PanDa – 5 – opis wzorców i dokumenatcja

Widok oraz warstwa prezentacji:

Widok gry został zaimplementowany w oparciu o **wzorzec Model-View-Controller** z zastosowaniemklas charakterystycznych dla biblioteki libgdx, którą wykorzytsaliśmy do stworzenia gry.

Obsługa różnych typów przeciwników:

Implemenacja gry pozwala na swobodne dodawanie nowych rodzajów przeciwników. Każdy z przeciwników może posiadać dowolny efekt, którym może atakować gracza w trakcie jego ucieczki. Do obsługi tej funkcjonalności skorzystano ze **wzorca strategia.** W projekcie obecna jest klasa Hunter, która korzysta z HunterPower. Poprzez implementację interfejsu HunterPower możliwe jest rozszerzanie i dodawanie dostępnych przeciwników i efektów jakimi atakują gracza. Tworząc nowego łowcę przekazujemy mu wybraną strategię działania. Pozwala to dodatkowo ograniczyć ilość obiektów klasy Hunter ponieważ wystarczy podmienić startegię walki. Korzystanie wpływa to na zarządzanie pamięcią.

Tworzenie obiektów i zarządzanie pamięcią :

Do tworzenia obiektów gry, które są wykorzystywane zastowsowano **wzorzec fabryki abstrakcyjnej.** Zastosowanie fabryki pozawala na elastyczne tworzenie obiektów oraz przygotowanie przejrzystej konfiguracji obiektów, które są opisane przez stałe z klasy Constatns. Dzieki oddzieleniu tworzenia obiektów od ich funkcjonalności możemy łatwo nimi zarządzać.

W celu poprawienia wydajności gry i efektywności zarządzania pamiecią korzystamy z **wzorca pyłek** (ang. *Flyweight*). Dla obiektów, których liczba w czasie gry byłaby znaczna takich jak platformy czy też pociski Hunter’a tworzymy przestrzeń Pool. Na początku wypełniamy przestzreć pewną liczbą obiektów, a następnie w momencie, gdy dany obiekt jest potrzebny bierzemy go z dostępnej puli obiektów, a po zakończeniu korzystania z niego zwracamy, go z powrotem do ponownego wykorzystania.

Wzmocnienie gracza – PowrUps:

Podczas rozgrywki gracz może zdobywać różne wzmocnienia. Może w jednym czasie posiadać więcej niż jedno wzmocnienie (np. tarczę oraz magnes na monety). Obsługę tego mechanizmu oraz zapewnienie rozszerzalności zapewniono dzięki zastosowaniu **wzorca dekorator**. Chcąc dodać nowy efekt należy zaimplementowac interfejs PowerUpEffect, a następnie należy udekorować BasicPowerUpEffect. Dzięki temu uzyskujemy możliwość jednoczesnego posiadania wielu efektów, które mają określone funkcjonalności.