Analiza Obiektowa

Architektura Systemu

Główny silnik(**Engine**) obsługuje moduły “gry” tj. **Menu** - obsługujący menu główne, **Scene** - obsługujący mechanikę gry, **Editor** - obsługujący edytor poziomów. Każdy moduł dalej zajmuje się obsługą powierzonego mu zadania. **Menu** tworzy przyciski na ekranie i obsługuje akcje z nimi związane - przełączanie między modułami, wychodzenie z gry itp., **Scene** wyświetla ekran gry, minimapę, ekwipunek itd. - pozwala grać, **Editor** wyświetla roboczą mapę i kafelki i pozwala ją zmieniać oraz zapisać.

1. **Engine**
   Silnik/Rdzeń łączący moduły. Obsługuje główne ustawienia wyświetlania, połączenie dostępnych modułów, przejścia między nimi, zdarzenia pygame.

2. **Module**
   Klasa bazowa reprezentująca moduł. Każdy moduł powinien obłsugiwać zdarzenie zmiany ustawień ekranu oraz wyświetlanie.

   a. **Menu**
      Klasa obsługująca wyświetlanie menu głównego i obsługi zdarzeń z tym związanych - wybieraniem następnego aktywnego modułu, zmiana rozdzielczości itp.

   b. **Scene**
      Klasa obsługująca główne okno gry, odpowiada za komunikację pomiędzy mapą, minimapą, ekwipunkiem bohatera podczas gry. A także przechowuje aktualny stan gry - aktualne potwory na mapie, wczytany loch.

   c. **Editor**
      Klasa pozwalająca na edycję lochów, jak **Scene** również dba o komunikację między mapą i minimapą a także **TilesGridem** - toolboxem z kafelkami i modyfikatorami do ustawiania na mapie. Udostępnia także możliwość zapisu lochu.

3. **MenuButton(+ FullscreenButton itd.)**
   MenuButton - Klasa bazowa reprezentująca przycisk w menu głównym. Od niej (poprzez dziedziczenie) są tworzone przyciski w menu tj. Pełny Ekran, Nowa gra itd. Jej budowa pozwala na aktualizację przycisku - zmianę tekstu, aktywności - w trakcie działania programu w zależności od stanu całej aplikacji.

4. **Item**
   Klasa bazowa reprezentująca przedmioty występujące w grze. Implementuje operacje wspólne dla wszystkich przedmiotów - podnoszenie, odłożenie, zniszczenie.

   a. **Weapon(+ Axe, Bow, Sword)**
      Klasa reprezentująca bronie występujące w grze. Wymaga ustawienia prędkości ataku, ilości obrażeń. Oraz jej dzieci - konkretne bronie: Topór, Łuk, Miecz.

   b. **Potion (+LifePotion, ManaPotion)**
      Klasa reprezentująca mikstury pozwalająca na użycie ich na bohaterze. Oraz jej dzieci Mikstura życia i many - implementujące konkretne sposoby użycia ich.

   c. **Arrows**
      Klasa reprezentująca strzały wymagane przez łuk.

   d. **Armor**
      Klasa reprezentująca pancerz. Wymaga ustawienia liczby punktów obrony.

5. **Field**

Reprezentuje pojedyńczą komórkę/pole na mapie. Obsługuje zdarzenia wejścia na nią, położenia przedmiotu. A także przechowuje modyfikator który określa dodatkowe własności pola.(Spawn potwora, pole nie do przejścia)

a. **Wall (+WallNorth, WallSouth itp.)**
   Reprezentują ściane. Pole na które nie można wchodzić.

b. **Stairs (+StairsUp, StairsUpNorth itp.)**
   Reprezentują schody pole które przenosi stworzenia pomiędzy poziomami lochu.

c. **Trap (+TrapMove, TrapTouch itp.)**
   Reprezentują pułapki(reagujące na ruch/nacisk) lodu/strzał/ognia.

d. **Chest (+ChestNorth itp.)**
   Reprezentują skrzynki pozwalają na otwarcie i zebranie z nich przedmiotów, nie wolno na nie wchodzić.

6. **Field.Sprite, Wall.Sprite (+WallNorth.Sprite itp.), Stairs.Sprite (+StairsUp.Sprite itp.), Trap.Sprite (+TrapMove.Sprite itp.), Chest.Sprite (+ChestNorth.Sprite itp.), Creature.Sprite, Hero.Sprite, Monster.Sprite, Troll.Sprite, Mage.Sprite, Spider.Sprite, Skeleton.Sprite**
   Wszystkie te klasy opisują graficzne reprezentacje odpowiednich im klas. Pozwalają na podpięcie klas odpowiadających za logikę pół do pygame i ich wyświetlanie.

7. **Creature**
   Klasa bazowa opisująca podstawowe własności, metody itp. stwora żyjącego w grze. Opisuje jego poruszanie się, poziom życia, many, punktów doświadczenia i innych właściwości.

a. **Hero**
   Klasa reprezentująca bohatera obsługująca jego akcje - używanie magi, uderzenie, zdobywanie doświadczenia.

b. **Monster**
   Klasa reprezentująca potwory z AI. Potwory mają wspólne dosyć proste AI opierające się na kilku zasadach:
   Jeśli w pobliżu nie ma bohatera, rusza się losowo w odległości od pewnego punktu odniesienia
   Jeśli bohater jest w pobliżu:
   Jeśli mogę atakować atakuje
   Jeśli nie to zbliżam się do niego

i. **Troll, Mage, Spider, Skeleton**
   Klasy reprezentujące potwory. Ustawiają konkretne współczynniki dla AI(atak wręcz, dystansowy) oraz niekiedy troche je rozszerzają - mag potrafi się uzdrawiać.

8. **Map**
   Klasa pozwalająca na wyświetlanie części poziomu/mapy na dostępnej powierzchni. Udostępnia mechanizm warstw, który pozwala na rysowanie niezależnie wielu obiektów(oddzielenie mapy - pól, od potworów, cienia) dzięki czemu można przerysowywać mniej obiektów na ekranie.

9. **MapLayer**
   Klasa określająca podstawowe własności warstwy mapy, ukrywanie jej, obsługę zdarzeń myszy.

a. **FieldsLayer**
   Warstwa pól. Odpowiada za wyświetlanie aktualnej mapy na danym obszarze.

b. **HoverLayer**
   Warstwa wskaźników. Pozwala na wyświetlanie “wskaźników” nad polami mapy. (np. edytor wykorzystuje ją do pokazania gdzie postawimy kafelek.)

i. **CreatureLayer**

Warstwa wyświetlająca potwory w zależności od odkrytego przez ShadowLayer terenu. Oraz obsługująca wygląd kursora w zależności od tego czy znajduje się nad potworem.

ii. **ShadowLayer**
Warstwa mgły wojny. Pozwala na ukrycie terenu na którym bohater jeszcze się nie znajdywał oraz blokuje przekazywanie zdarzeń myszy do niższych warstw jeśli to zdarzenie nie wystąpiło na odkrytym terenie.

iii. **MissilesLayer**
Warstwa pocisków i ich graficznych efektów - strzał, kuli ognia oraz lodowego podmuchu i uzdrawiania. Pozwala na wyświetlanie ich w obrębie widocznego terenu.

c. **MinimapLayer**
Warstwa służąca do wyświetlania minimapy. Nie wyświetla się ona tak jak inne nad główną mapą. t Tylko zbiera aktualizacje i wyświetla je w polu minimapy w prawym górnym rogu.

10. **TilesGrid**
Toolbox pozwalający wybierać aktualnie używane narzędzie w edytorze. Obsługuje jego wybór i użycie.

11. **Tool**
Opisuje podstawowe własności narzędzia oraz podstawowy wygląd tego obiektu.

a. **ToolSetFieldType**
Narzędzie służące do zmiany typu pola na którym zostanie użyte.

b. **ToolToggleModifier**
Narzędzie służące do zmiany modyfikatora pola na którym zostanie użyte

12. **Inventory**
Reprezentuje i pozwala na obsługę przez użytkownika ekwipunku bohatera. Przekazuje odpowiednie zdarzenia myszy komórkom oraz odpowiada za wyświetlanie opisów przedmiotów z ekwipunku.

13. **ChestItems**
Reprezentuje i pozwala na obsługę przez użytkownika aktualnie otwartej skrzynki. Przekazuje odpowiedznie zdarzenia myszy komórkom oraz odpowiada za wyświetlanie opisów przedmiotów ze skrzynki.

14. **EditorMenu**
Reprezentuje menu występujące w edytorze. Pozwala na zapis aktualnego lochu oraz na wyczyszczenie całego planu.

15. **Cell**
Podstawowa komórka pozwalająca pokazać przedmiot z ekwipunku/skrzyni użytkownikowi oraz odpowiednio go obsłużyć.

a. **LeftHandCell, ArmorCell, RightHandCell**
Specjalne komórki przyjmujące broń i pancerz aktualizujące statystki bohatera.

b. **StrengthCell, AgilityCell, IntelligenceCell**
Specjalne komórki wyświetlające aktualne statystyki siły, zręczności, inteligencji bohatera oraz jeżeli bohater awansował pozwalające na ich aktualizację.

16. **Cursor**
Reprezentacja kursora w grze. Przechwytuje jego ruch a także w razie potrzeby wymusza przesunięcię mapy.

17. **Minimap**
Reprezentacja panelu znajdującego się w edytorze i scenie który przedstawia minimapę aktualnego

poziomu lochu.

18. **StatusBar**
    Odpowiada za wyświetlanie pasków życia i many oraz aktualnie wybranego czary podczas gry.

19. **ModuleQuit, EngineQuit, SceneQuit, EditorQuit**
    Pomocnicze klasy służące do wychodzenia z danego modułu(rzucane jako wyjątki i wyłapywane dzięki czemu wiadomo co wyłączyć)

Zidentyfikowane wzorce projektowe:

Łańcuch zobowiązań - mechanika warstw mapy pozwalająca na sekwencyjne przekazywanie zdarzenia aż jakaś stwierdzi że zostało już odpowiednio przetworzone albo skończą się warstwy.

Stan - Edytor - TilesGrid przechowuje stan - aktualne narzędzie, Scene - stanem są potwory, ustawienia bohatera w zależności od nich podejmowane są akcje w grze.

Template - Module, MapLayer - wzorce jak budować klasy pochodne - wymagane metody itp.