**x.x Przeciwnicy koncepcja i algorytm**

Przeciwnicy są obiektami w przestrzeni gry które są podzielone na typy. Każdy typ posiada unikalne cechy takie jak:

* wygląd,
* maxHP - liczba punktów życia,
* damage - obrażenia,
* minTimeBetweenAttacks- minimalny czas pomiędzy atakami,
* speedModifier - modyfikator szybkości poruszania,
* lostCounterMax – ilość sekund w których przeciwnik kontynuuje poruszanie się po zgubieniu gracza,
* playerFollowersCount – ilość „cieni” gracza za którymi podąża przeciwnik (dokładniej wyjaśnione w dalszej części)

Na początku każdego poziomu przeciwnicy są losowo rozmieszczani w labiryncie. Ich ustawienie musi spełniać pewne kryteria:

* nie mogą znajdować się na obszarze tego samego wierzchołka grafu co gracz
* na obszarze jednego wierzchołka nie może znajdować się więcej niż jeden przeciwnik / grupa przeciwników

Każdy poziom posiada taki sam algorytm rozmieszczania przeciwników, jednak różnią się dane wejściowe które są unikalne dla każdej mapy. Na dane wejściowe składa się lista par gdzie jednym elementem jest obiekt przeciwnika, natomiast drugim wartość liczbowa na podstawie której określa się prawdopodobieństwo wylosowania. Kolejnym elementem danych wejściowych jest całkowita liczba przeciwników którzy pojawią się na danym poziomie.

Implementacja algorytm wyboru kolejnego typu przeciwnika oraz wylosowania jego pozycji w pseudo-kodzie:

sumaPunktów = suma drugich elementów listy parametrów wejściowych

listaWierzchołków = wierzchołki gdzie liczba sąsiadów jest większa od 1

ilośćPrzeciwników **razy**:

losowaWartość = losowa wartość od 0 do sumaPunktów

sumaTymczasowa = 0

**długość(**parametryWejściowe**) razy (iterator i):**

sumaTymczasowa = sumaTymczasowa + parametryWejściowe[i].wartość

**jeśli** losowaWartość < sumaTymczasowa:

wylosowanyWierzchołek = losowy wierzchołek z listaWierzchołków

listaWierzchołków.Usuń(wylosowanyWierzchołek)

stwórz przeciwnika typu parametryWejściowe[i].typPrzeciwnika

**przerwij pętle**

Przeciwnicy, jako jedna z przeszkód jakie musi pokonać gracz by ukończyć poziom, powinni stanowić pewne wyzwanie. Przeciwnik stoi w miejscu do czasu, aż gracz podejdzie do niego na określoną odległość, która zależy od typu przeciwnika. Następnie zaczyna podążać za graczem według algorytmu przedstawionego poniżej. Przeciwnik może przestań podążać za graczem w dwóch przypadkach.

* Punkty życia przeciwnika spadną do zera
* Gracz ucieknie na odpowiednią odległość

Gracz posiada 7 „cieni”, są to niewidzialne obiekty poruszające się dokładnie tak jak gracz, jednak z pewnym opóźnieniem.

//rysunek z widocznymi cieniami

Implementacja algorytmu podążania za graczem w pseudo-kodzie. Algorytm wykonuje się w każdej klatce podążania przeciwnika za graczem:

forceToApply - wartość to siła przykładana do przeciwnika by się poruszał

playerFollowers - gracz oraz cienie ustawione w kolejności od gracza do najbardziej opóźnionego cienia

followerToFollow – numer cienia za którym powinien podążać przeciwnik, domyślnie -1

**długość(**playerFollowers**) razy (iterator i):**

**jeśli**  playerFollowers[i] znajduje się na tym samym wierzchołku co przeciwnik lub ,,,,,,,,,,,,,,,,,,,,,,,wierzchołku dostępnym z danego

followerToFollow = i

**przerwij pętle**

**jeśli** followerToFollow nie jest równe -1**:**

diff = playerFollowers[followerToFollow].pozycja – przeciwnik.pozycja

diff = wersor z diff

obrót przeciwnika w kierunku gracza

forceToApply = diff \* speedModifier

lostCounter = 0

**w przeciwnym wypadku:**

lostCounter = lostCounter + 1

**jeśli** lostCounter jest równy lostCounterMax \* 30 **:**

przestań śledzić gracza