکت‌ها از یک سیستم به سیستم دیگر است. پورت‌ها در این فاز صرفا مربعی و مثلثی هستند و در فاز بعد پورت‌های متفاوتی معرفی خواهند شد. دقت کنید امکان ایجاد اتصال بین پورت ورودی و خروجی یک سیستم وجود ندارد. در فازهای بعد شرایط دیگری هم به شبکه اضافه خواهند شد.

### سیستم‌های مرجع

سیستم‌های مرجع سیستم‌های خاصی هستند که پکت‌ها را از پورت‌های ورودی خود به پورت‌های خروجی انتقال نمی‌دهند. هر سیستم مرجع تعداد مشخصی پکت را با فرکانس مشخصی (که در هر مرحله برای هر سیستم تعیین می‌شوند) از طریق پورت‌های خروجی وارد شبکه می‌کند و این پکت‌ها از طریق شبکه طراحی شده به سیستم‌های دیگر منتقل می‌شوند. وظیفه شما به عنوان اپراتور این است که با استفاده از تمام سیستم‌های موجود و طول سیم محدود داده شده شبکه‌ای طراحی کنید که هر پکت (که از پورت خروجی یک سیستم مرجع وارد شبکه شده است) در نهایت به یک سیستم مرجع بازگردد و Packet Loss محدود باشد.

### اتصالات سیستم

همانطور که پیش‌تر هم اشاره شده است، اتصالاتی که بین سیستم‌های شبکه ایجاد می‌شود با استفاده از طول سیم محدودی انجام خواهند شد و در نتیجه شما باید دائماً طول سیم مصرف شده (Wire Length) برای ایجاد شبکه را محاسبه کنید. (این کار در فازهای بعدی که اتصالات خطوط صاف نیستند چالش جدی‌ای خواهد بود) در شبکه طراحی شده هر دستگاه باید به هر دستگاه دیگر دسترسی داشته باشد و در نتیجه باید گراف تشکیل شده لزوماً همبند باشد. همچنین شبکه شما باید تمام پورت‌ها را پر کند.

### پیش‌روی زمان (Temporal Progress)

همانطور که اشاره شد، علاوه بر محدودیت‌های شبکه طراحی شده در بازی، شما باید جلوی از دست رفتن پکت‌ها را هم بگیرید. به این منظور شما باید یک نوار زمانی در بازی خود تعبیه کنید که با جلو و عقب بردن آن، وضعیت پکت‌ها را در زمان‌های متفاوت در شبکه خود ببینید. در این وضعیت شما باید بتوانید اتصالات خود را تغییر دهید و پس از رسیدن به یک شبکه معتبر، وضعیت پکت‌های شبکه جدید را در این زمان ببینید. این قابلیت به خصوص در فاز بعدی (که قابلیت تغییر شکل اتصالات سیستم‌ها وجود دارد) به شما برای گذراندن مراحل کمک خواهد کرد. در نهایت شما باید گزینه‌ای برای اجرای شبکه خود داشته باشید. با فشردن

دکمه اجرا، پکت‌ها به صورت برنامه‌ریزی شده از سیستم‌های مرجع وارد شبکه خواهند شد و به جریان در خواهند آمد. در صورتی که پکت‌های ارسالی با Loss کمتر از 50 درصد به سیستم‌های مرجع برگردند، مرحله با موفقیت پشت سر گذاشته می‌شود. در این موقعیت شما باید اطلاعات مرحله را به بازیکن نشان دهید و گزینه‌ای نیز برای بازگشت به منو یا رفتن به مرحله بعد طراحی کنید.

## به منو باز می‌گردیم!

### 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 بتواند در ازای مقداری از سکه‌هایش، قابلیت‌هایی را تنها برای همان مرحله خریداری کند تا همان لحظه شروع به اثر کنند. در این فاز باید قابلیت‌های زیر در این فروشگاه قابل خریدن باشند:

- **O' Atar:** بازیکن باید بتواند با پرداخت 3 سکه، اثر موج‌های Impact را به مدت 10 ثانیه غیرفعال کند.
- **O’ Airyaman:** بازیکن باید بتواند با پرداخت 4 سکه، برخورد پکت‌های موجود در شبکه را برای 5 ثانیه غیرفعال کند.
- **O' Anahita:** بازیکن باید بتواند در ازای پرداخت 5 سکه، نویز تمام پکت‌های حاضر در شبکه را صفر کند.

### Settings

شما باید بتوانید صدای بازی را در قالب یک slider تنظیم کنید. می‌توانید در این منو این بخش امتیازی را پیاده‌سازی کنید.

## طراحی مراحل مختلف بازی

در نهایت وقتی تمام بیس گرافیکی و لاجیک فاز اول را پیاده‌سازی کردید، نیاز است مراحلی برای بازی طراحی کنید. در این فاز تنها کافیست 2 مرحله متفاوت برای بازی خود طراحی کنید. توجه کنید که انتظار می‌رود مراحل طراحی شده از کیفیت مناسب برخوردار باشند به نحوی که مشخصه‌های متفاوت بازی در آن قابل مشاهده باشد و بتوان قوانین مختلف بازی را در آنها آزمایش کرد.

## 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 کنید. 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 Gradle، Maven 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!

### سیم‌کشی حرفه‌ای!

در این بخش شما باید بتوانید سیم‌ها را در لحظه به کاربر نشان دهید! پس از انتخاب پورت خروجی یک سیستم و نگه داشتن کلید مربوط به سیم‌کشی، با حرکت نشانگر ماوس به هر نقطه از صفحه بازی، سیم متصل کننده پورت مربوط به آن نقطه را نمایش دهید. همچنین شما می‌توانید ممکن بودن یا نبودن این سیم‌کشی با توجه به قوانین بازی را با در نظر گرفتن یک رنگ برای سیم نمایش دهید. برای مثال، تا وقتی که طول سیم باقی‌مانده برای انجام این سیم‌کشی کافی باشد، سیم را با رنگ سبز در صفحه نمایش دهید و در صورتی که کاربر یک پورت ورودی را انتخاب کند، سیم‌کشی با موفقیت انجام خواهد شد. اما در صورتی که ماوس روی یک پورت ورودی از همان سیستم قرار بگیرد، سیم‌کشی مجاز نخواهد بود و در نتیجه سیم به رنگ قرمز در خواهد آمد. توجه کنید در نظر گرفتن این رنگ‌ها جزو انتظارات این بخش نیست و صرفاً یک پیشنهاد است.

### 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 تخصیص 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 اروری را به کاربر نمایش دهید.

### Broad/Narrow Search

مسئله Collision Detection یکی از مسائل بنیادی در پیاده‌سازی تمام بازی‌های ویدیوییست. این مسئله نحوه تشخیص برخورد دو آبجکت گرافیکی را با روش‌هایی بهینه حل می‌کند. با اینکه شما در این فاز نیازمند رویه‌های پیشرفته Collision Detection نیستید (چون تمام اشکال شما دایره، مربع و مثلث هستند) اما مسئله دیگری که در اینجا حائز اهمیت است، لود محاسباتی شماست. به طور خلاصه در طول یک پیاده‌سازی ساده، اگر شما در آنِ واحد، 20 شکل مختلف در فریم بازی داشته باشید باید هر لحظه $\binom{20}{2}$ احتمال برخورد را بررسی کنید. با توجه به اینکه برای بررسی دقیق هر برخورد هم تا 16 برخورد مختلف از پاره‌خط‌ها را باید چک کنید، این اعداد با پیچیده‌تر شدن اشکال هندسی و بالاتر رفتن تعداد شکل‌های موجود در صفحه سر به فلک خواهند 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 آبجکت‌های نزدیک به هم" تقلیل می‌دهد. در این فرآیند مقداری زیادی از لود محاسباتی که ناشی از بررسی برخورد آبجکت‌هایی بود که به اندازه کافی از هم دور بوده‌اند (و مطمئناً نمی‌توانستند با هم برخوردی داشته باشند) از بین می‌رود. 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 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 تسلط 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 کپی کردن کد از منابع اینترنتی بلامانع است اما همچنان از شما انتظار می‌رود به عملکرد هر قسمتی از کد خود کاملا تسلط داشته باشید تا بتوانید فرآیند ارزشیابی را با موفقیت پشت سر بگذرانید.


با آرزوی موفقیت و سربلندی شما
تیم درس برنامه‌نویسی پیشرفته
دانشگاه صنعتی شریف