بسمه تعالی

مستندات پروژه

Mobile RPG Battle System (Unity)

۱. معرفی پروژه(Project Overview)

این پروژه یک پروژه تست برای پیاده‌سازی ساده اما ساختاریافته از سیستم مبارزه یک بازی RPG موبایلی است

هدف اصلی این پروژه:

- **نمایش نحوه برخورد با مسئله**
- **طراحی معماری مناسب**
- **پیاده‌سازی گیم‌پلی نوبتی**
- **مدیریت جریان بازی**
- **و رعایت اصول برنامه‌نویسی تمیز در Unity و#C**

تمرکز پروژه روی گیم‌پلی روان، بدون باگ، معماری واضح و قابل توسعه بوده و تمرکز خاصی روی گرافیک یا آرت پیشرفته نداشته است.

---

۲. پلتفرم هدف و محیط توسعه(Target Platform & Environment)

- **Game Engine: Unity**
- **Unity Version: 6000.3.0f1**
- **Programming Language: C#**
- **Target Platform: Android**
- **Input Support:**
  - **Touch (موبایل)**
  - **Mouse (Editor)**

پروژه به‌گونه‌ای طراحی شده که هم در Unity Editor و هم روی دستگاه اندرویدی بدون مشکل اجرا شود.

---

۳. نقش توسعه‌دهنده (Developer Responsibility)

این پروژه به‌صورت کاملاً انفرادی توسعه داده شده است.

تمام مراحل شامل:

- **طراحی معماری**
- **پیاده‌سازی منطق بازی**
- **توسعه سیستم مبارزه**
- **مدیریت داده‌ها**
- **پیاده‌سازی UI**

طراحی UX

- **تست و دیباگ**

---

۴. توضیح کلی گیم‌پلی(Core Gameplay Flow)

جریان اصلی بازی به صورت زیر است:

1. **بازیکن وارد صفحه اصلی می‌شود.**
2. **سپس وارد صفحه مشاهده کارت های انتخابی خود می شود.(User Deck Cards)**
3. **بازیکن دقیقاً ۳ قهرمان را از میان قهرمان‌های قابل انتخاب خود انتخاب می‌کند.**
4. **پس از انتخاب، امکان شروع مبارزه فعال می‌شود.**
5. **در صفحه ی انتخاب هر قهرمان با نگه داشتن انگشت/کلیک به مدت دو ثانیه می تواند جزئیات هر قهرمان را مشاهده کند.**
6. **میدان نبرد نمایش داده می‌شود:**
    - **۳ قهرمان بازیکن در سمت چپ**
    - **۱ قهرمان دشمن در سمت راست**
7. **مبارزه به صورت نوبتی انجام می‌شود:**

- ابتدا نوبت بازیکن
- سپس نوبت دشمن

۸. بازیکن با لمس یک قهرمان، دستور حمله می‌دهد.

۹. دشمن به صورت تصادفی به یکی از قهرمانان زنده حمله می‌کند.

۱۰. در هر لحظه فقط یک حمله انجام می‌شود و تا پایان انیمیشن، ورودی جدید پذیرفته نمی‌شود و علامت انتظار گوشه صفحه نمایان می شود.

۱۱. مبارزه زمانی پایان می‌یابد که:

- تمام قهرمان‌های بازیکن از بین بروند (باخت)
- یا دشمن از بین برود (برد)

۱۲. در صورت پیروزی ، تجربه ی قهرمان های کاربر که هنوز زنده هستند ۵ واحد افزایش می یابد.

۱۳. در صورت برد یا باخت (اتمام بازی در هر صورت) یک واحد به میزان تجربه ی بازیکن اضافه می شود که به ازای هر ۳ واحد یک قهرمان به صورت رندوم قفل گشایی می شود. بازیکن نهایتا ۱۰ قهرمان فعال می تواند داشته باشد (۳ قهرمان رایگان و ۷ قهرمان جایزه).

---

۵. سیستم مبارزه(Battle System)

سیستم مبارزه به صورت Turn-Based طراحی شده است.

ویژگی‌ها:

- فقط یک حمله در هر لحظه اجرا می‌شود.
- تا پایان انیمیشن حمله، ورودی بازیکن غیرفعال است.
- نوبت‌ها توسط BattleManager کنترل می‌شوند.
- حملات دشمن به صورت تصادفی روی قهرمان‌های زنده انجام می‌شود.

این طراحی از بروز مشکلات همگام‌سازی و حملات همزمان جلوگیری می‌کند.

---

۶. رابط کاربری و تجربه کاربری(UI & UX)

صفحه انتخاب قهرمان

- **لمس یک قهرمان → انتخاب**
- **وضعیت انتخاب به‌صورت بصری مشخص می‌شود**
- **نگه داشتن لمس به مدت ۲ ثانیه → نمایش پنجره اطلاعات قهرمان**

میدان نبرد

- **نمایش نام و سلامت قهرمان‌ها و دشمن**
- **نمایش مقدار آسیب به‌صورت اعداد محوشونده**
- **نمایش وضعیت نوبت (نوبت بازیکن / نوبت دشمن)**

پایان مبارزه

- **نمایش صفحه برد یا باخت**
- **نمایش قهرمان قفل گشایی شده**
- **امکان بازگشت به صفحه انتخاب قهرمان و یا رفتن به نبرد بعدی**

---

۷. سیستم پاداش و پیشرفت(Reward & Progression System)

تجربه قهرمان‌ها

- **پس از هر مبارزه‌ی برنده:**
  - **به تمام قهرمان‌های زنده تجربه اضافه می‌شود.**
- **هر ۵ امتیاز تجربه:**
  - **سطح قهرمان ۱ واحد افزایش می‌یابد.**
  - **سلامت و قدرت حمله قهرمان ٪۱۰ افزایش پیدا می‌کند.**
  - **برای جذابیت بیشتر برای قهرمان ها ماکسیسم تجربه تعریف شده است.**

تجربه بازیکن

- **پس از هر مبارزه (برد یا باخت)، تجربه بازیکن افزایش می‌یابد.**

باز کردن قهرمان جدید

- **هر ۳ امتیاز تجربه بازیکن:**
  - **یک قهرمان تصادفی جدید آزاد می‌شود**
- **حداکثر تعداد قهرمان‌ها :۱۰**
- **در ابتدای بازی :۳ قهرمان رایگان**
- **در حین بازی : ۷ قهرمان جایزه**

---

۸. ذخیره‌سازی داده‌ها(Data Persistence)

تمام اطلاعات مهم بازی ذخیره می‌شوند، از جمله:

- **قهرمان‌های آزاد شده**
- **قهرمان های انتخاب شده و ترتیب آنها**
- **تجربه و سطح قهرمان‌ها**
- **تجربه بازیکن**

پس از بستن و اجرای مجدد بازی:

- **بازیکن دقیقاً از همان نقطه قبلی ادامه می‌دهد**

---

۹. معماری و تصمیم‌های فنی(Code Architecture & Design Decisions)

برخی تصمیم‌های کلیدی در طراحی پروژه:

- **جداسازی منطق بازی از رابط کاربری**
- **استفاده ازManager ها برای کنترل جریان بازی**

- **استفاده از دیزاین پترن ها در توسعه**
- **سعی در رعایت اصول SOLID به خصوص اصل تک مسئولیتی بودن کلاس ها**
- **عدم استفاده از فریمورک‌های معماری سنگین(DI / ECS)**
- **تمرکز بر خوانایی، سادگی و قابلیت توسعه**

این تصمیم‌ها باعث شده کد:

- **قابل فهم باشد**
- **به راحتی توسعه یابد**
- **مناسب پروژه‌های موبایلی کوچک تا متوسط باشد**

در این پروژه با توجه به مقیاس کوچک، نیاز به بهینه‌سازی‌های سنگین نبوده اما ساختار به‌گونه‌ای طراحی شده که در صورت افزایش تعداد قهرمان‌ها و دشمن‌ها، امکان بهینه‌سازی وجود داشته باشد.

---

۱۰. ساختار داده و سیستم‌ها(Code Structure & Data Design)

- **سیستم ذخیره اطلاعات قهرمان ها : در این بازی یک ScriptableObject به نام HeroDatabase تعریف شده است که در آن اطلاعات قهرمان های بازیکن و اطلاعات قهرمان های دشمن وارد و ذخیره شده است ، با توجه به ساده بودن پروژه اطلاعات به این شکل ذخیره شده اند اما به صورت حرفه ای تر میتوان از دیتابیس های ساختاریافته مجزا برای قهرمان های بازیکن و قهرمان های دشمن استفاده کرد. برای اطلاعات قهرمان ها از یک ساختاری به نام HeroData استفاده شده است که برای ورود و نگه داری اطلاعات پایه ای هر قهرمان استفاده می شود.**
- **سیستم مدیریت تجربه بازیکن: در این بازی کلاسی به نام PlayerExperienceManager وجود دارد که وظیفه ی مدیریت میزان تجربه کاربر را بر عهده دارد ، همان تجربه ای که پس از اتمام هر بازی به دست می آورد و بر اساس آن قهرمان ها را قفل گشایی می کند.**
- **سیستم مدیریت قهرمان های بازیکن : در این بازی کلاسی به نام PlayerHerosManager وظیفه ی مدیریت قهرمان های بازیکن را بر عهده دارد که در واقع قهرمان های در دسترس و قهرمان های قفل شده را مدیریت می کند و با کمک آن می توان فهمید که کدام قهرمان ها برای بازیکن در دسترس هستند و یا قفل قهرمانی را باز کرد.**

- **سیستم مدیریت تجربه قهرمان ها : کلاسی با نام HeroExprienceManager میزان تجربه ی هر قهرمان را مدیریت میکند که با کمک آن میتوان تجربه ی قهرمان را افزایش داد یا اطلاعات و پارامتر های آن قهرمان را بر اساس تجربه دریافت کرد ، در این بازی هر قهرمان یک سری پارامتر های پایه ای دارد که در حین بازی و افزایش تجربه ی قهرمان ، پارامتر هایی مانند سطح ، طول عمر ، میزان ضربه و... افزایش پیدا می کند و در این کلاس این موارد پس از افزایش تجربه ی قهرمان ، محاسبه و مدیریت می شود. همانطور که این این فایل اشاره شد ، برای جذابیت بیشتر بازی و قهرمان ها ، برای قهرمان ها ماکسیسمم تجربه تنظیم شده است که باعث میشود سایز پارامتر های قهرمان نیز که بر اساس تجربه قهرمان محاسبه می شوند دارای ماکسیسمم بشوند. البته میتوان با تغییرات جزئی در کدنویسی و رابط کاربری ، این محدودیت را به راحتی حذف کرد.**

- **سیستم نمایش قهرمان ها : در این بازی قهرمان هایی برای بازیکن تعریف شده اند که بازیکن می تواند قهرمان های در دسترس را در دسته خود (Deck Cards) قرار بدهد و با آنها وارد نبرد بشود. برای این موضوع کلاس HeroItemsLoader وظیفه ی لود قهرمان ها را بر عهده دارد و کلاس HeroItemsManager نیز وظیفه ی مدیریت قهرمان های لود شده را بر عهده دارد و هر آیتم قهرمان لود شده در UI دارای یک کد به نام HeroItem هستند که وظیفه ی لود و عملیات مربوط به هر آیتم را بر عهده دارد.**

- **سیستم مدیریت کارت ها / قهرمان های انتخاب شده : در این بازی بازیکن سه قهرمان را انتخاب کرده و با آنها وارد مبارزه می شود که به مجموع این سه قهرمان Player Deck می گوییم وهر قهرمان انتخاب شده هم یک Card در نظر می گیریم ، دقیقا همانند بازی Clash royal. در این پروژه کلاس PlayerDeckManager وظیفه ی مدیریت این بخش را بر عهده دارد و کارت های انتخاب کاربر را مدیریت و بارگزاری می کند ، در صفحه ی چیدن کارت ها برای هر کارت اسکریپت PlayerDeckCard وجود دارد که وظیفه ی مدیریت آن کارت و اسلاتی که کارت در آن قرار می گیرد را بر عهده دارد.**

- **سیستم انتخاب قهرمان ها : کلاس HeroPicker فرایند انتخاب قهرمان برای قرار گیری در دسته ی کارت های کاربر را با کمک کلاس PlayerDeckManager انجام می دهد.**

- **سیستم نمایش اطلاعات هر قهرمان : در لیست انتخاب قهرمان ، با نگه داشتن انگشت/کلیک به مدت ۲ ثانیه می توان مشخصات آن قهرمان را در یک pop up به طور کامل مشاهده کرد ، مهم نیست قهرمان قفل یا باز باشد ، می توان این مشخصات را مشاهده کرد. وظیفه ی لود و نمایش مشخصات قهرمان انتخاب شده بر عهده ی کلاسی به نام HeroInformationViewer می باشد. در صفحه ی اول بازی یعنی Lobby کلاس HeroItem دستور نمایش این اطلاعات را می دهد. در صفحه ی بازی نیز با نگه داشتن انگشت/کلیک روی هر قهرمان حتی قهرمان دشمن می تواند مشخصات آن را با کمک همان HeroInformationViewer**

مشاهده کرد و در این بخش کلاس های **PlayerHero** و **EnemyHero** فرایند نمایش و دستور نمایش اطلاعات را انجام می دهند.

- سیستم مدیریت مبارزه : کد **BattleManager** وظیفه ی اجرای مبارزه را بر عهده دارد که پس از لود بازی فرآیند انتخاب دشمن به صورت رندوم و ایجادقهرمان ها و مدیریت فرآیند بازی شامل مدیریت نوبت ها و بررسی پایان بازی را بر عهده دارد. در هنگام ایجاد قهرمان ها ، کلاسی با نام **Spawner** به کمک این کلاس می آیند و فرایند ایجاد هر قهرمان در صحنه بازی را انجام می دهد.

- سیستم مدیریت قهرمان های در صحنه : هر قهرمان توسط یک اسکریپت کلی به نام **Hero** مدیریت می شوند که اطلاعات و شرایط جاری قهرمان را مدیریت می کند و همچنین فرایند ضربه زدن و ضربه خوردن قهرمان را بر عهده دارد ، کلاس های **EnemyHero** و **PlayerHero** نیز از این کلاس ارث بری می کنند و به ترتیب موارد اضافه تر مربوط به قهرمان دشمن و قهرمان بازیکن را بر عهده دارند ، مثلا در **PlayerHero** فرایند انتخاب قهرمان توسط بازیکن برای حمله مدیریت می شود.

- سیستم تعیین هدف : در این بازی هدف حمله قهرمان های بازیکن ثابت و تنها دشمن بازی است ولی هدف حمله ی قهرمان دشمن به صورت رندوم از بین قهرمان های زنده بازیکن انتخاب می شود که این فرایند انتخاب توسط **TargetSelector** انتخاب می شود.

- سیستم حمله : پس از انتخاب هدف حمله ، کلاس **HeroMoveToTarget** وظیفه ی حرکت قهرمان به سمت هدف را بر عهده دارد و پس از رسیدن به آن ، به هدف ضربه وارد کرده و به محل اولیه خود بر می گردد.

- سیستم گرافیکی قهرمان ها : برای یک اسلایدر برای نمایش میزان سلامتی و یک **Text** برای نمایش نام وجود دارد و یک **Text** هم برای نمایش میزان آسیب وارد شده به قهرمان وجود دارد که توسط **HeroUI** مدیریت می شود. پس از ساخته شدن هر قهرمان روی صحنه باید گرافیک یا **Skin** آن نیز لود بشود که این بخش نیز توسط **HeroUI** مدیریت می شود و با کمک **HeroesDatabase** ، **Sprite** های مربوط به قهرمان از پوشه **Resourecs** لود شده و ست می شوند. همچنین برای هر قهرمان ها انیمیشن های **Idle** و**Walk** ، **Attack** و **Die** به صورت کاملا ساده و نمادین پیاده سازی شده است که توسط کلاس **HeroAnimationManager** مدیریت می شود.

- سیستم جایزه : پس از پایان هر نبرد ، در صورت پیروزی ، سطح قهرمان های زنده کاربر افزایش پیدا می کند (به عنوان پیشنهاد اگر این مورد برای قهرمان های دشمن هم اعمال شود باعث جذابیت می شود). پس از پایان هر مبارزه سطح تجربه بازیکن نیز بیشتر شده و قهرمان به او جایزه داده می شود که این دو فرایند که به عنوان جایزه در نظر گرفته شده اند توسط **RewardManager** مدیریت می شوند.

- **سیستم پایان بازی : فرآیند پایان بازی توسط GameEndManager مدیریت می شود که شامل نمایش صفحه ی پایانی و اجرای فرایند جایزه بر اساس نتیجه بازی (افزایش سطح بازیکن و قهرمان های زنده) است.**
- **سیستم مدیریت صفحات بازی : سیستم باز و بسته شدن صفحات و همچنین جابجا شدن بین صحنه های بازی توسط MenuManager مدیریت می شود.**
- **سیستم موزیک و افکت های صوتی : با توجه به اینکه در سناریو بازی توضیحی در مورد موزیک و افکت های صوتی نوشته نشده بود اما یک موزیک پس زمینه و افکت صوتی کلیک روی دکمه ها در بازی پیاده سازی شد که توسط MusicManager و ButtonClickSound مدیریت می شوند . برای افکت های قهرمان ها این امکان بود که افکت های صوتی مانند راه رفتن ، ضربه زدن ، ضربه خوردن و... نیز پیاده سازی شود.**

---

۱۱. نحوه اجرای پروژه(How to Run)

۱. پروژه را در Unity Hub باز کنید

۲. صحنه اصلی را Load کنید

۳. اجرای بازی در Editor یا Build برایAndroid

٤. فایل APK در پوشه خروجی قرار دارد

---

۱۲. ابزارها و منابع جانبی(External Assets & Libraries)

- **هیچ فریمورک معماری خارجی استفاده نشده است.**
- **آرت‌ها و اسپرایت‌ها ساده هستند و از پکیج های در دسترس استفاده شده اند.**
- **تمرکز اصلی روی منطق و ساختار کد بوده است.**

جمع‌بندی نهایی

این پروژه با هدف نمایش:

- **توانایی طراحی سیستم**

- **معماری مناسب**
- **مدیریت گیم‌پلی نوبتی**
- **و پیاده‌سازی بدون باگ**

توسعه داده شده و تمام الزامات خواسته‌شده در توضیحات پروژه را پوشش می‌دهد.

نکته اضافه :

با توجه به این که مدت زمان انجام این پروژه یک هفته بوده و پس از ارسال این پروژه برای بنده ، هنوز درگیر کار های پایانی پروژه ی قبل بوده ام و با تاخیر آن را شروع نمودم و همچنین مشکلات شدید دسترسی به اینترنت و مجبور شدن به استفاده کردن از تصاویر و منابع در دسترس در آن شرایط ، پروژه به صورت پاره وقت انجام شد و مدت زمانی آن طولانی تر از مدت زمان مشخص شده گردید که مدیریت محترم منابع انسانی در جریان این دو مورد قرار داشتند.

با توجه به عدم دسترسی به اینترنت پروژه به صورت local در Github قرار گرفت و پس از دسترسی به GitHub به صورت public آپلود شد و تمامی مراحل Commit شده و با توضیحات مختصر قابل مشاهده هستند.

با تشکر

محمدسعید نصیری