SHARIF AI CHALLENGE 2020

داستان بازی

در دوران دوم زندگانی مردم در زمین، پادشاهان و سلاطین مختلفی در ایالات مختلفی زندگی کرده و مردم هم در آسایش بودند؛ همکاری بین پادشاهان به جایی رسید که برای تجارت بهتر و گسترش همکاری‌ها راه‌ها و جاده‌های مختلفی را بین ایالات خود کشیدند و باعث پیشرفت فناوری در کشورهای هم شدند. اما در دوران سوم در میان همه‌ی پادشاهان و سلاطین، حرص، نفس و جاه‌طلبی بزرگترین پادشاهان بودند که تمامی سلاطین را برای گرفتن بقیه‌ی ایالت‌ها به جان هم انداختند. سلاطین برای بدست آوردن ایالات دیگر ناگزیر از اتحاد با بقیه‌ی کشورها بودند و بهترین متحدین را می‌جستند، یارهای متفاوت را امتحان می‌کردند تا با یاران قوی‌تر ایالات بیشتری کسب کنند. بنابراین همه‌ی پادشاهان با یارانشان در جنگ حاضر می‌شدند و از راه هایی که بین ایالات مختلف بود استفاده می‌کردند تا یا به یکدیگر کمک کنند یا با کمک هم دشمنان را نابود کنند. حال شما سپه‌سالاران این پادشاهان هستید و وظیفه‌ی راهی کردن یگان‌ها و سربازان مختلف را برعهده دارید. به امید پیروزی...

عناصر بازی

تیم‌ها

هر بازی شامل ۴ تیم می‌شود که به صورت ۲ به ۲ در مقابل هم قرار گرفته و بازی می‌کنند. در واقع در هر بازی کد شما یاری از جنس کد و هوش مصنوعی دارد که باید برای رسیدن به پیروزی با هم همکاری کنید.

نقشه

هر بازی در یک نقشه که شامل عناصر زیر است انجام می‌شود:

- تایل (Cell): به هر خانه از نقشه یک تایل گفته می‌شود.

- شاه (King): خانه‌ی شاه خانه‌ای ۳x۳ در نقشه است که قابلیت حمله به حریفان را دارد. هر شاه بردی دارد که از تایل وسط آن در نظر گرفته می‌شود.

- مسیر (Path): هر مسیر دنباله‌ای با عرض یک از تایل‌ها است. هر مسیر از یک خانه‌ی شاه شروع و به خانه‌ی شاه دیگری ختم می‌شود (بین دو شاه دوست تنها یک مسیر یکتا وجود خواهد داشت). تمامی راه‌ها از وسط این شاه‌ها آغاز شده و تا تایل وسط از شاه‌های دیگر ادامه دارد. این راه‌ها می‌توانند overlap داشته باشند. مسیرها از شاه‌ها عبور نمی‌کنند. توابع پیاده‌سازی شده‌ی کلاینت از این مسیرها استفاده می‌کنند.

- فاصله: فاصله‌ی یک تایل با تایل دیگر مجموع قدر مطلق اختلاف ستون‌های دو تایل و قدر مطلق اختلاف سطرهای دو تایل است. (فاصله‌ی منهتنی)

یونیت‌ها

یونیت‌ها (Units) سربازان شما هستند که با سرعت هایی یکسان در بازی حرکت می‌کنند. سربازانی که ویژگی‌های زیر را دارا است:

- مکان (Location): هر سرباز در یک تایل جای می‌گیرد؛ حال میتواند در این تایل در هوا باشد یا روی زمین.

- سلامتی (HP): میزان سلامتی معیار سنجیده شدن جان و سلامتی یک سرباز است که اگر کوچکتر مساوی ۱ باشد آن سرباز مرده محسوب می‌شود.

- ضربه (Damage): میزان ضربه‌ی که یک سرباز به سرباز حریف ، میزانی است که از سلامتی سرباز حریف می‌کاهد.

• هدف (Target): هدف هر سرباز، در واقع مکان سربازانی است که می‌تواند به آنها را ضربه بزند: به عنوان مثال اگر هدف یک سرباز هوا باشد تنها می‌تواند سربازان حریف را که در هوا هستند مورد ضربه قرار دهد یا اگر هدف هم زمین و هم هوا باشد ، می‌تواند تمامی سربازان حریف چه در هوا باشند چه در زمین را بزنند. از طرفی هر یونیت می‌تواند یک یونیت در یک تایل را بزند (Single) یا کل یونیت‌های داخل یک تایل را مورد حمله قرار دهد(Multiple).

• امتیاز عمل (AP): شما یک میزان امتیاز عمل کلی دارید که در هر نوبت میزانی به آن افزوده می‌شود. حال برای استفاده از هر سرباز باید معادل امتیاز عمل مورد نیاز آن سرباز خرج کنید.

• برد (Range): هر سرباز بردی برای ضربه زدن به حریف دارد که از جنس فاصله از سرباز یا شاه حریف است.

• اولویت ضربه: یونیت‌ها در اختیار بازیکن‌ها نیستند؛ اولویت هدف ضربهٔ هر یونیت، با یونیتی از حریف است که به ترتیب به آن نزدیک‌تر و یا جانش کمتر است یا میزان دمج آن بیشتر است؛ در نهایت اگر همه این ملاک ها در دو یونیت برابر باشد، یونیت با آیدی کمتر به عنوان هدف انتخاب می‌شود. این هدف تا زمانی که زنده است و در دامنه ضربه زدن قرار دارد مورد حمله‌ی یونیت قرار می‌گیرد و روی آن فیکس می‌ماند. همین رویکرد در مورد شاه نیز به همین شکل است.

• اسپل (Spell): اسپل‌ها، عناصری هستند که با استفاده‌ی آن‌ها روی تایل‌هایی که مدنظر دارید می‌توانید اثراتی روی یونیت‌های خودی (هم یونیت‌های خودتان ، هم یارتان) و یونیت‌های دشمن بگذارید.

• دک (Deck): مجموعه‌ی سربازانی که در طول بازی می‌توانید استفاده کنید.

• هند (Hand): سربازانی که در نوبت فعلی می‌توانید از آن‌ها استفاده کنید مجموعه‌ای را تشکیل می‌دهند به نام هند که با استفاده از هر سرباز آن سرباز از هند خارج می‌شود و سربازی دیگر که در هند نیست و در دک هست به هند اضافه می‌شود.

• امتیاز عمل هر بازیکن (MaxAP): هر بازیکن در بازی از یک میزان امتیاز عمل برخوردار است که آن‌ها را برای استفاده از یونیت‌های مختلف خرج می‌کند.

روند بازی

شروع بازی

در این بازی ۴ بازیکن وجود دارد که به صورت ۲ به ۲ بازی می‌کنند؛

هر بازیکن تنها بر روی شاه خود تسلط دارد. بین شاه‌های حریف راه‌های متعددی وجود دارد. تعداد این راه‌ها برای هر زوج شاه حریف یکسان خواهد بود. بازیکن‌ها با قرار دادن یونیت‌های خود در این راه‌ها از طریق شاه خود و راهی کردن آن‌ها به سمت شاه‌های حریف، آن‌ها را تهدید می‌کنند. هنگامی که یک شاه نابود شد راه‌های منتهی به آن از طریق راه متصل به شاه یار، به شاه یار متصل می‌شوند.

دست یا «هَند» این گونه تعیین می‌شود که بازیکن ابتدا ۵ یونیت را انتخاب می‌کند که در دست اولیه باشند؛ با بازی کردن با هر یونیت، آن یونیت از دست خارج شده و یکی از یونیت‌های دیگر که داخل دست نیستند به صورت تصادفی وارد دست می‌شود. وارد شدن به صورت تصادفی نیز به شکل وزن‌دار خواهد بود؛ یعنی از هر یونیت که کمتر استفاده شده باشد ، احتمال اینکه وارد دست شود بیشتر می‌شود. (وزن هر یونیت برابر است با معکوس تعداد باری که از یونیت استفاده شده است به علاوه ۱)

اطلاعات بازیکن از بازی

هر بازیکن می‌داند که یونیت‌هایی که استفاده کرده است در کجای مپ و روی کدام راه هستند.

هر یونیت یا حرکت می‌کند یا اگر امکان حمله به شاه یا سرباز حریف (بر حسب برد و هدف سرباز) را داشت، ضربه می‌زند. به عبارت دیگر یونیت‌هایی که در راه‌ها راهی شدند اگر به یونیت دشمن برخورد کردند بر حسب اینکه در هوا است یا در زمین در صورت توانایی با آنها مبارزه می‌کنند یا از آن‌ها عبور می‌کنند و در انتهای راه به شاه حریف می‌رسند و به شاه صدمه وارد می‌کنند تا از میزان سلامتی آن کاسته شود.

هر شاه برای دفاع از خود تا برد معینی را می‌تواند اتک بدهد. این برد برابر ۵ است که از وسط آن و به صورت منهتنی حساب می‌شود.

هر Agent می‌تواند این اطلاعات را از یار خود ببیند که در این ترن چه یونیتی را بر روی چه راهی قرار داده است و الان چه اسپلی (اگر دارد) در اختیار دارد. هر Agent از حریفانش این را می‌داند که چه یونیتی را روی زمین گذاشته است ولی نمی‌داند در چه راهی گذاشته است.

تعامل با اسپل‌های بازی

هر ۱۰ ترن ۲ اسپل متفاوت به صورت رندوم از مجموعه‌ی اسپل‌ها انتخاب شده و به صورت یکسان به تیم‌ها داده می‌شود؛ به عنوان مثال اسپل‌های Heal و Haste انتخاب شده و به هر دو تیم داده می‌شود. سپس به صورت رندوم در اختیار هرکدام از Agentهای هر تیم یکی از آن اسپل‌ها قرار داده می‌شود و می‌توانند بعد از آن هر موقع که دلخواهشان بود استفاده کند. اسپل‌ها روی برج‌ها تاثیر نمی‌گذارند.

آپگرید یونیت‌ها

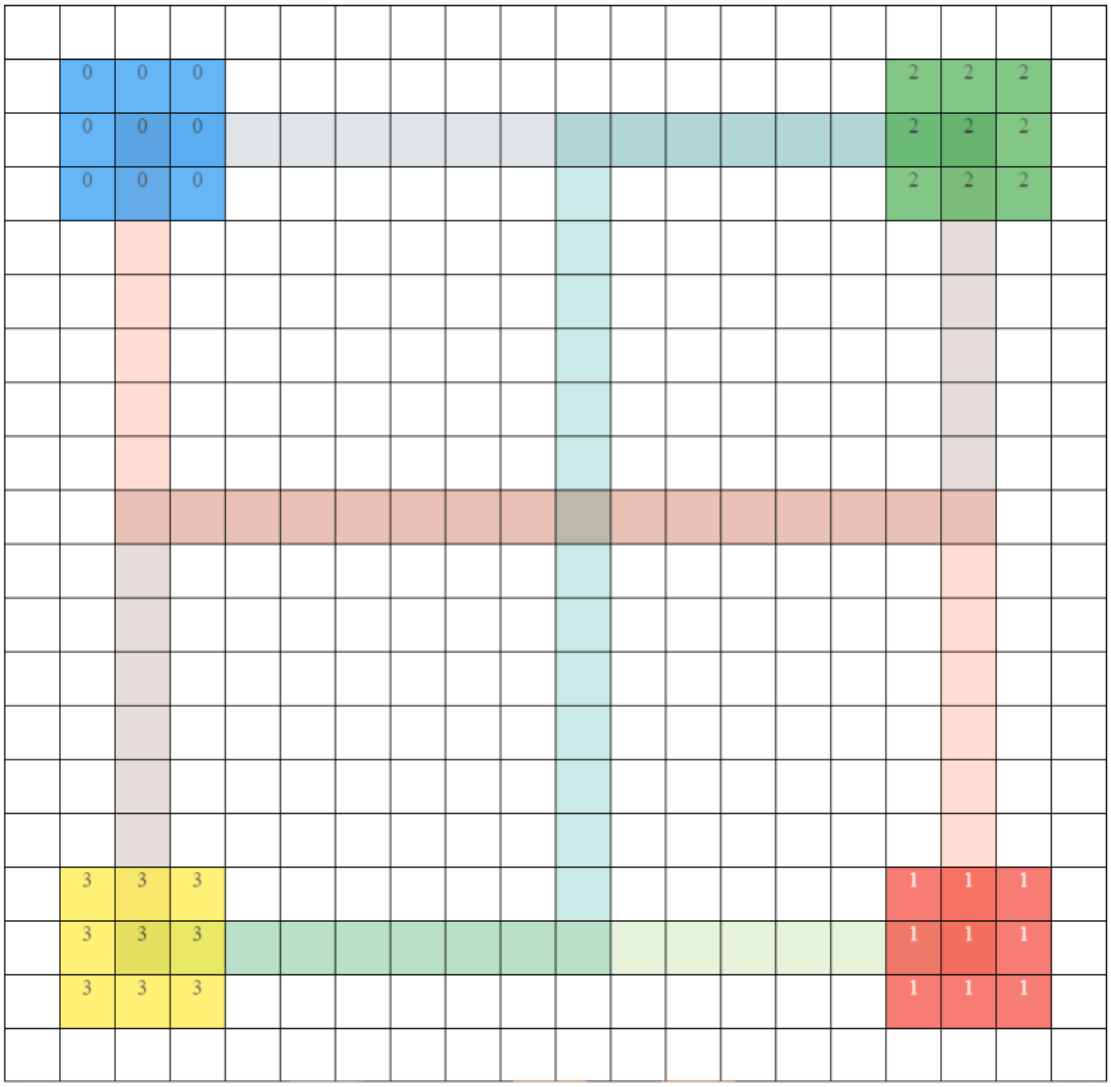
در طی بازی تیم ها بعد از ۲۳ نوبت یک ژتون می‌گیرند که می‌تونند **یکی** از یونیت هایی که روی زمین در اختیار دارند را آپگرید کنند و این آپگرید بر میزان ضربه ، برد یونیت تاثیر می‌گذارد. مانند اسپل‌ها، به هر تیم یک ژتون آپگرید برد و یک ژتون آپگرید ضربه داده می شود و بین شاه‌های هر تیم به صورت تصادفی تقسیم می‌شود.

نحوه‌ی گذشتن یک نوبت

یک نوبت به این ترتیب انجام می‌شود که بازیکن اسپلی را (اگر بخواهد) استفاده می‌کند و یونیتی را (اگر بتواند و بخواهد) روی راهی مشخص می‌گذارد. سپس یونیت‌ها کارهایی که می‌توانند را انجام می‌دهند (یعنی اگر هدف قابل ضربه‌زدنی داشت، ضربه می‌زند اما اگر نداشت با توجه به سرعتش بر روی راه مشخص خود حرکت می‌کند).

توجه داشته باشید که در هر نوبت میزان زمانی برای تصمیم گیری وجود دارد (به طور پیش فرض ۴۰۰ میلی‌ثانیه) و پس از گذشت این زمان باقی دستورات آن نوبت انجام نمی‌شوند. هر ترن دستورات کلاینت ها با هم انجام می‌شوند؛ یعنی در یک نوبت دو یونیت می‌توانند همدیگر را بکشند.

نکات تکمیلی در رابطه با راه‌ها



در عکس بالا راه‌های مختلف با رنگ‌های مختلف نمایان هستند. (اعداد روی برج‌ها آیدی آن‌ها است)
شاه‌ها به صورت مربع‌های ۳x۳ هستند که راه‌ها از تایل وسطی آنها آغاز و به تایل وسط مربع شاه دیگر ختم می‌شود.

راه‌ها می‌توانند با هم هم‌پوشانی داشته باشند و یک راه از قسمتی از راه دیگر رد شود. توجه داشته باشید که برد یونیت‌ها ورای راه‌ها است و در صورتی که یونیتی از حریف در راه دیگر و در برد یونیت شما باشد، آن یونیت را به عنوان هدف خواهد شناخت و تا زمانی که یکی از یونیت‌ها از بین بروند یا از برد هم خارج شوند به آن ضربه خواهد زد.

در طول بازی ممکن است که شاه‌ها منهدم شوند در این صورت راه‌های منتهی به آن شاه از طریق راه یکتایی که بین دو شاه یار هست به تایل وسط شاه یار آن متصل می‌شود و در صورت وجود یونیت روی راه‌های منتهی به شاه منهدم شده، آن یونیت‌ها از طریق راه یکتای مذکور به شاه دیگر حریف راه پیدا می‌کنند (توضیحات داده شده روی راه‌ها و یونیت‌ها هستند که توسط بازیکن کنترل نمی‌شوند و این توضیحات جنبه‌ی پیاده‌سازی ندارند بلکه قوانین طبیعی بازی محسوب می‌شوند)

از آنجایی که شاه‌ها از ۹ تایل ساخته می‌شوند، در هر راه در صورتی که هر تایل از شاه‌ها در برد یونیت قرار بگیرد آن شاه مورد ضربه‌ی یونیت قرار خواهد گرفت.

محدوده‌ی اختیارات هر برج

تیم‌ها تنها روی برج‌های بازی کنترل دارند؛ در ادامه کارهایی که هر برج می‌تواند را شرح می‌دهیم (در قسمت کلاینت و سرور توابع برنامه‌نویسی متناظر با هر کدام از این‌کار ها توضیح داده شده‌اند):

- **گذاشتن یونیت:** هر برج در هر نوبت می‌تواند تعدادی یونیت روی راه‌هایی که در این‌برج آغاز می‌شوند، بگذارد؛ گاهی ممکن است، برج برای کمک به یار خود از راه یکتای موجود بین ۲ یار استفاده کند که از یونیت از برج آغاز می‌کند و در ادامه به برج یار می‌رسد و در صورتی که بخواهد حرکت کند در یکی از راه‌هایی که از برج یار می‌گذرد و آن را انتخاب کرده است ادامه می‌دهد.

- **استفاده از آپگرید:** اگر برج ژتون آپگرید در اختیار داشته باشد، می‌تواند آن را روی یکی از یونیت‌های خود استفاده کند و ضربه یا برد آن را افزایش دهد.

- **استفاده از اسپل:** در صورتی که برج اسپلی در اختیار داشته باشد، می‌تواند آن را براساس نوع آن روی یک یونیت یا یک تایل از نقشه استفاده کند.

رتبه‌بندی

در آخر تیمی برنده می‌شود که مجموع سلامتی شاه‌هایش بیشتر باشد.

تیم برنده و بازنده در مجموع به ترتیب ۱۴ امتیاز و ۶ امتیاز دریافت می‌کنند؛ در تیم برنده بازیکنی که شاهش از سلامت بیشتری برخوردار باشد ۸ و بازیکن دیگر تیم ۶ و در تیم بازنده بازیکنی که شاهش میزان بیشتری سلامتی داشته باشد ۴ و دیگری ۲ امتیاز دریافت می‌کند. اگر در انتهای بازی در هر تیم سلامتی شاه‌ها یکسان بود امتیاز تیم بین اعضا تقسیم می‌شود. اگر بازی مساوی شود به هر تیم ۱۰ امتیاز می‌دهیم و به بازیکن با سلامتی بیشتر ۶ امتیاز و به بازیکن با سلامتی کمتر ۴ امتیاز می‌دهیم.

هر بازیکن در نهایت بر اساس امتیازی که در بازی‌ها گرفته است رتبه‌بندی می‌شود.

جزییات اسپل‌های بازی

اسپل‌ها دو دسته‌اند:

۱- اسپل‌هایی که می‌توانند یک منطقه‌ی مربعی را مورد تاثیر قرار دهند؛ بازیکنان می‌توانند با نشانه کردن یک tile در جهت ۸ خانه‌ی همسایه‌اش تاثیر مدنظر را بگذارند. (Area Spell)

۲- اسپل‌هایی که روی یک یونیت اعمال می‌شوند. (Unit Spell)

تمامی اسپل‌ها حداکثر تا ۳ ترن تاثیر می‌گذارند و در هر ترن فقط ۱ اسپل توسط هر بازیکن قابل استفاده است. تاثیر اسپل فقط روی یونیت‌هایی است که الان در محدوده اثر آن باشند نه اینکه در آینده به آن منطقه منتقل شوند و Duration میزان نوبتی است که تاثیر اسپل روی یونیت‌ها می‌ماند.

ترتیب اثر هر اسپل هم به صورت زیر است:

۱- damage/poison
۲- heal
۳- tele
٤- haste
٥- duplicate

type	Detail	Duration (turn)	Area/Unit	Target unit
HP (damage)	به تمامی یونیت‌های حریف که در آن tile و ۸ همسایه‌اش هست ۴ ضربه می‌زند.	۱	AreaSpell	**ENEMY**
HP (heal)	HP تمام یونیت های خودی در آن tile و ۸ همسایه اش ۲تا افزایش می‌یابد.	۱	AreaSpell	**ALLIED**
Tele	می‌توان یکی از یونیت‌های خودش را به یکی از خانه‌های یکی از راه‌های دلخواه که از نیمه‌ی آن راه عقب‌تر است منتقل کرد.	۱	UnitSpell	**SELF**
Duplicate	از روی یونیت‌های یک تایل یک یونیت کپی می‌سازد که از نظر Damage و HP یک‌چهارم است.	۳	AreaSpell	**ALLIED**
HP (poison)	به تمامی یونیت‌های حریف که در آن tile و ۸ همسایه‌اش هست یک سم می‌زند. (که این سم هر ترن ۱ واحد ضربه وارد می‌کند).	۳	AreaSpell	**ENEMY**
Haste	سرعت تمام یونیت های خودی در آن tile و ۸ همسایه اش به ۲ tile بر turn افزایش می‌یابد.	۲	AreaSpell	**ALLIED**

** در توضیح اسپل duplicate باید بگوییم که خود یونیت تغییر نمی‌کند، صرفا یک یونیت کپی از آن ساخته می‌شود که ویژگی‌های آن ذکر شد. بعد از گذشت تعداد نوبتی که این اسپل طول می‌کشد، کپی ساخته شده (بدون توجه به میزان سلامتی آن) از بین می‌رود.

جزییات یونیت‌های بازی

id	Location	Range	Damage	Target	HP	AP
0	Ground	4	15	Single Ground/ Air	15	**4**
1	Ground	4	5	Multiple Ground	30	**3**
2	Ground	4	10	Multiple Air	20	**4**
3	Ground	1	12	Single Ground	25	**6**
4	Ground	1	10	Multiple Ground	50	**10**
5	Air	1	5	Single Air	30	**3**
6	Air	4	5	Single Ground/ Air	15	**2**
7	Air	2	8	Multiple Ground	30	**5**
8	Air	2	15	Single Ground/ Air	20	**6**

مشخصات کلی بازی

- امتیاز عمل اولیه: ۱۰
- تعداد ترن بازی: ۱۰۰
- تعداد ترن برای آمدن اسپل‌های بعدی: ۱۰
- سایز هند: ۵
- سایز دک: ۹
- امتیاز عمل اضافه‌شده در هر ترن: ۲
- شاه‌ها:
  - سلامتی: ۹۰
  - قدرت ضربه: ۱۰
  - برد: ۶

شروع کدنویسی (بایدها و نبایدها)

- شما باید کد هوش مصنوعی خود را در توابع pick و turn و end در فایل AI.java (برای زبان های دیگر فایلی به همین نام) قرار دهید.

- شما می‌توانید کد کلاینت داده شده را تغییر دهید، به آن فایل اضافه کنید یا از آن فایل حذف کنید، به شرط آنکه تغییرات داده شده در کامپایل و اجرای کلاینت و ارتباط آن با سرور اختلالی ایجاد نکند. در مورد هر کلاینت نکاتی ذکر شده که به آنها نیز باید توجه شود. همچنین باید تغییرات احتمالی فایل های دیگر کلاینت (فایل هایی غیر از فایلی که در آن کد میزنید) را در نظر بگیرید.

- شما می‌توانید برای به روز بودن کلاینت ها یا سرور خود به آخرین نسخه منتشر شده در repository مسابقات مراجعه کنید.

- همواره نسخه آخر مستندات بازی، نقشه‌ها و سرور بازی در مخزن AIC20-Game موجود هستند.
https://github.com/SharifAIChallenge/

- ریپو کلاینت ها با پیشوند AIC20-Client آغاز می‌شود.

- در کلاینت پایتون برای ساده‌تر کردن برنامه نویسی، از تایپ‌ها (type hint) استفاده شده است. بنابراین باید از ورژن پایتون ۳٫۶ به بالا استفاده کنید. همچنین توصیه می‌شود از نسخه های جدیدتر نرم افزار pyCharm استفاده کنید زیرا نسخه‌های پیشین سازگاری پایینی با این مورد دارند.

- در کلاینت پایتون مجاز به استفاده از کتابخانه‌های زیر هستید:

```
torch==1.4.0
tensorflow==1.15.0
scikit-learn==0.22.1
numpy==1.17.5
pandas==0.25.3
```

نحوه اجرا بازی به صورت لوکال

- ابتدا باید فایل jar سرور را اجرا کنید.

```
java -jar server.jar
```

- سپس یک پنجره باز می‌شود که فایل مپ باید به آن ورودی داده شود. برای خودکارسازی این امر از وارد کردن مسیر فایل مپ در environment variable که در بالا اشاره شد میتوانید استفاده کنید.

- پس از آن با ران کردن ٤ کلاینت به صورت همزمان بازی اجرا می‌شود. توجه کنید که ترتیب ران کردن کلاینت ها همتیمی ها را مشخص میکند. کلاینت اول و سوم با هم و کلاینت دوم و چهارم با هم همتیمی میشوند.

- برای توضیحات تکمیلی برای ران کردن کلاینت به فایل readme آنها مراجعه کنید.

دستورات کمکی محیط Command Line

- با دستور زیر میتوانید از طریق Command Line نقشه مورد نظر خود را به عنوان ورودی پیش از اجرا به سرور بدهید:

```
export AICMap=/path/to/map
```

- در ویندوز از طریق environment variable یا با استفاده از دستور set یا setx این کار امکان پذیر است.

```
setx AICMap /path/to/map
```

- با آپشن extra میتوان به محدودیت زمانی های بازی اضافه کرد. برای مثال با دستور زیر ۴۰۰ میلی ثانیه به تایم اوت هر ترن اضافه می‌شود.

```
java -jar server.jar --extra=400
```

نکات فنی

- همه فیلدهای کلاس‌ها که در زیر آمده‌اند در کلاینت های پایتون و گو به طور مستقیم و در کلاینت‌های دیگر توسط getter شان قابل دسترسی میباشند.

- در بین دستوراتی که از کلاینت به سرور فرستاده می‌شوند، در صورت نامعتبر بودن یک دستور، این دستور نادیده گرفته میشود و اولین دستور معتبر بعدی انجام می‌شود. ترتیب دستوراتی که از یک جنس هستند به همان ترتیبی است که شما در کلاینت ارسال می‌کنید.

- در صورتی که یک حرکت یا قابلیت به هر دلیل انجام نشود، امتیاز عمل آن از تیم کسر نمی‌شود.

- رفرنس آبجکت world در توابع کلاس AI هر ترن از ابتدا ساخته می‌شوند و رفرنس‌های نوبت قبل معتبر نیستند. برای حفظ دسترسی به آبجکت‌ها، توصیه می‌شود از آیدی آنها، که همواره ثابت است، استفاده کنید. البته در مورد مسیر (path)، به جز برای کلاینت سی پلاس پلاس، استفاده از رفرنس موردی ندارد.

- اکثر توابعی که ورودی‌های player, unit, spell و cell دارند، با ورودی‌های شناسه و row, col هم پیاده‌سازی شده‌اند.

- در مورد بالا در کلاینت پایتون، متدها به صورت دیگری مقداردهی می‌شوند. برای مثال فرض کنید در یکجا Cell و در جای دیگر ورودی های row و col داده شوند. نحوه نگارش به صورت زیر خواهد بود:

```
method_name( Cell = … , … )method_name (row = … , col = ..., , … )
```

• توابعی که اطلاعات را از کلاینت به سرور می‌فرستد، روی اطلاعات همان ترن کلاینت تاثیری نمی‌گذارند. مثلا با صدا زدن تابع putUnit یونیت مورد نظر همان موقع از دست خارج نمی‌شود و میزان ap بازیکن کم نمی‌شود؛ بلکه در ترن بعد این اطلاعات آپدیت خواهد شد. بنابراین کدی مثل

```
While(myPlayer.getAp() > 3) {
      world.putUnit(…);
}
```

ممکن است هیچ وقت به پایان نرسد و در نتیجه دستورات بعدی آن ترن اجرا نشوند. در صورت تمایل میتوانید خودتان فیلد ap را برای بازیکن ست کنید (دسترسی وجود دارد) تا در ترن فعلی استفاده کنید اما توجه داشته باشید که ترن بعد رفرنس‌ها عوض میشوند و فیلدها آپدیت خواهند شد.

• کلمات attack و damage از لحاظ مفهومی معادلند. برای مثال آپگرید کردن دمیج یک یونیت، فیلد attack آن را افزایش می‌دهد.

• یونیت پایه را نمی‌توان به طور صریح روی مسیر بین دو یار قرار داد. برای اینکه یونیت را از یار عبور دهید باید از مسیرهای متصل به یار (به جز مسیر بین دو یار) استفاده کنید؛ در این صورت یونیت قرار داده شده از مسیر بین دو یار عبور می‌کند و پس از رسیدن به شاه یار ادامه مسیرش را از مسیر انتخاب شده می‌رود. همچنین در صورتی که شاهی بمیرد، یونیت ها ادامه مسیرشان را از مسیر بین دو یار ادامه می‌دهند تا به شاه دیگر حمله را ادامه دهند. در تمام طول مسیر، مسیرِ یونیت مسیرِ انتخاب شده را نشان می‌دهد.

• برای یونیت و شاه‌های مرده، فیلدهای مربوط به تارگت معتبر نیستند.

AI

Void pick(World);
این تابع در ابتدای بازی صدا می شود – برای انجام پیش‌پردازش بازی و انتخاب دک

Void turn(World);
این تابع هر ترن یک بار صدا می شود – برای انجام حرکات مورد نیاز بازی

Void end(World, scores);
این تابع تنها در انتها پس از پایان بازی صدا زده می شود و اعمال انجام شده در این تابع روی بازی تاثیری نمیگذارد. با استفاده از این تابع به نتیجه بازی دسترسی وجود دارد و از آن میتوان برای آموزش هوش مصنوعی و پردازش‌های پس از بازی استفاده کرد. ورودی scores یک مپ از int به int است که کلید آن آیدی بازیکن و مقدار آن امتیاز بازیکن است.

World

Void chooseHandById(typeIds: List[int]) Void chooseHand(baseUnits: List[baseUnit])
انتخاب دک در ابتدای بازی، یونیت پایه های اول در لیست به عنوان هند انتخاب می‌شوند.

Player getMe() Player getFriend() Player getFirstEnemy () Player getSecondEnemy()
بازیکن های بازی را برمیگرداند

Map getMap()
مپ بازی را برمیگرداند

```
List[Path] getPathsCrossingCell(Cell)
List[Path] getPathsCrossingCell(row, col)
```

تمام مسیرهایی که از خانه داده شده رد می‌شوند (شامل آن خانه هستند) را برمیگرداند.

```
List[Unit] getCellUnits(Cell)
List[Unit] getCellUnits(row, col)
```

یونیت های داخل خانه داده شده را برمیگرداند.

```
Path getShortestPathToCell(Player, Cell)
Path getShortestPathToCell(Player, row, col)
```

با ورودی گرفتن یک بازیکن و یک خانه کوتاه ترین مسیر از آن بازیکن به آن خانه را برمیگرداند. مسیر میتواند از همتیمی عبور کند. در صورتی که این کوتاه‌ترین مسیر، مسیری باشد که به شاه یار رفته و از یکی از مسیرهای یار عبور می‌کند، خروجی تابع مسیر یار را برمی‌گرداند، یعنی ابتدای مسیر روی شاه یار خواهد بود. مسیر بین یارها به عنوان خروجی برگردانده نمی‌شود.

```
void putUnit(BaseUnit, Path)
```

یونیت داده شده را روی مسیر داده شده راهی میکند. نتیجه ترن بعد اعمال می‌شود.

```
int getCurrentTurn()
```

ترن فعلی را میدهد.

```
int getRemainingTime()
```

زمان باقی‌مانده تا تایم‌اوت را به میلی‌ثانیه میدهد.

```
Void castUnitSpell(Unit, Path, Cell, Spell)
Void castUnitSpell(Unit, Path, row, col, Spell)
```

برای اسپل تلپورت یونیت داده شده را در زمین به مسیر داده شده و به خانه داده شده میبرد. خانه ورودی میتواند خانه ای از مسیر یکتای بین شما و یارتان باشد. نتیجه ترن بعد اعمال می‌شود.

```
Void castAreaSpell(Cell, Spell)
Void castAreaSpell(row, col, Spell)
```

روی خانه داده شده اسپل محیطی را کست میکند، طوری که مرکز اسپل روی آن خانه قرار می‌گیرد. نتیجه ترن بعد اعمال می‌شود.

```
List[Unit] getAreaSpellTargets(Cell, Spell)
List[Unit] getAreaSpellTargets(row, col, Spell)
```

این تابع میگوید اگر اسپل محیطی روی خانه‌ای کست شود به چه یونیت‌هایی میخورد.

```
int getRemainingTurnsToUpgrade()
int getRemainingTurnsToGetSpell()
```

به ترتیب ترن های باقی مانده تا دریافت توکن آپگرید و اسپل را میدهد. در ترنی که توکن یا اسپل داده می شود مقدار این تابع ماکزیمم خود است؛ به عبارت دیگر خروجی تابع هیچ‌وقت صفر نیست.

```
int getRangeUpgradeNumber()
int getDamageUpgradeNumber()
```

به ترتیب تعداد توکن‌های آپگرید رنج و دمیج قابل استفاده را میدهند.

```
Spell getReceivedSpell()
```

اسپلی که در این ترن به دست‌مان رسیده را میدهد. اگر اسپل این ترن داده نشده باشد، خروجی null است.

```
Spell getFriendReceivedSpell()
```

اسپلی که در این ترن به دست هم‌تیمی رسیده. اگر این ترن اسپل نرسیده باشد خروجی null است.

```
Void upgradeUnitRange(Unit)
Void upgradeUnitDamage(Unit)
```

به ترتیب رنج و دمیج یونیت داده شده را آپگرید میکند. در صورت معتبر بودن یکی از توکن‌های آپگرید کم می شود. نتیجه ترن بعد اعمال می‌شود. ممکن است در همان ترنی که یونیت آپگرید می‌شود، کشته شود؛ در این حالت هم توکن آپگرید مصرف شده است.

```
List[BaseUnit] getAllBaseUnits()
```

لیست تمام یونیت‌های پایه (کارت های) بازی را میدهد.

```
List[Spell] getAllSpells()
```

لیست تمام اسپل های بازی را میدهد.

`King getKingById(playerId)`
`Spell getSpellById(spellId)`
`BaseUnit getBaseUnitById(typeId)`
`Player getPlayerById(playerId)`
`Unit getUnitById(unitId)`

این توابع با ورودی گرفتن آیدی آبجکت‌ها خود آبجکت را باز می‌گردانند.

`GameConstants getGameConstants()`
ثابت های بازی را برمیگرداند.

Player

`int playerId`	آیدی بازیکن
`List[BaseUnit] deck`	دک بازیکن. تنها برای بازیکن خودی(و نه همتیمی و حریف‌ها) معتبر است
`List[BaseUnit] hand`	دست بازیکن. تنها برای بازیکن خودی(و نه همتیمی و حریف‌ها) معتبر است
`int ap`	میزان ایپی باقی مانده. تنها برای بازیکن خودی(و نه هم تیمی و حریف‌ها) معتبر است

`King king`	کینگ بازیکن مسیرهایی که از کینگ بازیکن شروع میشوند. خانه ابتدای مسیر روی کینگ اوست. مسیر بین همتیمی‌ها جزو آن نیست.
`List[Path] pathsFromPlayer` `Path pathToFriend`	مسیر بازیکن به هم تیمی اش.ابتدایش از آن کینگ شروع می شود.
`List[Unit] units`	یونیت‌های زنده بازیکن
`CastAreaSpell castAreaSpell`	اسپل محیطی که در ترن قبل توسط این بازیکن کست شده بود. اگر چیزی کست نشده باشد نال است.

`CastUnitSpell castUnitSpell`	اسپل یونیت که در ترن قبل توسط این بازیکن کست شده بود. اگر چیزی کست نشده باشد نال است.
`List[Unit] duplicateUnits`	یونیت های داپلیکیت و زنده بازیکن
`List[Unit] hastedUnits`	یونیت های هیست و زنده بازیکن
`List[Unit] playedUnits`	یونیت هایی که بازیکن آن ها را در ترن قبل بازی کرد (روی یک مسیر قرار داد)
`List[Unit] diedUnits`	یونیت های بازیکن که در ترن قبل مردند.
`Unit rangeUpgradedUnit`	یونیتی که ترن قبل توسط بازیکن رنجش آپگرید شد. اگر ترن قبل بازیکن آپگریدی روی رنج انجام نداده باشد، نال است.
`Unit damageUpgradedUnit`	یونیتی که ترن قبل توسط بازیکن دمیج آپگرید شد. اگر ترن قبل بازیکن آپگریدی روی دمیج انجام نداده باشد، نال است.
`List[Spell] spells`	اسپل های قابل استفاده بازیکن را میدهد. تنها برای بازیکن خودی و هم تیمی معتبر است.
`bool isAlive()`	نشان میدهد بازیکن زنده است یا نه که معادل با زنده بودن کینگ اوست.
`int getHp()`	سلامتی بازیکن را میدهد که معادل با سلامتی کینگ اوست.
`int getSpellCount(Spell)`	تعداد اسپل های اسپل داده شده که بازیکن میتواند استفاده کند را برمیگرداند. تنها برای بازیکن خودی و هم تیمی معتبر است.

Unit

`BaseUnit baseUnit`	اطلاعات پایه یونیت در این آبجکت قرار دارد.
`Cell cell`	خانه ای که یونیت در آن قرار دارد.
`int unitId`	آیدی یونیت
`int hp`	سلامتی یونیت، اگر مرده باشد ۰ است.

`Path path`	مسیری که یونیت روی آن قرار دارد. این مسیر برای یونیت های دشمن null است. ابتدای مسیر روی شاه بازیکنی که یونیت را بازی کرده، یا یار آن بازیکن، قرار دارد.
`Unit target`	کسی که قرار است این یونیت به آن حمله کند. اگر به کسی حمله نمی‌کند یا به کینگ حمله میکند مقدارش null است
`Cell targetCell`	خانه ای که یونیت به آن حمله میکند. اگر به کینگ حمله میکند هم آن خانه از کینگ که به آن حمله میکند را برمی‌گرداند.
`King targetIfKing`	در صورتی که یونیت قرار است به کینگ حمله کند، آن کینگ را برمی‌گرداند.
`int playerId`	آیدی بازیکنی که این یونیت برای اوست
`int damageLevel`	لول دمیج یونیت که با آپگرید زیاد می شود. در ابتدا ۰ است.
`int rangeLevel`	لول رنج یونیت که با آپگرید زیاد می شود. در ابتدا ۰ است.
`int range`	رنج یوینت
`int attack`	دمیج یونیت
`bool isDuplicate`	نشان میدهد یونیت داپلیکیت هست یا نه
`bool isHasted`	نشان میدهد یونیت هیست شده یا نه(سرعتش زیاد شده یا نه)

`List[CastSpell] affectedSpells`	اسپل هایی که روی یونیت خورده بودند و ترن قبل تاثیر گذاشتند را برمیگرداند.

King

`Cell center`	خانه وسط شاه را میدهد
`int hp`	سلامتی شاه، اگر مرده باشد ۰ است.
`int attack`	دمیج شاه
`int range`	رنج کینگ که از خانه وسط آن محاسبه می شود.
`bool isAlive`	زنده است یا خیر
`int playerId`	آیدی بازیکن صاحب کینگ
`Cell targetCell`	خانه ای که حمله کینگ آنجا میخورد.

Map

`int rowNum`	تعداد سطرهای مپ
`int colNum`	تعداد ستون‌های مپ
`List[Path] paths`	لیست تمام مسیرهای مپ
`List[Unit] units`	لیست یونیت‌های زنده داخل مپ
`List[King] kings`	لیست کینگ‌های مپ
`Cell[][] cells`	لیست دو بعدی خانه های مپ
`Cell getCell(int row, int col)l`	خانه با مختصات داده شده را برمیگرداند.

Cell

`int row`	شماره سطر
`int col`	شماره ستون

Path

`int id`	آیدی مسیر
`List[Cell] cells`	لیست خانه‌های مسیر

BaseUnit

`int typeId`	آیدی یونیت پایه
`int maxHp`	سقف سلامتی یونیت
`int baseAttack`	دمیج اولیه یونیت
`int baseRange`	رنج اولیه یونیت
`UnitTarget targetType`	اینکه یونیت چه دسته ای از دشمنان را هدف قرار می‌دهد را نشان میدهد.
`bool isFlying`	اینکه یونیت روی هوا راه میرود یا روی زمین را نشان میدهد.
`bool isMultiple`	اینکه یونیت به تمام یونیت های خانه حمله میکند یا تنها به یک نفر را نشان میدهد.
`int ap`	میزان ایپی لازم برای قرار دادن یونیت روی مپ

Enum UnitTarget

`GROUND`
`AIR`
`BOTH`
	اینکه یونیت چه دسته ای از دشمنان را هدف قرار می‌دهد را نشان میدهد.

Spell

`SpellType type`	نوع اسپل را نشان میدهد (HP, TELE, HASTE, DUPLICATE)
`int typeId`	آیدی اسپل را نشان میدهد. این با فیلد قبل فرق دارد چون ممکن است از یک تایپ دو نوع اسپل وجود داشته باشد.
`int duration`	زمان اثر اسپل را نشان میدهد
`int priority`	اولویت اسپل را نشان میدهد. یعنی اسپل با اولویت کمتر زودتر اعمال می شود.

`SpellTarget target`	نوع هدف اسپل را نشان میدهد.
`int range`	رنج اسپل
`int power`	قدرت اسپل را نشان میدهد. در صورتی که اسپل دمیج وارد میکند مقدارش منفی است.
`bool isDamaging`	نشان میدهد که اسپل دمیج وارد میکند یا خیر که در صورتی که دمیج وارد کند حتما روی یونیت‌های دشمن اثر میگذارد. (target آن ENEMY میباشد.)
`bool isAreaSpell( )`	اگر اسپل محیطی باشد
`bool isUnitSpell( )`	اگر اسپل یونیتی باشد

Enum SpellType

`HP`
`TELE`
`DUPLICATE`
`HASTE`
	نوع اسپل

Enum SpellTarget

SELF	این یعنی اسپل تنها روی یونیت های بازیکن کست کننده اعمال می شود.
ALLIED	این یعنی اسپل روی یونیت های بازیکن کست کننده و هم تیمی اش اعمال می شود.
ENEMY	این یعنی اسپل روی یونیت های بازیکن‌های حریف اعمال می شود.

CastSpell

Spell spell	اسپلی که کست شد را نشان میدهد.
int id	آیدی کست‌اسپل
int casterId	آیدی بازیکنی که اسپل را کست کرد
Cell cell	برای اسپل AreaSpell خانه مرکز اسپل را نشان میدهد، برای اسپل یونیتی (تلپورت) خانه‌ای که یونیت به آن رفته را نشان میدهد.
List[Unit] affectedUnits	لیست یونیت هایی که اسپل رویشان اثر گذاشته را میدهد.

CastAreaSpell

این کلاس از کلاس CastSpell ارث‌بری میکند و مربوط به اسپل های AreaSpell که کست شدند می‌باشد.

int reamainingTurns	تعداد ترن باقی‌مانده تا پایان تاثیر اسپل را نشان میدهد.

CastUnitSpell

`Unit unit`	یونیتی که اسپل روی آن اعمال شده را نشان میدهد.
`Path path`	مسیری که یونیت روی آن رفته را نشان میدهد. اگر بازیکن کست کننده جزو دشمنان باشد، این متغیر نال می شود.

GameConstants

`int maxAp`	سقف ap را نشان میدهد
`int maxTurns`	تعداد ترن بازی
`int turnTimeout`	زمان هر ترن را به میلی‌ثانیه نشان میدهد

`int pickTimeout`	زمان پیک را به میلی‌ثانیه نشان میدهد
`int turnsToUpgrade`	تعداد ترنی که یک توکن آپگرید به بازیکنان داده می شود. مثلا اگر ۲۳ باشد یعنی هر ۲۳ ترن یکبار یک توکن به هر بازیکن داده می شود.
`int turnsToSpell`	تعداد ترنی که اسپل به بازیکنان داده می شود. مثلا اگر ۱۰ باشد یعنی هر ۱۰ ترن یکبار یک اسپل به هر بازیکن داده می شود.
`int damageUpgradeAddition`	با هر آپگرید رنج چه میزان رنج افزایش می‌یابد
`int rangeUpgradeAddition`	با هر آپگرید دمیج چه میزان دمیج افزایش میبابد
`int deckSize`	سایز دک
`int handSize`	سایز دست

راهنمای اپلیکیشن بازی

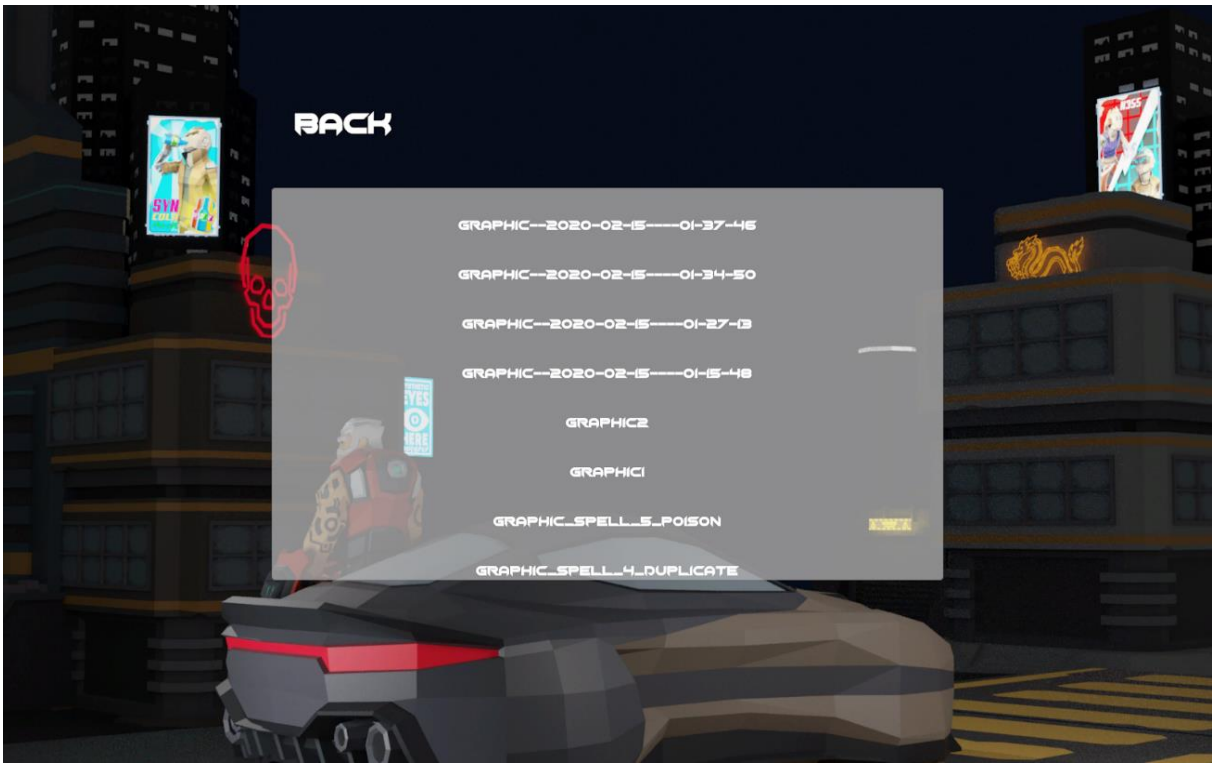
اجرای فایل سرور

بعد از زدن دکمه‌ی RUN SERVER وارد صفحه‌ی زیر می‌شویم. در این صفحه زمان تاخیر اضافی سرور را انتخاب می‌کنیم.

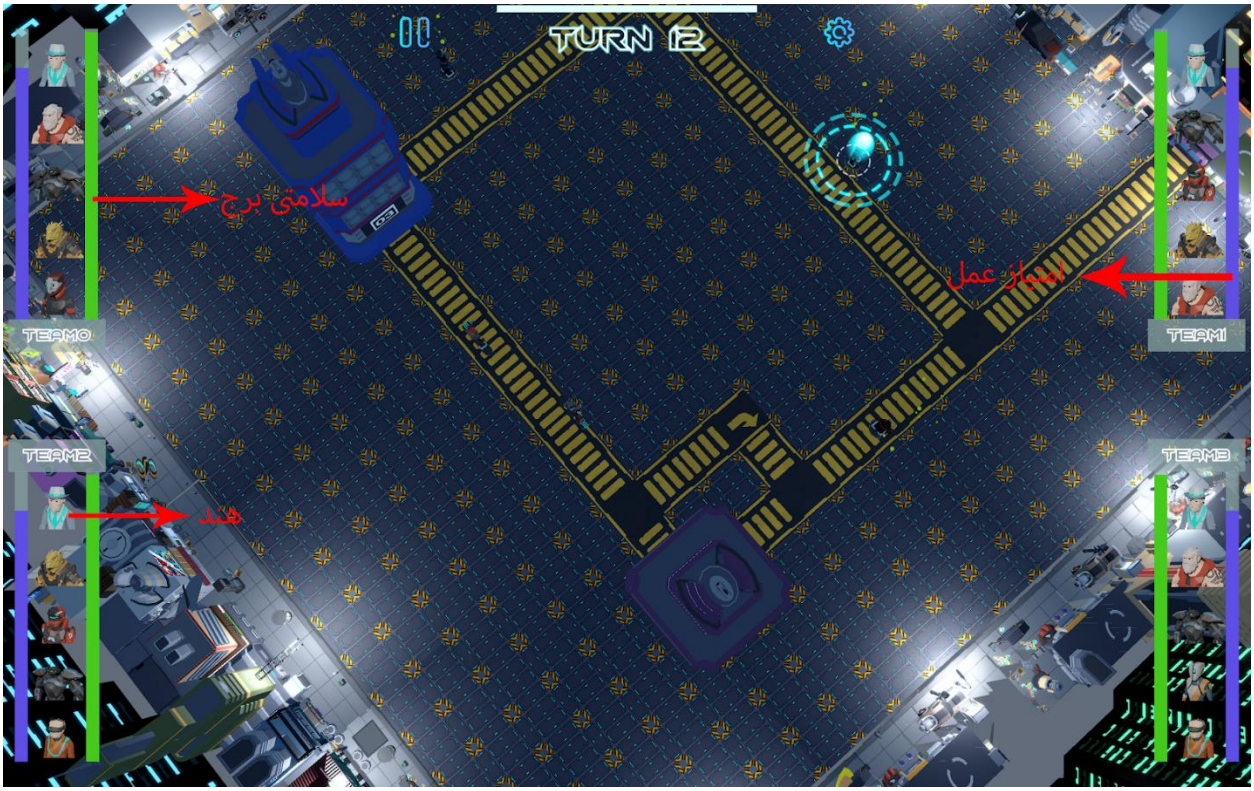
با انتخاب زمان تاخیر فایل jar سرور اجرا می‌شود و باید فایل نقشه را در صفحه باز شده انتخاب کنید. برنامه سرور تا زمانی که کلاینت‌ها اجرا نشده‌اند و بازی به اتمام نرسیده باز می‌ماند. بعد از اجرای بازی فایل لاگ بازی ساخته می‌شود.

اجرای لاگ بازی‌ها

بعد از زدن دکمه‌ی RUN LOG صفحه‌ی زیر باز می‌شود. در این صفحه لاگ مورد نظر برای اجرا را انتخاب کنید.



نمایش‌گر بازی



برای حرکت دوربین در نمایش‌گر لاگ بازی می‌توانید از دکمه‌های زیر استفاده کنید.

E: حرکت به سمت جلو

Q: حرکت به سمت عقب

A: حرکت به سمت چپ

D: حرکت به سمت راست

W: حرکت به سمت مکانی که دوربین به آن سمت است

S: حرکت خلاف مکانی که دوربین به آن سمت است

C: تغییر دوربین

همچنین برای تغییر جهت دوربین با نگه‌داشتن راست کلیک موس، موس را بچرخانید.

برای تغییر سرعت اجرای بازی و تنظیمات دیگر از منوی تنظیمات که از طریق دکمه‌ی چرخ‌دنده‌ی بالای صفحه قابل دسترسی‌است، استفاده کنید.

محل قرارگیری فایل لاگ‌ها و سرور

برای دیدن لاگ‌های دانلود شده از سایت یا لاگ‌هایی که از سرور بازی گرفته‌اید در اپلیکیشن با توجه به سیستم‌عامل خود از روش‌های زیر استفاده کنید.

مک: روی آیکون بازی راست کلیک کرده show package contents را انتخاب کنید سپس فایل لاگ را در فولدر
Contents/Server/Log قرار دهید.
ویندوز و لینوکس: فایل لاگ را در فولدر AIC_Data/Server/Log قرار دهید. توجه کنید فولدر AIC_Data همیشه کنار فایل بازی باشد.

لطفا از بروز بودن فایل سرور (با اسم server.jar) در فولدر Server اطمینان حاصل کنید.