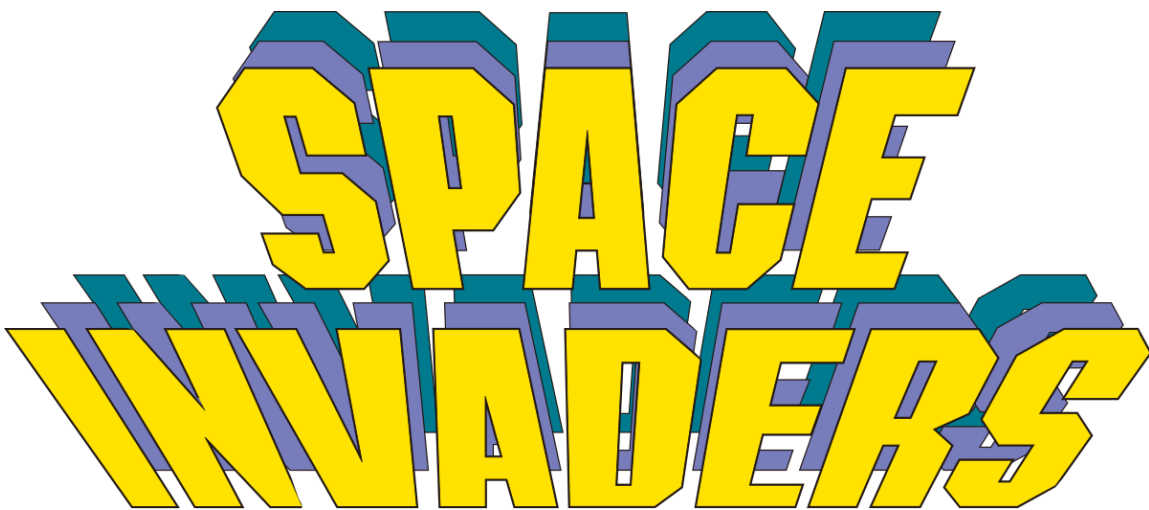




Basics of Programming Final Project

Winter 2025

Course Instructor: Dr. Saeed Reza Kheradpisheh



معرفی پروژه

سلام دوستان! امیدواریم حالتان خوب باشد و ترم خوبی را سپری کرده باشید.

پروژه این ترم شما بازی Space invaders (مهاجمان فضایی) است که اولین بار در سال ۱۹۷۸ در قالب یک بازی آرکید معرفی شد که به سبک fixed shooter بوده و جزو اولین بازی ها در این ژانر می باشد. در ادامه، بیشتر با نحوه بازی آشنا خواهیم گشت.

از اهداف انجام این پروژه دانستن تسلط کامل شما روی تمام مباحث مبانی برنامه سازی و مواجهه با یک پروژه بازی سازی و ایجاد همکاری برای انجام دادن یه کار گروهی است.

پیشنهاد می کنیم که با توجه به حجم پروژه، این کار را در تیم های دو نفره انجام دهید ولی انجام پروژه به صورت تک نفره هم مجاز است...

در حین انجام پروژه اگر جایی به مشکل یا سوالی برخوردید می توانید از کمک منتور پروژه تان استفاده کنید.

توضیحات بازی

شما کنترل یک سفینه ی فضایی کوچک در پایین صفحه را در اختیار دارید. دشمنان، به صورت گروهی از مهاجمان فضایی در بالای صفحه ظاهر می شوند و به صورت منظم به سمت چپ یا راست حرکت می کنند تا به مرور زمان به پایین برسند.



سفینه بازیکن و مهاجمان فضایی دارای قابلیت شلیک کردن هستند. در ابتدای بازی، موانعی به عنوان سنگر بین سفینه و مهاجمان قرار دارد که در صورت برخورد تیر با آنان، دچار آسیب می شوند تا زمانی که کاملاً نابود گردند. در زمان های تصادفی میان بازی، یک بشقاب پرنده قرمز مهاجم در بالای مهاجمان دیگر به پرواز در می آید که نابود کردن آن توسط بازیکن امتیاز بیشتری نسبت به نابود کردن مهاجمان معمولی دارد.

اجزای بازی

- سفینه‌ی بازیکن



در پایین صفحه قرار دارد و توسط بازیکن کنترل می شود.

- مهاجم فضایی نوع یک



در دو ردیف پایین قرار دارد و نابود کردن آن ۱۰ امتیاز دارد.

- مهاجم فضایی نوع دو



در ردیف وسط قرار دارد و نابود کردن آن ۲۰ امتیاز دارد.

- مهاجم فضایی نوع سه



در ردیف بالا قرار دارد و نابود کردن آن ۴۰ امتیاز دارد.

- بشقاب پرنده قرمز



در زمان های تصادفی میان بازی، در ردیفی بالاتر از همه ی مهاجمان وارد می شود و نابود کردن آن ۱۰۰ امتیاز دارد.

- سنگرها



بین سفینه بازیکن و مهاجمان هستند و هم نقش دفاعی برای بازیکن را دارد و هم مانع شلیک مستقیم او می شود. در صورت برخورد تیر می شکنند و به تدریج از بین می روند.

قوانین بازی

- حرکت سفینه

بازیکن می‌تواند سفینه‌ی خود را تنها در دو جهت افقی (چپ و راست) حرکت دهد.

- شلیک

بازیکن می‌تواند گلوله‌هایی به سمت دشمنان شلیک کند.

قبل از شلیک گلوله‌ی بعدی باید صبر کرد تا گلوله قبلی از صفحه خارج شده یا به مهاجمی برخورد کند.

- مهاجمان

مهاجمان به صورت گروهی حرکت می‌کنند و با رسیدن به لبه‌ی صفحه تغییر جهت داده و یک ردیف پایین‌تر می‌آیند.

هرچه تعداد دشمنان کاهش یابد، سرعت حرکت آن‌ها بیشتر می‌شود.

در لحظه‌های تصادفی از بازی هر کدام از مهاجمان می‌توانند موشکی به سمت پایین پرتاب کنند.

- مانع‌ها

هر مانع یک ظرفیت سختی دارد که در زیر آن نمایش داده می‌شود و پس از اتمام ظرفیت، آن مانع نابود می‌شود.

- امتیازگیری

هر دشمن دارای امتیاز مشخصی است که با نابودی آن‌ها به بازیکن تعلق می‌گیرد.

- باخت بازی

اگر دشمنان به پایین صفحه برسند یا سفینه‌ی بازیکن نابود شود، بازی تمام می‌شود.

بازیکن تعداد محدودی جان در اختیار دارد.

- سطوح بازی

با نابودی تمام دشمنان، بازیکن به سطح بعدی می‌رود که معمولاً چالش‌برانگیزتر و سریع‌تر است.

لینک‌ها و منابع برای آشنایی بیشتر با بازی:

- [نمونه از بازی به صورت آنلاین](#)
- [ویدیو گیم پلی از بازی](#)
- [فونت برای کاراکترهای داخل بازی](#)

لیست فیچرها

در اینجا لیستی از تمام فیچرهای اجباری که باید پیاده‌سازی کنید مشاهده می‌کنید:

- زمانی که بازی رو اجرا می‌کنیم باید منوی اصلی (شامل بخش های زیر) به یوزر نمایش داده شود:
 - New Game
 - Load Game
 - How to Play
 - Leaderboard
 - Exit
- بازی باید شامل **حداقل دو سطح سختی** متفاوت باشد. اینکه چجوری بازی را سخت‌تر کنید را خودتان باید فکر کنید (به ایده‌هایی مثل افزایش تعداد شلیک مهاجمان و افزایش جان آنها هم فکر کنید)! وقتی که بازیکن گزینه **New Game** را انتخاب کرد، پلیر ابتدا باید اسمش را ورودی بدهد و بعد ابعاد صفحه را (در صورت لزوم) مشخص کند.
- بازیکن یک **health bar** دارد که می‌تواند به صورت چند سفینه یا به صورت عددی مانند ۱۰۰ تعیین شود. بازیکن توانایی شلیک مستقیم و رو به بالا به سمت مهاجمان را دارد؛ در عین حال مهاجمان نیز بسته به نوعشان، شلیک مخصوص خود را دارند که می‌تواند دارای سرعت و شدت متفاوت باشد.
- با شروع بازی، ۴ بلوک مانع بین دو ساید، حداقل ۳ ردیف مهاجم با طول و ترکیب مناسب و سفینه باید در صفحه اصلی بازی ساخته شوند؛ همچنین تایمر و **scoreboard** باید عدد ۰ را نشان دهند.
- بازیکن با کلیدهای **a** و **d** می‌تواند به سمت چپ و راست حرکت کند و با **space** تیر به سمت مهاجمان شلیک کند. در صورت اضافه کردن فیچر دیگری در گیم پلی، می‌توانید کلیدهای بیشتری تعریف کنید.
- حداقل ۴ نوع مهاجم باید در بازی وجود داشته باشند. مهاجمان **striker** که دو نوع دارند و یکی حملاتی دیر به دیر و آسیب زیاد دارد و دیگری با آهنگ سریع‌تری حمله می‌کند ولی تیرهای کم تاثیرتری دارد. نوع سوم سفینه‌ای است که هر چند وقت یک بار از بالای مهاجمان عبور می‌کند و برخورد تیر به آن امتیاز بالاتری به همراه خواهد داشت.

- بازی در محیط ترمینال اجرا می‌شود و رابط گرافیکی آن کاراکترهای unicode هستند. صفحه بازی باید score و health bar بازیکن در هر لحظه را نشان دهد. همچنین با استفاده از فونت‌های اشاره شده در قسمت آشنایی با بازی باید سفینه و مهاجمان بازی را بسازید.
- بازی هیچ پایانی ندارد و صرفاً رکوردهای مختلفی برای بازیکن ثبت می‌شود. اما طبق چیزی که توضیح داده شد به دو شکل باخت اتفاق می‌افتد.
- بازی باید دارای pause menu باشد، به شکلی که بازی متوقف شده و موارد زیر به کاربر نمایش داده شوند:
 - ذخیره و خروج: بازی در فایل ذخیره شده و بازیکن از بازی خارج می‌شود.
 - شروع مجدد بازی
 - ادامه بازی: بازیکن می‌تواند بازی را ادامه دهد
- هر چقدر که بازی ادامه پیدا کند، به مرور باید سختی بازی بیشتر شود. به این شکل که سرعت پایین آمدن مهاجمان افزایش یابد و درجه سختی بازی بالاتر رود (طبق سیستم سختی چیده شده).
- اگر فایل جدول امتیازات موجود باشد برنامه باید اطلاعات جدول امتیازات را از همان فایل بخواند و بعد هم دوباره همان فایل را آپدیت کند. اگر اسم شخصی قبلاً در جدول امتیازات آمده بود، در صورتی که امتیازی که به دست آورده بیشتر از امتیاز قبلی باشد باید در سطر مربوطه از فایل امتیاز و زمان جدید شخص را بروزرسانی کنید.
- جدول امتیازات شامل رتبه‌بندی افرادی است که در بازی شرکت کرده‌اند، با انتخاب گزینه Leaderboard باید اسم بازیکن‌ها را به صورت مرتب شده بر حسب زمان و امتیاز نمایش دهید. یعنی اگر دو بازیکن امتیاز برابر داشته باشند، رتبه بازیکنی بالاتر است که زمان کم‌تری داشته.

فازبندی

شما به دلخواه خودتان می توانید پروژه را از هرجایی که مناسب دیدید شروع کرده و کامل کنید. فازبندی زیر صرفاً یک پیشنهاد برای شماست که اگر سردرگم هستید از آن استفاده کنید. توجه کنید که استفاده از این فازبندی کاملاً اختیاری بوده و این قسمت فقط برای راهنمایی شماست.

فاز ۱: پیاده سازی لوپ اصلی بازی

لوپ اصلی بازی یک لوپ `while` است که تا زمانی که بازی منجر به باخت نشده ادامه دارد. تمام منطق بازی و آپدیت کردن امان های صفحه بازی و منطق بازی داخل این لوپ آپدیت می شوند.

فاز ۲: پیاده سازی سفینه بازیکن

سفینه بازیکن که در پایین صفحه است توانایی بالا یا پایین رفتن ندارد و فقط می تواند به چپ یا راست حرکت کند. در این گام باید بازه حرکت سفینه بازیکن با `Arrow key` یا حروف کیبورد به چپ یا راست برود.

فاز ۳: منطق شلیک لیزر ها

باید از محل قرار گرفتن کاراکتر تانک به سمت بالا یک کاراکتر تیر پرتاب شود تا زمانی که به مانع یا انتهای صفحه برخورد کند. نحوه پیاده سازی این بخش باید بر اساس آپدیت کردن کل صفحه یا فقط خود لیزر باشد.

فاز ۴: پیاده سازی آدم فضایی ها

حرکت آدم مهاجمان به صورت مارپیچ است. در واقع هر سطر یک شیفت به چپ یا راست می خورد و سپس پس از رسیدن به آخرین ستون مجاز به پایین حرکت می کند. در طول زمان بازی این حرکت رو به پایین برای کوتاه تر کردن فاصله بین بازیکن و آدم مهاجمان سریع تر می شود که باید الگوریتم سخت تر شدن بر اساس زمان را پیاده سازی کنید.

فاز ۵: پیاده سازی بشقاب پرنده قرمز

در قسمت هایی از بازی به صورت رندوم یک سفینه قرمز از بالای صفحه رد می شود که در صورت هدف قرار دادن این سفینه در مدت زمانی که در صفحه حاضر است امتیاز قابل توجهی به بازیکن تعلق می گیرد.

فاز ۶: چک کردن برخورد موجودات

به صورت کلی بازی شامل تعداد زیادی از قوانین برای پیاده سازی برخورد بین قسمت‌های در حال حرکت در صفحه است. برخورد لیزر با بازیکن یا آدم مهاجمان، بیرون رفتن کاراکترها از صفحه بازی و موارد دیگر از این قبیل که باید چک شوند و پیاده سازی شوند که در صورت برخورد دنباله‌ای از حرکات انجام شود که یا شامل امتیازدهی به بازیکن است یا کسر health از بازیکن.

فاز ۷: شرط باخت بازی

در صورتی که بازیکن سه بار مورد اصابت قرار بگیرد باید بازی به اتمام برسد شرط اینکه بازیکن چه زمانی مورد اصابت قرار می‌گیرد باید برای اتمام game loop چک شود و در یک متغیر نگهداری شود.

فاز ۸: پیاده سازی سیستم امتیازی دهی

در نهایت می‌توانید با ثبت امتیازها به ازای هر برخورد موفق با آدم مهاجمان یا موارد دیگری که ممکن است به صورت امتیازی پیاده سازی کرده باشید به هر بازیکن یک امتیاز مجموع تعلق بدهید که در گام‌های بعد و پایان بازی از این امتیاز برای لیدربورد استفاده کنید.

نکات

شما اجازه دارید از تمامی مطالبی که در کلاس درس استاد و همین‌طور در کلاس‌های ورکشاپ بهتون آموزش داده شده استفاده کنید. شرط‌ها، حلقه‌ها، آرایه‌ها، پوینتر، کار با فایل و استراکت از جمله ابزارهایی هستند که می‌توانید برای انجام پروژه از آنها استفاده کنید.

لیست کردن دقیق تمامی ابزارهایی که اجازه دارید از آنها استفاده کنید کار راحتی نیست، لذا اکیدا پیشنهاد می‌شود هر چیزی خارج از مطالب کلاس را نیاز داشتید، حتما با منتور پروژه‌تان در میان بگذارید تا بعدا مشکل ساز نشود.

اما ابزارهایی که اجازه‌ی استفاده از آنها را هیچ جوره ندارید:

- **کلیدواژه‌ی auto**

هر چقدر هم که به نظرتان ساده باشد، از شما انتظار می‌رود روی **data type** ها مسلط باشید؛ لذا حق استفاده از این کلید واژه را ندارید و در صورت مشاهده نمره شما کسر می‌شود.

- **موتورهای بازی‌سازی (Game Engine)**

حق استفاده از موتورهای بازی‌سازی که ابزارهای پیش ساخته را در اختیارتان قرار می‌دهند ندارید، بلکه از شما انتظار می‌رود بتوانید خودتان توابع و ابزارهایتان را از صفر بسازید.

- **کتابخانه‌های OpenGL و SFML**

کارهای گرافیکیتان را می‌بایستی فقط با استفاده از کاراکترهای **ascii** و یونیکد انجام دهید و نمی‌توانید رابط‌های گرافیکی با این کتاب‌خونه‌ها بنویسید. تمامی کد شما باید در محیط ترمینال اجرا شود.

- **هر گونه API**

برنامه شما باید به خودی خود همه کاری را انجام دهد، برنامه‌ی جداگانه، **api** یا هر چیزی که خودتان کدش را ندارید را نمی‌توانید استفاده کنید.

نکته بسیار مهم: هرچیزی که مورد استفاده‌تان است را باید بلد باشید و بتوانید حین ارائه در موردش توضیح دهید و دلیل استفاده ازش را هم بیان کنید!

ابزارها و کتابخانه‌های کاربردی

- **یادآوری Unicode:**

برای نمایش کاراکترهایتان در ترمینال هم دقیقاً مثل پروژه **Minesweeper** یکی از یونیکدهایی که می‌توانید استفاده کنید کاراکترهای [box drawing](#) سایت است و برای گرفتن یونیکدهای بیشتر و متنوع‌تر می‌توانید سری به سایت [AmpWhat](#) هم بزنید! همچنین برای کاراکترهای بازی می‌توانید از لینک قسمت اول داک استفاده کنید.

برای خلاقیت بیشتر هم می‌توانید در برنامه‌تان از **ASCII Art** استفاده کنید که این بخش را به دلیل امتیازی بودن به عهده خودتان می‌گذاریم.

- کتابخانه **conio.h**:

از این کتابخانه ممکن است به تابع **getch** نیاز داشته باشید که برای گرفتن یک کاراکتر از ورودی بدون نمایش آن در کنسول استفاده می‌شود و برای دریافت آن کاراکتر نیازی به **enter** زدن ندارید.

- کتابخانه **chrono**:

ممکن است در پروژه‌تان نیاز به محاسبه تایم اجرای یک تابع یا یک فرایند را داشته باشید یا حتی بخواهید تایمی که بازی در حال اجرا بوده را نمایش دهید. برای این جور کارها باید از یک کتابخانه به اسم **chrono** استفاده کنید که توابعی برای طول زمان یا همون "**duration**" و یا ذخیره زمان در لحظه فراخوانی همان تابع دارد. از آنجایی که کتابخانه **chrono** تعداد تابع‌های زیادی دارد برایتان لینک **document** ها که تمام توابع را شامل می‌شوند قرار دادیم:

- [Cplusplus](#)
- [CppReference](#)

- مفهوم‌های **Frame, FPS, Delta Timing**:

برای نمایش دادن گرافیکی که با یونیکدها توی ترمینال ساختید، باید رابط کاربریتان را برای هر تغییر که ایجاد می‌کنید با استفاده از یک لوپ **while** (که به آن **game loop** هم می‌گوییم) تا زمانی که بازی ادامه دارد چاپ و پاک کنید به هر کدام از این چاپ کردن‌ها به اصطلاح یک **فریم** می‌گوییم.

سرعت اجرا شدن هر دور از این حلقه‌ها روی هر سیستمی ممکن است متفاوت باشد. برای مثال یک سیستم قوی ممکن است خیلی سریع‌تر هر فریم را نمایش دهد که سرعت بازی را بالا می‌برد و این باعث می‌شود که رابط کاربری شما دچار مشکلی به اسم **flicker** که همان چشمک زدن بیش از حد است شود.

برای همین شما می‌توانید مفهومی به اسم **delta timing** را به کار گیرید. این یک مفهوم بسیار کاربردی در بازی‌سازی است که ابتدا یک عدد مشخص که می‌خواهیم **fps** یا همان تعداد فریم بر ثانیه باشد را به عنوان یک **constant** تعریف می‌کنیم، بعد باید فاصله زمانی بین چاپ شدن دو فریم را محاسبه کنیم. یعنی مثلاً اگر می‌خواهیم ۴ فریم بر ثانیه بگیریم، فاصله بین هر فریم ۰.۲۵ ثانیه می‌شود. به این فاصله زمانی **delta time** یا **frame time** می‌گوییم.

با استفاده از کتابخانه chrono می‌توانیم زمان اجرای game loop را به دست آوریم. حالا چک می‌کنیم که آیا این زمان به اندازه مقدار delta time است یا نه؟ اگر بود اجازه نمایش تغییرات و ادامه اجرای بازی را به برنامه می‌دهیم. در غیر این صورت به میزان اختلاف delta time و تایم اجرای حلقه، برنامه را متوقف می‌کنیم و سپس عملیات مربوط به نمایش فریم بعدی را شروع می‌کنیم. برای متوقف کردن برنامه می‌توانید از کتابخانه thread استفاده کنید.

• تغییر رنگ خروجی در ترمینال:

روش اول:

برای بهتر کردن رابط گرافیکی باید از رنگ‌هایی که ترمینال‌ها دارند استفاده کنید. مثلاً در ویندوز برای تغییر رنگ خروجی، می‌توانید قبل از چاپ، کد ANSI را هم چاپ کنید.

[لیست کد ANSI رنگ‌ها](#)



```
cout << "\033[34m" << "Blue" << endl
      << "\033[31m" << "Red" << endl
      << "\033[32m" << "Green" << endl
      << "\033[0m " << "Normal" << endl;
```

Blue
Red
Green
Normal

شما می‌توانید از متغیرهای رشته‌ای برای ذخیره کدهای ANSI استفاده کنید. دقت کنید که بعد از چاپ یک کد، رنگ نوشته‌های ترمینال تا زمانی که کد رنگ جدیدی چاپ شود به همان رنگ اول باقی خواهند ماند.

روش دوم:

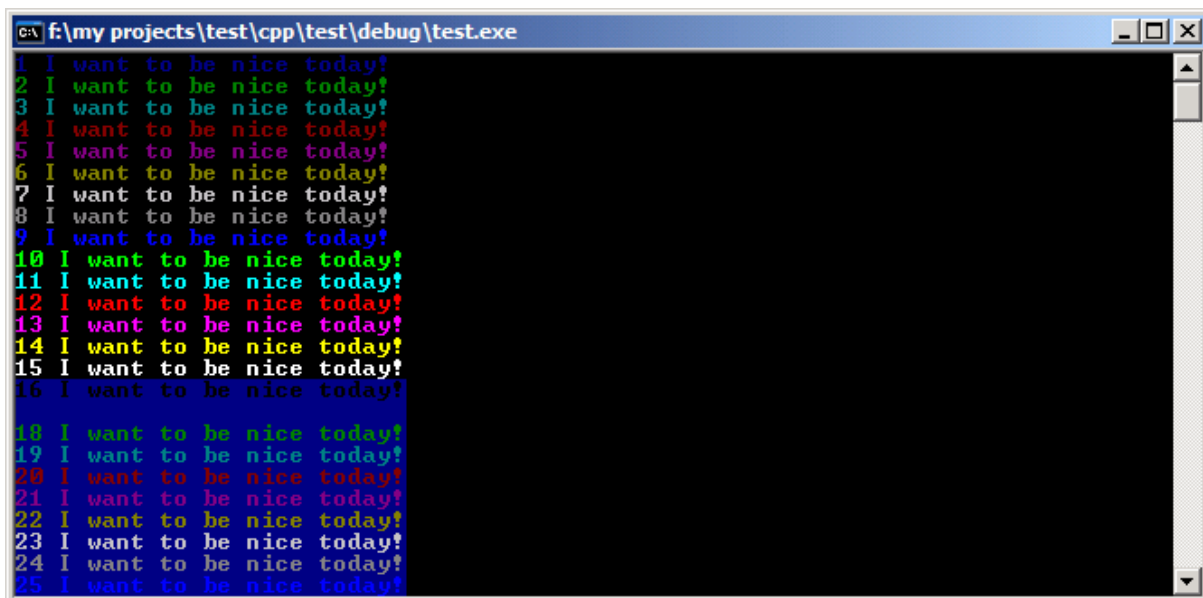
برای بهتر کردن رابط گرافیکی باید از رنگ‌هایی که ترمینال‌ها دارند استفاده کنید. مثلاً در ویندوز برای تغییر رنگ

```
1 HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
2 // you can loop k higher to see more color choices
3 for(int k = 1; k < 255; k++){
4     // pick the colorattribute k you want
5     SetConsoleTextAttribute(hConsole, k);
6     cout << k << " I want to be nice today!" << endl;
7 }
```

خروجی `cout` می‌توانید از تابع زیر در کتابخانه `windows.h` استفاده کنید.

خروجی کد بالا تصویر زیر است. یکی از مشکلاتی که ممکن است به آن برخورد کنید این است که برخی رنگ‌ها به آن شکلی که در تصویر پایین می‌بینید نباشند. برای رفع این مشکل ممکن است نیاز باشد محیطی که به عنوان ترمینال استفاده می‌کنید را عوض کنید.

(جایگزین‌های خوب `cmd`، ترمینال‌های `linux` مثل `mingw` هستند.)



• Git:

گیت ابزاری است که با استفاده از آن می‌توانید پروژه خودتان را مدیریت کنید. می‌توانید کدهایتان را در صفحه گیت‌هاب خودتان آپلود کنید و به طور همزمان با همگروهی خود به پروژه دسترسی داشته باشید و کمتر درگیر دردسر در انتقال درست کد شوید. هرچند که گیت خودش می‌تواند پیچیده باشد و استفاده از آن اختیاری است. در آینده با این ابزار به طور رسمی‌تر آشنا می‌شوید ولی اگر علاقه دارید که برای این پروژه از آن استفاده کنید، یادگیری آن کامل به عهده خودتان می‌باشد.

فیچر های امتیازی پیشنهادی

- سیو کردن کل بازی به این صورت که بازی زمانی که از آن خارج شوید یا ببازید، در فایل ذخیره شود تا در آینده بتوانید آن را load کنید. بازی های قبلی به عنوان فیچر باید قابل load شدن باشند.
- اضافه کردن موسیقی به آهنگ و تغییر موسیقی متن بر اساس درجه سختی.
- موانع می‌توانند به صورت تصادفی یا الگوهای متنوع تولید شوند.
- مهاجمان می‌توانند حملات انتحاری انجام دهند؛ به صورتی که به شکل تصادفی یکی از آنها به سمت پایین حرکت می‌کند و با برخورد با مانع یا سفینه بازیکن صدمه زیادی وارد می‌کند.
- نوع خاصی از شلیک (ترجیحا به صورت charge attack یا...) می‌تواند به صورت زاویه‌دار سه جهته باشد و برای این منظور باید دیوارها به صورت آینه‌ای عمل کنند و تیر را بازتاب دهند. استفاده از شلیک لیزری با قابلیت شارژ - لیزر همه مهاجمان مقابل خود را نابود می‌کند و با برخورد با اولین مهاجم از بین نمی‌رود.
- باس فایت؛ می‌توان در مراحل خاصی از بازی دشمنی با سختی بالا و قابلیت‌های خاص اضافه کرد تا بازی از یکنواختی خارج شود و چالش بیشتری داشته باشد.
- بازیکن کمکی - می‌توان به صورت رندوم در زمان‌هایی از بازی، سفینه‌ای کمکی به بازی فرستاد که به صورت خودکار عمل می‌کند و به کمک به بازیکن می‌آید.
- تم‌های رنگی مختلف که توسط بازیکن قابل انتخاب باشند یا به صورت اتفاقی اعمال شوند.
- افکت‌های انیمیشن انفجار و افکت‌های صوتی شامل موسیقی پس‌زمینه و صدای شلیک یا ورود سفینه‌ها و...

- ابعاد صفحه **dynamic** باشد. به شکلی که کاربر هنگام ورود به بازی طول و عرض آن را انتخاب کند؛ برای این منظور از خواص آرایه‌های **dynamic** استفاده کنید.
- می‌توانید تمامی بازی‌های انجام شده را در فایل نگه دارید و در آینده بتوانید از هر کدام از آنها **load** کنید و بازی را ادامه دهید.
- قابلیت خرید آیتم‌ها یا ارتقاها خاص از طریق فروشگاه در بازی. (این رو در صورتی میشه پیاده سازی کرد که ویژگی ذخیره اطلاعات بازیکن رو زده باشین)
- با شکست مهاجمان، آیتم‌هایی مانند بازیابی سلامت، سپر دفاعی، یخزدگی مهاجمان، یا تقویت شلیک به زمین می‌افتند که بازیکن می‌تواند با جمع‌آوری و مدیریت آنها شرایط بازی را به نفع خود تغییر دهد.
- طراحی یک مد جدید به صورتی که، بازی به جای تک‌نفره، به صورت دو نفره انجام می‌شود که هر بازیکن کنترل یک سفینه را به عهده می‌گیرد. در این حالت، دو بازیکن با هم در یک محیط مشابه نبرد می‌کنند و هدف آنها نابودی سفینه حریف است.

ارزیابی

موارد زیادی برای ارزیابی کدتان در نظر گرفته می‌شود، از جمله:

- رعایت نکات **Clean Code**، مانند خوانایی و سادگی کد
- رعایت اصول **DRY** و **KISS**

DRY: Don't repeat yourself

KISS: Keep it simple stupid

این دو اصل، از اصول مهم **Clean Code** هستند، که اولی به این معنی است که تکه‌های کدتان را تکرار نکنید، و اگر به یک کد بیشتر از یک بار نیاز دارید، آن را تبدیل به فانکشن کنید.

دومی هم به این نکته اشاره می‌کند که تا جای ممکن بهتر است ساده کد بزنید، و از پیچیدگی بیش از حد و اضافه در کد جلوگیری کنید. برای مثال وقتی چند راه حل برای یک مسئله وجود دارد، ساده‌ترین راه را انتخاب کنید.

برای فهم بهتر این دو مفهوم به [این لینک](#) می‌توانید مراجعه کنید.

- معماری کد

بهتر است که بخش‌های مختلف پروژه را جدا کنید و مجزا پیاده‌سازی کنید، مثلا منطق بازی، بخش گرافیک، بخش دسترسی به فایل و غیره. در اینصورت برای دیباگ کردن کد همکاران ساده‌تر خواهد بود.

- کامنت گذاری (به خصوص برای توابع و سکشن‌های مختلف کد)

- همکاری و تقسیم کار درست (در صورتی که پروژه را گروهی انجام می‌دهید)

ارائه پروژه به صورت حضوری است و از کل اعضای تیم انتظار می‌رود که به تمامی بخش‌های پروژه مسلط باشند و بدانند هر فانکشن و هر خط کد چه نقشی دارد.

در کنار فیچرهای اصلی پروژه، موارد امتیازی و هرگونه ویژگی خلاقانه که پیاده‌سازی کنید در ارزیابی در نظر گرفته می‌شود و زیبایی و تمیزی کار قطعا تاثیر مثبت دارد.

ددلاین و تایم ارائه

برای تحویل پروژه تا پایان روز **جمعه ۵ بهمن** باید پروژه را به صورت یک فایل ZIP ایمیل کرده باشید. حتما در کنار فایل‌های مربوط به خود برنامه، چند اسکرین‌شات از برنامه خود در حال اجرا هم قرار دهید. اگر رپپورت یا توضیحی هم در مورد پروژه‌تان نوشته‌اید می‌توانید ضمیمه کنید. نکته نهایی: ارائه پروژه به صورت حضوری در روز **یکشنبه ۷ بهمن** انجام خواهد شد. حضور تمامی اعضای تیم برای ارائه اجباریست.