## Szablon

|  |  |
| --- | --- |
| **Opis ogólny** | |
|  | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
|  |  |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
|  |  |

## Ship

|  |  |
| --- | --- |
| **Opis ogólny** | |
| Klasa, która przechowuje w sobie wszystkie cechy postaci / statku grywalnej (-go). Określa położenie postaci, jej możliwości oraz aktualny stan.  Obiekt statku zostaje utworzony przy rozpoczęciu gry oraz usuwany po jej zakończeniu. | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
| Pozycja X | Pozycja postaci względem osi poziomej |
| Pozycja Y | Pozycja postaci względem osi pionowej |
| Szybkość | Tylko jak zaimplementujemy Focus. Względnie to może być bool ifFocus.  Ale jak będzie 2+ postaci, to każda może mieć swoją szybkość. |
| Max. liczba żyć | Tak jak w Bangu! – każda postać może mieć swój limit, zamiast jakiegoś globalnego. |
| Liczba żyć | W danej chwili |
| ~~Początkowa liczba żyć~~ | Z jaką się startuje. |
| Powyższe ~~3~~ 2, tylko dla bomb | -||- |
| Moc | Obecna moc postaci. |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
| Strzelaj | Wystrzelenie pojedynczej salwy pocisków |
| Ruszaj się | Kierunek może być przekazywany jako parametr |
| Użyj bomby | Wykorzystanie pojedynczej bomby. Nie można użyć kolejnej dopóki działanie pierwszej się nie skończy. W trakcie działania bomby można się ruszać i strzelać, ale nie oberwać. |
| Użyj Focusa | Obiekt gracza porusza się wolniej, by łatwiej było unikać pocisków. |

## Pocisk

|  |  |
| --- | --- |
| **Opis ogólny** | |
| Pojedynczy z pocisków jakie może wystrzelić postać grywalna bądź wróg bądź powietrze.  Obiekt zostaje utworzony w momencie wystrzelenia i żyje do momentu:   * trafienia w cel * wylecenia z planszy * uruchomienia bomby * zakończenia działania spellcardu * przegranej bossa, jeśli będą * przegranej gracza | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
| Kształt | Należałoby go jakoś powiązać ze swoim sprajtem lub obiektem 3D. |
| Punkt startowy X | Miejsce rozpoczęcia lotu na osi poziomej |
| Punkt startowy Y | Miejsce rozpoczęcia lotu na osi pionowej |
| Trasa | Pocisk porusza się wg swojego algorytmu. To gdzie doleci zależy tylko od tego i punktu startowego.  Można też utworzyć taki rodzaj pocisków, który goni gracza. Wtedy należałoby określić jak długo go goni oraz w jaki sposób. |
| Szybkość | Z jaką się porusza.  I tu otwiera się miejsce na zastosowanie jakichś podstawowych wzorów Newtona na ruch do sprawka. |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
| Ruszaj się | Kierunek i odległość byłyby wyliczane na bieżąco |

## Pattern

|  |  |
| --- | --- |
| **Opis ogólny** | |
| Wzór, który jest zbiorem pocisków. Wzór nie może istnieć bez pocisków. Wzór zajmuje się układaniem pocisków oraz wystrzeliwaniem każdego w odpowiednim momencie i z określonymi parametrami.  Należałoby określić w jaki sposób wzór generuje swoje pociski:   1. jedna możliwość: wszystko jest tworzone w momencie utworzenia obiektu. Np. jak wzór składa się z 20 pocisków to wszystkie tworzy w tle i dopiero po nagromadzeniu wszystkiego rozpoczyna się atak.   Efekt: potrzeba większej ilości pamięci, możliwe lagi, szybsza gra po załadowaniu.   1. druga możliwość: wszystko jest tworzone na bieżąco. Np. jak wzór składa się z 20 pocisków i strzela 4 salwami po 5 to każda piątka jest generowana w swoim momencie i od razu wypuszczana na planszę.   Efekt: zmniejszenie potrzebnej pamięci, konieczność rozsądnego zarządzania pamięcią.  W skrócie: statyczne vs dynamiczne. Ja, Forczu, skłaniam się ku drugiej opcji, bo obejmuje więcej problemów programistycznych. | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
| Algorytm | Należy określić rozsądnie w jaki sposób każdy wzór generuje swoje pociski i co z nimi robi. |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
| Startuj |  |

## Spellcard

|  |  |
| --- | --- |
| **Opis ogólny** | |
| Karta czarów jest zbiorem Wzorów, choćby jednoelementowym. | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
| Algorytm | Podobnie jak wyżej, zarządzanie Wzorami. |
| Czas trwania |  |
| Bonus | Bonus punktowy jaki się zdobywa za niewykorzystanie bomby / brak utraty życia w trakcie działania spellcardu |
| Czy Bonus? | Nie – to ni chuja nie ma bonusu + zaprzestanie naliczania |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
| Startuj |  |
| Zakończ |  |

## Enemy

|  |  |
| --- | --- |
| **Opis ogólny** | |
| Przeciwnik, którego należy wyeliminować. Jego śmierć może być warunkiem koniecznym dla ukończenia poziomu / pewnych jego części. | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
| Liczba życia | HP w danej chwili. Jeżeli nie przewidujemy regeneracji życia, nie ma potrzeby pamiętać maksymalnego HP. |
| Schemat ataku | Jak strzela pociskami / wzorami / spellcardami. |
| Kształt | Analogicznie do pocisku. |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
| Atakuj |  |
| Ruszaj się |  |
| Oberwij xD | Gracz umiera jak zostanie trafiony, tu może być prosta funkcja odejmująca punkty życia, sprawdzająca je z 0 i usuwająca obiekt, jak zajdzie potrzeba (+ ewentualny resp bonusu, może być random, albo i nie). |

## Game Session

|  |  |
| --- | --- |
| **Opis ogólny** | |
| Uruchomienie gry i obsługa całej infrastruktury gdy gracz wszystko wybierze i naciska „start”. | |
| **Cechy** | |
| *Nazwa* | *Komentarz* |
| Liczba punktów | Jakie w danej chwili zostały zdobyte. Początkowo jest równe zero. |
| Życia i bomby | Jeżeli każda postać będzie miała tę samą maksymalną liczbę żyć i bomb, info to możemy przechowywać tutaj. Jak nie to olać. |
| Graze | Charakterystyczny tylko dla każdego obiektu sesji gry. |
| **Możliwości** | |
| *Nazwa* | *Komentarz* |
| Rozpocznij |  |
| Zakończ |  |
| Podlicz punkty |  |

# Proponowane klasy

## Stage

Przechowuje informacje o wszelkim rozkładzie wrogów i kształtu planszy – w tym również tło, muzyka itd.

## Game

Klasa gry, która składa się ze Stage’y. Taka furtka, gdyby nasza gra składała się „z kilku Touhou”.

Wówczas zarys struktury wyglądałby tak:

Sesja\_Gry odpala Grę, która w miarę przechodzenia odpala kolejne Stejdże. W momencie przejścia Stejdżu, wychodzimy z niego do Gry, informujemy ją o tym, a ona przekierowuje nas dalej. Idea jest taka, że to Gra wie, gdzie mamy iść dalej, zamiast tworzyć Łańcuch\_Stejdży.

Sesja\_Gry przechowuje wszystkie informacje charakterystyczne dla danej rozgrywki, Gra i Stejdże – niezmienne cechy.

## Hitbox

Trzeba w jakiś sposób wprowadzić sprawdzanie, czy nastąpiła kolizja gracza z pociskiem/wrogiem, mógłby też odpowiadać za znikanie wrogów/pocisków za ekran (postać gracza „zatrzymywałoby” na krańcu). Z tego co widzę, nie ma atrybutu określającego aktualne położenie Enemy/Pocisk, więc mogliby (gracz też) posiadać atrybut typu Hitbox. Sam Hitbox mógłby mieć atrybuty X, Y i promień (określający rozmiar okręgu, dla jakiego sprawdzamy kolizję).

Należałoby też stworzyć jakąś kolekcję Hitboxów, które byłyby przetrzymywane w np. w Game Session. Do tego sprawdzane byłyby 3 typy kolizji (gracz -> wróg, gracz -> pocisk wroga, pocisk gracza -> wróg), więc można by te hitboxy jakoś rozróżniać (albo klasy pochodne, albo kolejny atrybut będący intem czy enumem) i sprawdzać tylko te kombinacje, które nas interesują. A są jeszcze bonusy.

Sprawa graze’u. Tu dochodzi jeszcze więcej obliczeń, bo dla hitboxa gracza potrzebny jest dodatkowy atrybut („grazeRadius”), określający czy dla danego pocisku zaliczamy graze. Z drugiej strony, potrzebny jest atrybut w stylu „isGrazed”, żeby nie doliczało graze’u przy każdej pętli programu dla tego samego przycisku.

Brzmi jak coś wymagającego sprzętowo przy tych porównaniach. Musiałbym przetestować, ja to wygląda.

## Bonus

Wszelkie doładowania do mocy, dodatkowe bomby + życia. Gdyby istniała klasa hitbox, to w zasadzie nie wymaga nic oprócz atrybutów „kształt” i „rodzaj bonusu”, bo spadanie odbywałoby się automatycznie (jednostajnie prostoliniowo ☺) w dół ekranu aż do natrafienia na gracza lub krawędź.