



Basics of Programming Final Project

Winter 2024

Course Instructor: Dr. Saeed Reza Kheradpisheh



معرفی پروژه

سلام دوستان! امیدواریم حالتون خوب باشه و ترم خوبی رو سپری کرده باشید.

پروژه این ترم شما پیاده سازی بازی تتریس (Tetris) هست که برای اولین بار در سال 1984 با زبان Pascal ساخته شد. توجه کنید که این پروژه بخش خوبی از نمره نهایی شما رو تشکیل میده پس حتما به اندازه‌ی کافی روش وقت بگذارین.

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 هست.

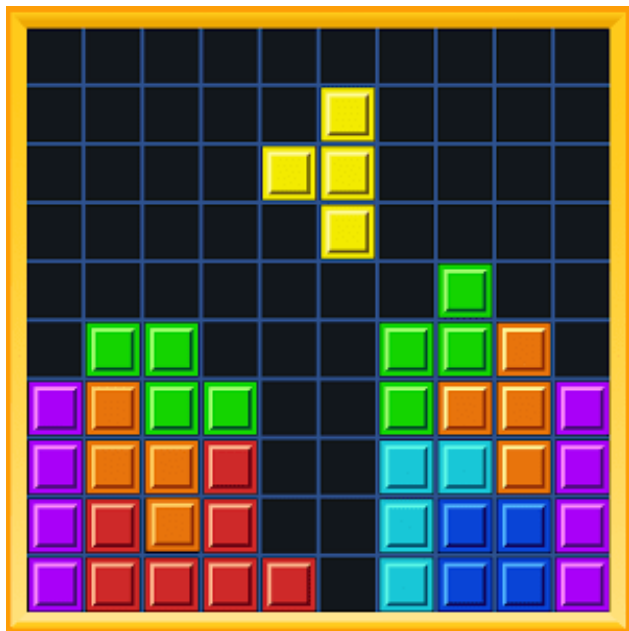
پیشنهاد می‌کنیم که با توجه به حجم پروژه، این کار رو تو تیم های دو نفره انجام بدین ولی انجام پروژه به صورت تک‌نفره هم مجازه. بعد از تیم بندی اطلاعات تیمتون رو تو [این لینک](#) وارد کنید.

پس از تایید شدن گروه‌بندی‌ها برای هر تیم یه منتور مشخص میشه که در انجام پروژه اگر جایی به مشکل یا سوالی برخوردید بتونید از کمک منتور پروژه‌تون استفاده کنید.

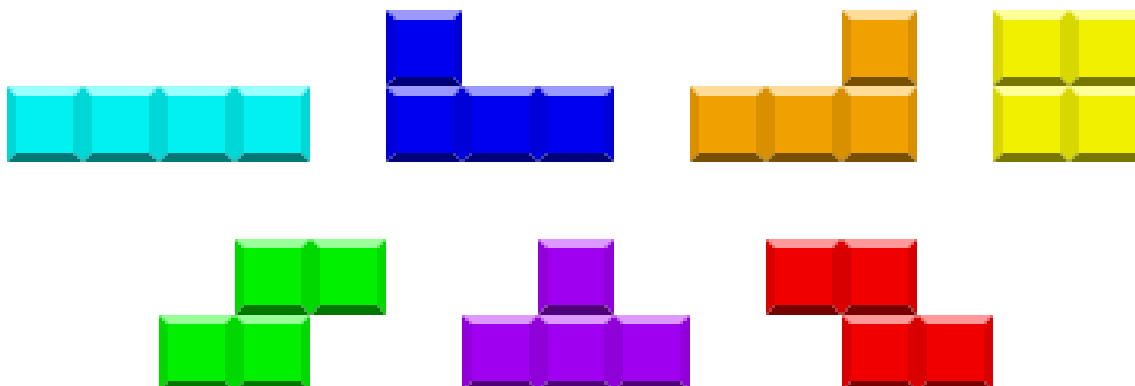
توضیحات بازی

حتما با بازی تتریس آشنا هستید، ولی محض اطمینان یه دور نحوه بازی و قوانینش رو توضیح می‌دیم.

تتریس یک بازی تک‌نفره‌ست که ابتدا با یک صفحه خالی شروع می‌شه و با مرور زمان، بلوک‌های رنگی از بالای صفحه سقوط می‌کنن و این وظیفه بازیکنه که محل فرودشون رو مشخص کنه. با ورود هر بلوک به صفحه، بازیکن توانایی این رو داره که به چپ یا راست حرکتش بده یا 90 درجه هر کدوم بچرخونه. تا زمانی که بلوک فرود بیاد، توانایی جابه‌جایی و چرخش رو داره ولی به محض نشستن دیگه تغییری نمی‌کنه و بلوک بعدی وارد صفحه می‌شه.



همه بلوک‌های بازی از 4 مربع کوچک تشکیل شدن و 5 شکل متفاوت از شون ساخته می‌شه. البته دقت کنید که دو تا از این 5 شکل، قرینه هم دارند پس دو نوع قطعه L شکل و دو نوع قطعه Z شکل داریم:



شکل هر بلوک به صورت تصادف تعیین می‌شه و علاوه بر این رنگ هر بلوک رو هم باید رندم تعیین کنید که در مورد این براتون بیشتر توضیح می‌دیم. علاوه بر حرکت به چپ و راست و چرخش، دکمه‌ای هم برای افزایش سرعت سقوط بلوک وجود داره. وقتی هم که یه بلوک وارد صفحه بازی شد، در کنار صفحه اصلی بلوک بعدی به بازیکن نمایش داده می‌شه.

هر وقت که بازیکن بتونه بلوک‌ها رو جوری کنار هم بچینه که یک ردیف افقی کامل پر بشه، اون ردیف حذف می‌شه و به امتیاز بازیکن اضافه می‌شه. هرچقدر تعداد ردیف‌هایی که با هم حذف میشن بیشتر باشه، امتیاز بیشتری دریافت می‌کنه.

نتریس در حقیقت پایانی نداره و تا زمانی که ارتفاع بلوک‌های صفحه به سقف برسه (طوری که نشه بلوک دیگه‌ای روش قرار داد) ادامه پیدا می‌کنه. در این صورت، بازیکن می‌بازه و امتیازش وارد لیدربرد میشه. شما باید تایم بازی هر بازیکن رو هم ذخیره کنید.

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 بشه:

- **New Game**
- **How to Play**
- **Leaderboard**
- **Exit**

- بازی باید شامل **حداقل دو سطح سختی متفاوت** باشد. اینکه چجوری بازی رو سخت‌تر کنید رو خودتون باید در موردش فکر کنید! وقتی که بازیکن گزینه New Game رو انتخاب کرد، پلیمر ابتدا باید اسمش رو ورودی بده و بعد سطح سختی رو مشخص کنه.
- سائیز زمین بازی باید **داینامیک** باشه، یعنی به صورت یک مستطیل به طول و عرض دلخواه که قبل از شروع بازی باید این دو پارامتر رو از بازیکن ورودی بگیرید. مینیمم سائیز صفحه بازی هم مستطیلی به طول و عرض 5 است.
- با شروع بازی، به صورت تصادفی یکی از **7 نوع بلوک** با یک رنگ دلخواه از بالای صفحه به پایین حرکت می‌کنه. همچنین باید شکل و رنگ **بلوک بعدی** رو هم بیرون صفحه بازی نمایش بدید.
- بلوک باید به صورت خودکار به سمت پایین حرکت کنه اما بازیکن باید بتونه سرعت حرکت رو به پایین رو افزایش بده و یا بلوک رو به **چپ و راست حرکت** بده. همچنین بازیکن میتونه بلوک رو 90 درجه به چپ (یا راست) **دوران** بده. دقت کنید که چرخش فقط در صورتی باید انجام بشه که بلوک پس از دوران با بلوک‌های دیگر برخورد نداشته باشه. مرکز دوران هر بلوک رو هم باید از قبل مشخص کرده باشید.
- بعد از قرار گرفتن یک بلوک در صفحه باید بررسی کنید که **اگر ردیفی کامل پر شده**، اون ردیف رو از صفحه حذف کنید و تمام بلوک‌هایی که بالای اون ردیف قرار داشتند رو یک واحد به سمت پایین حرکت بدید.
- بعد از حذف ردیف، متناسب با ابعاد زمین و تعداد ردیف‌هایی که همزمان حذف شدن باید به امتیاز بازیکن اضافه کنید. در کل برای **امتیازدهی** از خلاقیت خودتون و هر پارامتر دیگه‌ای که نیاز دارید می‌تونید استفاده کنید.
- باید برای بازی یک **منوی Pause** هم طراحی کنید و یک کلید رو برای pause کردن بازی در نظر بگیرید. بعد از unpause شدن بازی، پلیمر باید بتونه دقیقاً از جایی که بازی رو رها کرده ادامه بده.
- همونطور که گفتیم بازی پایانی نداره، ولی **باخت** در بازی باید وقتی اتفاق بیوفته که بلوکی که وارد صفحه شده دیگه جایی برای پایین اومدن نداشته باشه. در این صورت باید امتیاز بازیکن رو ثبت کنید و به منوی اصلی برنامه برگردید.
- در قسمت **How to Play**، می‌تونید یه توضیح کوتاه از نحوه بازی قرار بدید ولی مهم‌تر از اون باید کنترل‌های بازی رو آموزش بدید.

- بعد از پایان بازی باید با استفاده از فایل اسم و امتیاز و زمان بازیکن رو ذخیره کنید و با استفاده از این فایل لیدربرد بسازید. برنامه‌تون ابتدای اجرا باید چک کنه که آیا فایلی برای لیدربرد وجود داره یا نه، اگر فایل لیدربرد وجود نداره باید یه فایل جدید ساخته بشه.
- اگر فایل لیدربرد موجوده برنامه باید اطلاعات لیدربرد رو از همون فایل بخونه و بعدا هم دوباره همون فایل رو آپدیت کنه. اگر اسم شخصی قبلا در لیدربرد اومده بود، در صورتی که امتیازی که به دست آورده بیشتر از امتیاز قبلیه باید در سطر مربوطه از فایل امتیاز و زمان جدید شخص رو آپدیت کنید.
- لیدربرد شامل رنکینگ افرادی که در بازی شرکت کردن، با انتخاب گزینه **Leaderboard** باید اسم بازیکن‌ها رو به صورت مرتب شده برحسب زمان و امتیاز نمایش بدید. یعنی اگر دو بازیکن امتیاز برابر داشته باشن، رتبه بازیکنی بالاتره که زمان کمتری داشته. طبیعتا برای سطوح سختی متفاوت باید لیدربردهای جداگانه در نظر بگیرید.

فازبندی

شما به دلخواه خودتون میتونید پروژه رو از هرجایی که مناسب دیدید شروع کرده و کامل کنید. فازبندی زیر صرفا یک پیشنهاد برای شماست که اگر سردرگم هستید از اون استفاده کنید. توجه کنید که استفاده از این فازبندی کاملا اختیاری بوده و این قسمت فقط برای راهنمایی شماست.

- **فاز ۱:** در پایان این فاز بازی شما باید شامل منو باشد و بازی را بتوان در ابعاد ثابت زمین و یک درجه سختی انجام داد و توقف کرد.
- **فاز ۲:** در پایان این فاز هر بازی دارای امتیاز بوده و میتوان آن را ذخیره کرد. همچنین بازی را باید بتوان در ابعاد دلخواه و حداقل دو درجه سختی انجام داد.
- **فاز ۳:** در پایان آخرین فاز بازی شما شامل سطح بندی و لیدربرد است. لیدربرد شما باید توانایی به روز شدن داشته باشد.
- **فاز ۴:** این فاز شامل پیاده سازی قسمت های امتیازی است.

نکات

شما اجازه دارین از تمامی مطالبی که در کلاس درس استاد و همین‌طور در کلاس‌های ورکشاپ بهتون آموزش داده شده استفاده کنین. شرطها، حلقه‌ها، آرایه‌ها، پوینتر، کار با فایل و استراکت از جمله ابزارهایی هستن که می‌تونین برای انجام پروژه ازشون استفاده کنین.

لیست کردن دقیق تمامی ابزارهایی که اجازه دارید ازشون استفاده کنید کار راحتی نیست، لذا اکیدا پیشنهاد می‌شه هرچیزی خارج از مطالب کلاس رو نیاز داشتنین، حتما با منتور پروژمتون در میون بگذارین تا بعدا مشکل‌ساز نشه.

اما ابزارهایی که اجازه‌ی استفاده ازشون رو هیچ‌جوره ندارید:

• کلیدواژه‌ی auto

هر چه قدر هم که به نظرتون ساده باشه، از شما انتظار میره روی data type ها مسلط باشین؛ لذا حق استفاده از این کلید واژه رو ندارین و در صورت مشاهده نمره ازتون کسر میشه.

• موتورهای بازی‌سازی (Game Engine)

حق استفاده از موتورهای بازی‌سازی که ابزارهای پیش‌ساخته رو در اختیارتون قرار می‌دن ندارین، بلکه ازتون انتظار می‌ره بتونین خودتون توابع و ابزارهاتون رو از صفر بسازین.

• کتابخانه‌های OpenGL و SFML

کارهای گرافیکیتون رو می‌بایستی فقط با استفاده از کاراکترهای اسکی و یونیکد انجام بدین و نمی‌تونین رابط‌های گرافیکی با این کتابخونه‌ها بنویسین. تمامی کد شما باید در محیط ترمینال اجرا بشه.

• هرگونه API

برنامه‌تون باید به خودی خود همه کاری رو انجام بده، برنامه‌ی جداگونه، api یا هر چیزی که خودتون کدش رو ندارین رو نمی‌تونین استفاده کنین.

نکته بسیار مهم: هرچیزی که مورد استفاده‌تون هست رو باید بلد باشین و بتونین حین ارائه در موردش توضیح بدین و دلیل استفاده ازش رو هم بیان کنید!

ابزارها و کتابخانه‌های کاربردی

• یادآوری Unicode:

برای نمایش کاراکتر هاتون توی ترمینال هم دقیقا مثل پروژه 2048 یکی از یونیکدهایی که می‌تونید استفاده کنید کاراکترهای [box drawing](#) هست و به صورت کلی باید تجربه هاتون توی اون پروژه با یونیکد هارو اینجا هم به کار بگیرید.

برای خلاقیت بیشتر هم میتونید توی برنامه تون از ASCII Art هم استفاده کنید که این بخش رو چون بیشتر امتیازی حساب میشه به عهده خودتون میذاریم:)

• کتابخانه chrono:

ممکنه تو پروژه‌تون نیاز به محاسبه تایم اجرای یک تابع یا یک فرآیند رو داشته باشید یا حتی بخواهید تایمی که بازی در حال اجرا بوده رو نمایش بدید. برای این جور کارها باید از یه کتابخونه به اسم chrono استفاده کنید که توابعی برای طول زمان یا همون "duration" و یا ذخیره زمان در لحظه فراخوانی همان تابع دارد. از اونجایی که کتابخانه chrono تعداد تابع‌های زیادی داره براتون لینک document ها که تمام توابع رو شامل میشن براتون قرار دادیم:

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

• مفاهیم‌های Frame, FPS, Delta Timing:

برای نمایش دادن گرافیکی که با یونیکدها توی ترمینال ساختید، باید رابط کاربریتون رو برای هر تغییر که ایجاد می‌کنید با استفاده از یک لوپ while (که بهش game loop هم می‌گیم) تا زمانی که بازی ادامه داره چاپ و پاک کنید به هرکدوم از این چاپ کردن ها به اصطلاح یک فریم می‌گیم.

سرعت اجرا شدن هر دور از این حلقه‌ها روی هر سیستمی ممکنه که فرق بکنه. برای مثال یک سیستم قوی ممکنه خیلی سریع تر هر فریم رو نمایش بده که سرعت بازی رو می‌بره بالا و این باعث میشه که رابط کاربری شما دچار مشکلی به اسم flicker که همون چشمک زدن بیش از حد هست بشه.

برای همین شما می‌تونید مفهومی به اسم delta timing رو به کار ببرید. این یک مفهوم بسیار کاربردی توی بازی‌سازی که ابتدا یک عدد مشخص که می‌خواهیم fps یا همون تعداد فریم بر ثانیه باشه رو به عنوان یک constant تعریف می‌کنیم. بعد باید فاصله زمانی بین چاپ شدن دو فریم رو محاسبه کنیم. یعنی

مثلا اگر می‌خواهیم 4 فریم بر ثانیه بگیریم، فاصله بین هر فریم میشه 0.25 ثانیه. به این فاصله زمانی **delta time** یا **frame time** می‌گیم.

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

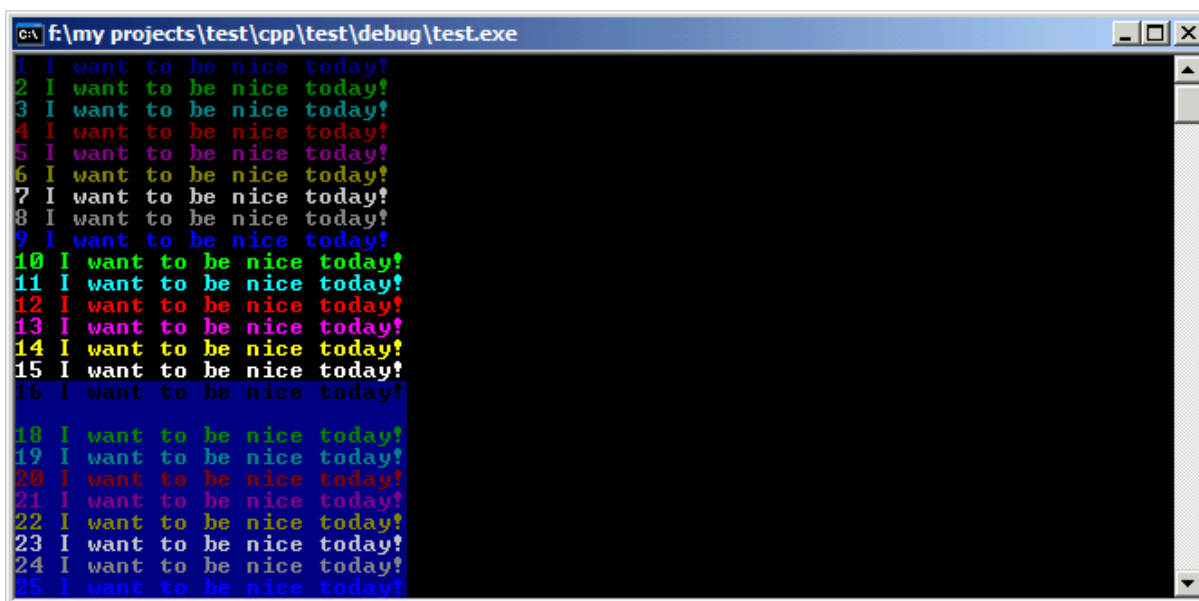
• تغییر رنگ خروجی در ترمینال:

برای بهتر کردن رابط گرافیکی تون باید از رنگ‌هایی که ترمینال‌ها دارند هم استفاده کنید. مثلا برای ویندوز برای تغییر رنگ خروجی cout، می‌تونید از تابع زیر در کتابخانه windows.h استفاده کنید.

```
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 کد بالا تصویر زیر است. یکی از مشکلاتی ممکنه بهش برخورد کنید اینه که برخی رنگ‌ها اون شکلی که در تصویر پایین می‌بینید نباشند. برای رفع این مشکل ممکنه نیاز باشه محیطی که به عنوان ترمینال استفاده می‌کنید رو عوض کنید.

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



● Git:

گیت ابزاریه که با استفاده از اون میتونید پروژه خودتون رو مدیریت کنید. کدهاتون رو میتونید تو صفحه گیت‌هاب خودتون آپلود کنید و به طور همزمان با همگروهیتون به پروژه دسترسی داشته باشید و دیگه کمتر درگیر دردسرهای انتقال درست کد بشید. هرچند که گیت خودش میتونه پیچیده باشه و استفاده ازش **اختیاریه**. در آینده با این ابزار به طور رسمی‌تر آشنا می‌شید ولی اگر علاقه دارید که برای این پروژه استفاده کنید، یادگیریش کامل به عهده خودتونه.

فیچرهای امتیازی پیشنهادی

- میتونین با گذشت زمان به طور خودکار سرعت پایین اومدن بلوک‌ها رو بیشتر کنین.
- انجام بازی به صورت دو نفره (Player vs Player). هر بازیکن میتونه در نیمی از صفحه بازی کنه یا بازیکن‌ها در دو دستگاه جدا بازی کنن.
- بازی رو تا هر مرحله‌ای که پیش رفتین Save کنین و بعد که برگشتین بتونید بازی قبلی رو از ادامهش Load کرد. Save ها رو هم باید داخل فایل ذخیره کنید.
- گیم مودهای جدید: مثلاً هرگاه بلوکی جدید روی بلوکهای قبلی با رنگ یکسان قرار گرفت، هم آن بلوکها و هم یک لایه دور آن را نابود کند. (مثل بازی Candy Crush)
- یک منوی Pause پیچیده‌تر طراحی کنین که آپشن‌های Restart و Exit داشته باشه. اگر توانایی Save و Load کردن بازی رو هم پیاده‌سازی کردید، از این منو می‌تونید بیشتر استفاده کنید.

ارزیابی

موارد زیادی برای ارزیابی کدتون در نظر گرفته می‌شه، 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 هستن، که اولی به این معنی هست که تکه‌های کدتون رو تکرار نکنید، و اگر به یک کد بیشتر از یک بار نیاز دارید، اون رو تبدیل به فانکشن کنید.

دومی هم به این نکته اشاره می‌کند که تا جای ممکن بهتره که 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 داره، ساده ترین راه رو انتخاب کنید.

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

• معماری کد

بهتره که بخش های مختلف پروژه رو جدا کنید و مجزا پیاده سازی کنید، مثلا منطق بازی، بخش گرافیک، بخش دسترسی به فایل و غیره. اینجوری برای دیباگ کردن کد هم کارتون ساده تره.

• کامنت گذاری (به خصوص برای توابع و سکشن های مختلف کد)

• همکاری و تقسیم کار درست (در صورتی که پروژه رو گروهی انجام می‌دید).

ارائه پروژه به صورت حضوری و از کل اعضای تیم انتظار میره که به **همه ی بخش های پروژه** مسلط باشند و بدون هر فاندکشن و هر خط کد چه نقشی داره.

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

ددلاین و تایم ارائه

برای تحویل پروژه تا هفته بعد امتحان پایان ترم درس مبانی برنامه سازی فرصت دارید. یعنی تا پایان روز **چهارشنبه 2 بهمن** باید پروژه رو به صورت یک فایل ZIP ایمیل کرده باشید.

حتما در کنار فایل های مربوط به خود برنامه، چندتا اسکرین شات از برنامه تون در حال اجرا هم قرار بدید. اگر رپورت یا توضیحی هم در مورد پروژه تون نوشتید می‌تونید ضمیمه کنید.

نکته نهایی: ارائه پروژه به صورت حضوری در روز **شنبه 7 بهمن** انجام خواهد شد. حضور همه ی اعضای تیم برای ارائه اجباریست.