مبرَکبه

فاز اول پروژه پایانی

ترم ۴۰۲۲

استاد:

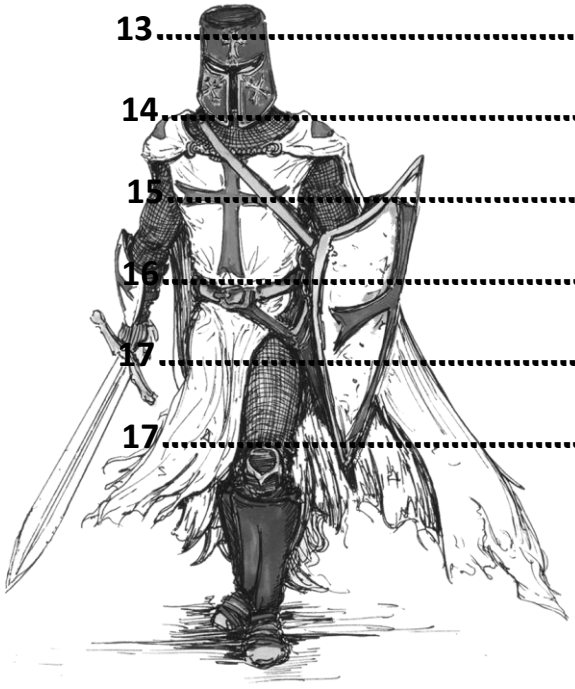
مهدی سخایی نیا

اعضا تیم:

مهرداد ورمزیار

محمد سینا ناظمی

فهرست:



مقدمه :

با سلام

در این فاز از پروژه قصد داریم 3 کارت جدید و قابلیت ذخیره سازی و بارگزاری بازی را به پروژه اضافه کنیم .

در مرحله اول تصمیم گرفتیم که ابتدا کد کلاس های جدید رو به پروژه قبلی اضافه کنیم و سپس گرافیکی کردن بازی رو انجام بدیم .

به طور کلی برای 3 کارت بنفش جدید 3 کلاس جدید ایجاد کردیم که از کلاس Card ارث بری می کنند و برای ذخیره و بارگزاری بازی از طریق txt فایل کلاس Data را ایجاد کردیم که در ادامه روش کار توضیح خواهم داد .

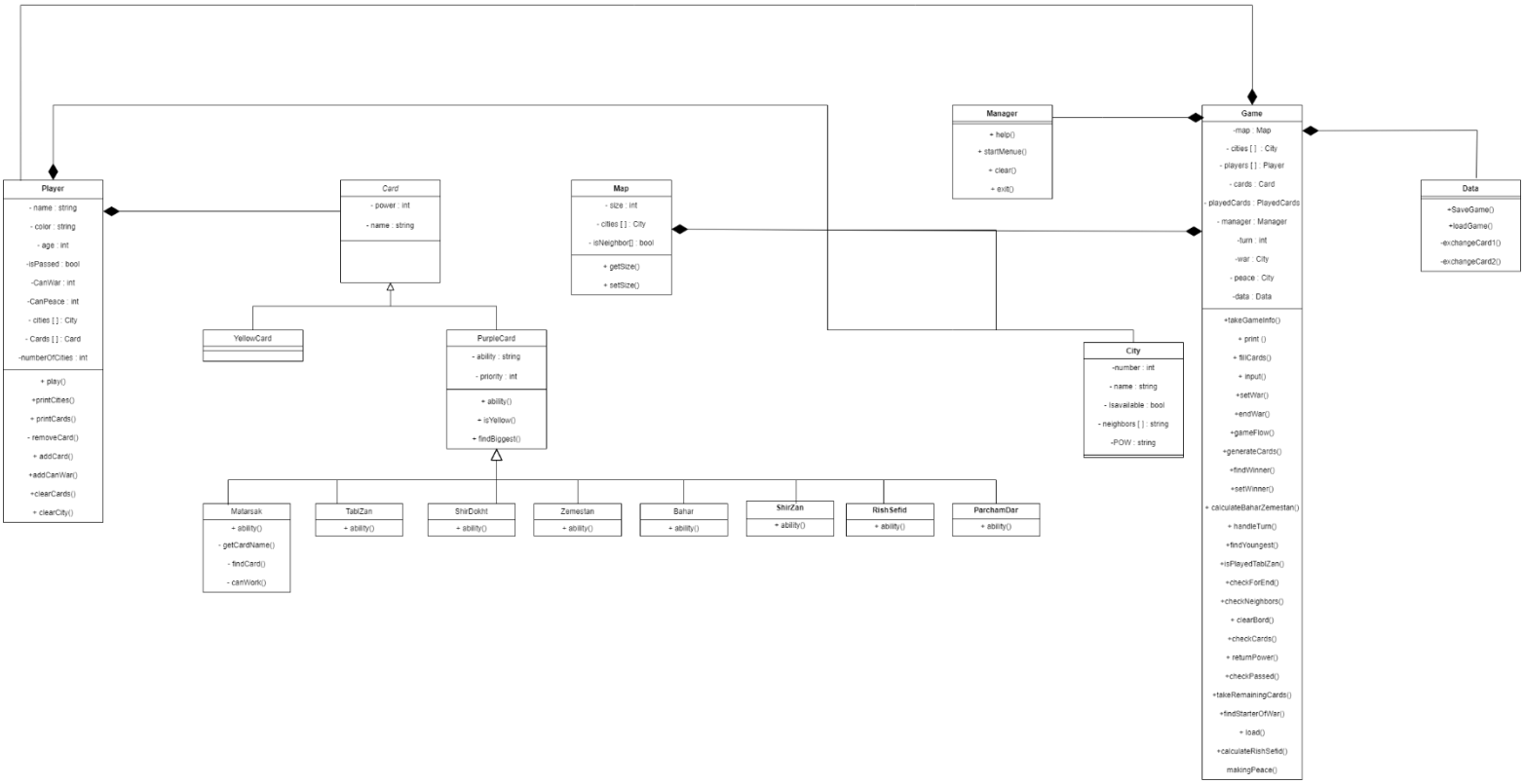
در روند این گزارشکار ابتدا UML و همانند گزارشکار فاز 1 یکی یکی کلاس های جدید را شرح می دهیم و در ادامه به گزارش نحوه گرافیکی کردن می پردازیم .

گیت :

در روند این پروژه از گیت استفاده خواهیم کرد .

لینک

:UML


Player
- name : string
- color : string
- age : int
-isPassed : bool
-CanWar : int
-CanPeace : int
- cities [ ] : City
- Cards [ ] : Card
-numberOfCities : int
+ play()
+printCities()
+ printCards()
- removeCard()
+ addCard()
+addCanWar()
+clearCards()
+ clearCity()
Card
- power : int
- name : string
YellowCard
PurpleCard
- ability : string
- priority : int
+ ability()
+ isYellow()
+ findBiggest()
Matarsak
+ ability()
- getCardName()
- findCard()
- canWork()
TabiZan
+ ability()
ShirDokht
+ ability()
Zemestan
+ ability()
Bahar
+ ability()
ShirZan
+ ability()
RishSefid
+ ability()
ParchamDar
+ ability()
Map
- size : int
- cities [ ] : City
- isNeighbor[ ] : bool
+ getSize()
+ setSize()
Manager
+ help()
+ startMenue()
+ clear()
+ exit()
City
-number : int
- name : string
- isavailable : bool
- neighbors [ ] : string
-POW : string
Game
-map : Map
- cities [ ] : City
- players [ ] : Player
- cards : Card
- playedCards : PlayedCards
- manager : Manager
-turn : int
-war : City
- peace : City
-data : Data
+takeGameInfo()
+ print ()
+ fillCards()
+ input()
+setWar()
+endWar()
+gameFlow()
+generateCards()
+findWinner()
+setWinner()
+ calculateBaharZemestan()
+ handleTurn()
+findYoungest()
+isPlayedTabiZan()
+checkForEnd()
+checkNeighbors()
+ clearBord()
+checkCards()
+ returnPower()
+checkPassed()
+takeRemainingCards()
+findStarterOfWar()
+ load()
+calculateRishSefid()
makingPeace()
Data
+SaveGame()
+loadGame()
-exchangeCard1()
-exchangeCard2()

کلاس های جدید:

کلاس ShirZan :

رفتار ها :

- **سازنده**
- **اجرا توانایی**

توانایی این کلاس یک پارامتر از نوع Player را به صورت refrence می گیرد و توانایی آغاز جنگ بازیکن دریافتی را افزایش می دهد و در نهایت در Game یک مقایسه بین بازیکن ها و توانایی آغاز جنگ آنها صورت می گیرد و در نهایت یک بازیکن طبق معیار ها مقایسه به عنوان آغازگر جنگ تعیین می شود .

چالش : یک چالش حدی که در این کلاس داشتیم این بود که هنگام مقایسه توانایی جنگ بازیکن ها اگر دو بازیکن توانایی برابری داشتند برنده بازی باید محل جنگ بعدی را تعیین کند از همین جهت افزایش توانایی جنگ را ضریبی از 2 کردیم و برنده توانایی جنگ فرد دارد که به راحتی می توان برنده را تشخیص داد .

کلاس RishSefid :

رفتار ها :

- سازنده
- اجرا توانایی

در شرح کلی توانایی این کلاس تنها هدف 0 کردن قدرت قوی ترین کارت بازی شده است اما کاربرد دیگر این کارت ارائه توانایی صلح به یک بازیکن است که این امر در توانایی کلاس پیاده نشده و در کلاس Game آخرین بازیکنی که این کارت را بازی کرده پیدا می کنیم و نشان صلح را به او می دهیم .

چالش : به طور کلی چالش این کلاس پیاده سازی نشان صلح بود که با بررسی هایی که انجام دادیم ترجیح شد که ارائه این نشان در کلاس Game باشد .

کلاس ParchamDar :

رفتار ها :

- **سازنده**
- **اجرا توانایی**

این کارت به محض بازی شدن جنگ را پایان می دهد که ما در توانایی این کارت refrence از بازیکن ها پاس می دهیم و isPassed همه را true می کنیم تا جنگ پایان پیدا کند .

چالش : این کلاس چالشی نداشت .

کلاس Data :

رفتار ها :

- **ذخیره بازی**
- **بارگزاری بازی**
- **جابه جایی کارت 1 و 2**

این کلاس با کمک یک فایل متنی در متود SaveGame مشخصات بازیکن ها و کارت های بازی شده و نوبت و صلح و جنگ را ذخیره می کند و در متود loadGame این اطلاعات را از فایل خانده در یک struct از نوع GameData که اطلاعات بازی را دارا است بر می گرداند و در Game یک attribute از کلاس Data می سازیم و اطلاعات را در بازی جاگذاری می کنیم و بازی به جریان در می آید .

چالش : خواندن اطلاعات از فایل به کمک sstream دشواری زیادی داشت و به logical error های فراوانی بر خوردیم اما در نهایت کلاس به درستی تکمیل شد . [ارور های مختلف را می توانید در commit های github مشاهده کنید]

تغییرات فاز سوم :

1. استان جدید
2. جابه جایی 2 استان
3. زمستان ضعیف می شود
4. طبل رو بزن
5. بهار زیبا
6. نوبت کیه؟
7. رخش سفید
8. شیرین عقل
9. کارت فک سفید

تمام این تغییرات ابتدا در ورژن کنسول اعمال شدند و در مرحله گرافیکی سازی این ورژن ترمینال نهایی گرافیکی می شود.

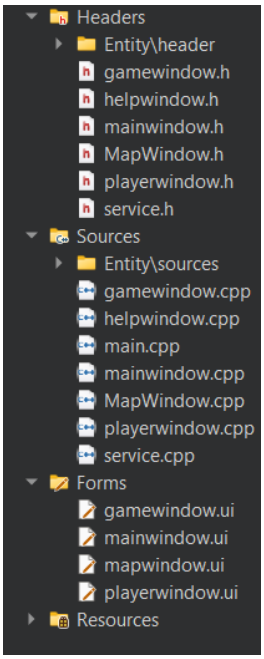
گرافیکی سازی بازی :

در بخش گرافیکی سازی پروژه از نرم افزار QT کمک گرفتیم که در ادامه به بررسی چالش ها و نحوه پیاده سازی می پردازیم .

تغییرات لازمه :

- تمامی بخش ها و توابع کد که مسئولیت input و output در ترمینال داشتند حذف می شوند تا این مسئولیت به گرافیک داده شود .
- String ها به Qstring تغییر داده می شوند .
- Save و load بر روی slot های بیشتری انجام می شود.
- تابع gameFlow که مسئولیت برقراری ارتباط بین توابع را داشت حذف و این عملکرد به گرافیک واگذار می شود .

پنجره های ایجاد شده :



- منوی اولیه [main window]
- منوی دریافت اطلاعات بازیکن ها [playerWindow]
- منوی دریافت نام شهر محل جنگ و نمایش مپ [mapWindow]
- منوی بازی [gameWindow]

به طور کلی روند نمایش پنجره ها به این صورت است که ابتدا پنجره اصلی نمایش داده می شود و سپس با استارت بازی منوی دریافت اطلاعات و مپ نمایش داده شده و در نهایت منوی game نمایش داده می شود و بازی به جریان در می آید .

در این میان اشاره ای هم به service داشته باشیم بد نیست :

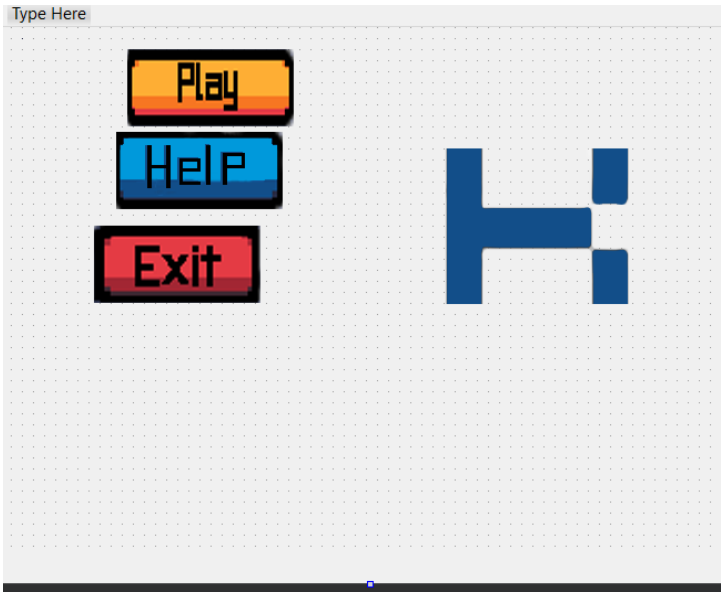
این کلاس مسئول تبادل اطاعات میان تمامی این پنجره ها و کلاس هاست و داده هایی از نوع کارت ها و بازیکن ها و .. دارد و از این کلاس در تمامی form های دیگر شئ ایجاد می کنیم .

منوی اولیه :

در منوی اولیه بازی کاربر لوگو بازی به همراه 4 دکمه شامل شروع و بارگزاری و راهنما و خروج می شود که می توانید در تصویر زیر مشاهده کنید که در صورت شروع فرم دریافت اطلاعات از بازیکنان باز می شود و سپس با توجه به اطلاعات دریافتی بازی جدیدی ایجاد می شود و در صورت بارگزاری کاربر می تواند بازی قبلی خود را ادامه دهد که اطلاعات از فایل txt خوانده می شود.

چالش :

این form چالش جدی ای نداشت جز یک مورد مهم که برای تعیین موقعیت دکمه ها در وسط صفحه بود که از Qscreen کمک گرفتیم و موقعیت دکمه هارا در وسط صفحه قرار دادیم.



منوی بازیکن :

در این منو ابتدا تعداد بازیکن هارا تعیین می کنیم و سپس با توجه به تعداد اطلاعات را دریافت می کنیم و از service برای تبادل اطلاعات استفاده می کنیم .

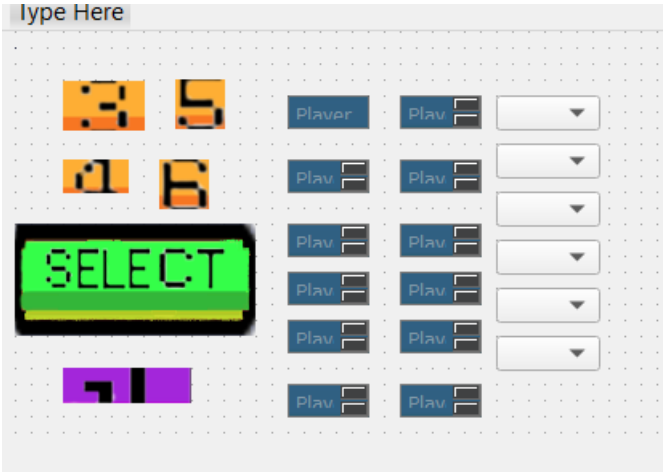
در نهایت با کلیک بر روی دکمه play وارد پنجره map می شویم .

چالش :

این پنجره چالشی مشابه منوی اولیه برای تعیین موقعیت textline ها داشت

و همچنین چالش دیگر برای تعداد بازیکن های کمتر از 6 تا برای دریافت

ورودی داشت که از visible و invisible کردن کمک گرفتیم .



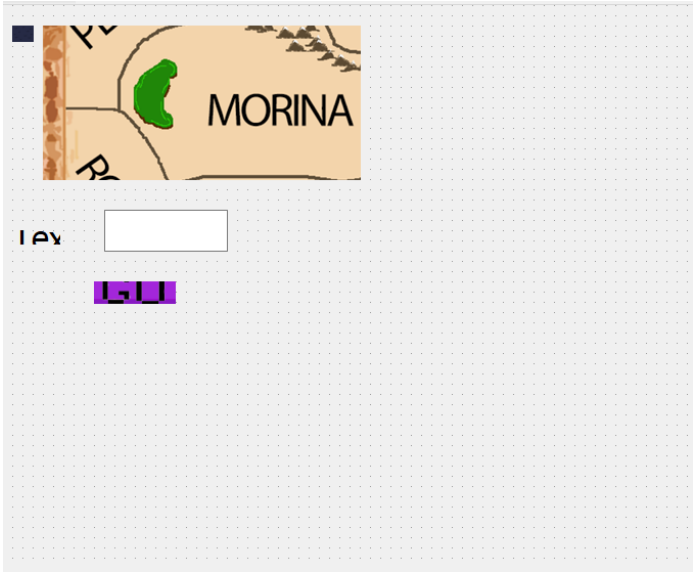
منوی نقشه :

این پنجره map بازی را به بازیکن نمایش می دهد و از جوان ترین بازیکن در ابتدا سپس با توجه به شرایط از سایر بازیکن ها درخواست می کند محل جنگ را تعیین کنند .

و در نهایت با زدن دکمه Go بازی آغاز می شود .

چالش :

این کلاس چالش جدی ای نداشت و فقط نیاز بود تا متود پیدا کردن جوان ترین بازیکن و به تبع برنده و ... را در این کلاس پیاده کنیم .

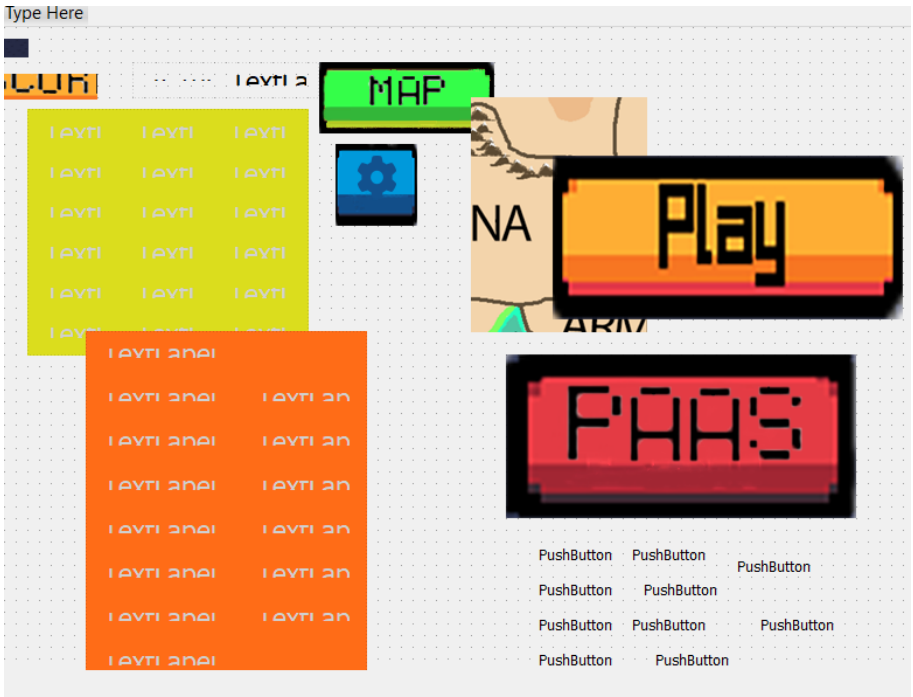


منوی بازی :

این کلاس و پنجره جریان کلی بازی را هدایت می کند و کلاس game در آن ایجاد شده است و به صورت نوبت به نوبت کارت های بازیکن هارا نمایش می دهد و دکمه هایی از قبیل نمایش نقشه یا امتیاز را دارد تا در طول پروژه بازیکن بتواند امتیاز خود را برسی کند و همچنین کارت ها بازیکن که از نوع دکمه اند ایجاد می شوند .

چالش :

- ایجاد tween برای کارت ها
- تعیین موقعیت ها
- مدیریت بازی به جای تابع gameflow



جمع بندی :

به طور کلی تجربه گرافیک یکی از دشوار ترین تجربه های برنامه نویسی برای تیم ما بود اما سعی شد تا حد قابل قبولی پیاده سازی شود و اما سایر بخش ها به طور تمام کمال شامل 3 کارت جدید و ذخیره بازی بر روی فایل انجام شده .

منابع :

- هوش مصنوعی
- Stackoverflow
- Geekforgeeks
- youtube