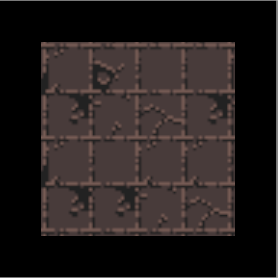
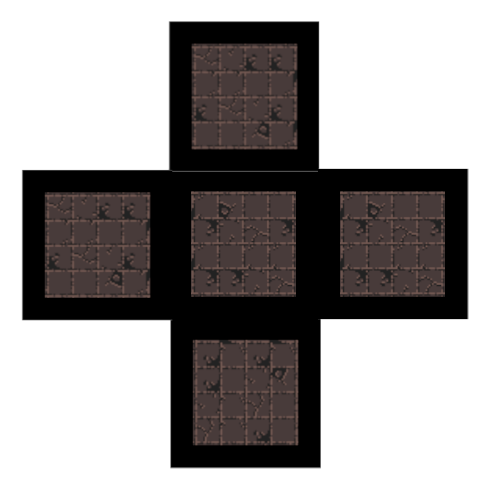
**x.x Generowanie labiryntu, koncepcja i algorytm**

Podstawą do stworzenia labiryntu w grze jest graf nieskierowany oraz nieposiadający wag, gdzie maksymalny stopień dla każdego wierzchołka to 4 a minimalny 1. Sąsiadem każdego wierzchołka jest wierzchołek znajdujący się bezpośrednio obok danego, na linii poziomej i pionowej w graficznej reprezentacji grafu.



Grafika x.1

Grafika x.1 przedstawia graficzną reprezentację wierzchołka grafu. Czarna obwódka reprezentuje ścianę, element przez który gracz nie może przejść. Natomiast na obszarze brązowym wewnątrz ramki gracz może się poruszać.



Grafika x.2

Na grafice x.2 przedstawiony jest przykładowy graf składający się z pięciu wierzchołków. Element środkowy jest sąsiadem każdego innego elementu zawartego w grafie.

Wygenerowany labirynt musi spełniać kilka ograniczeń:

- musi być możliwe dotarcie do każdego wierzchołka grafu

- z każdego wierzchołka musi istnieć dokładnie jedna ścieżka do każdego innego wierzchołka

Ograniczenie 1 zostało określone, by wygenerowany labirynt posiadał taką samą liczbę wierzchołków jak graf z którego został wygenerowany. Sprawia to, że wielkość wygenerowanego labiryntu jest przewidywalna. Ograniczenie 2 zostało określone by zapewnić brak pętli w labiryncie, co uniemożliwi graczowi uciekanie przed przeciwnikami w łatwy sposób.

Algorytmem który umożliwia wygenerowanie labiryntu zgodnego z ograniczeniami jest zmodyfikowany algorytm Prima. Modyfikacja będzie polegała na zmianie kryterium wyboru wierzchołka, który ma zostać w następnej kolejności dodany do już utworzonego minimalnego drzewa rozpinającego. W podstawowej wersji algorytmu, wierzchołek był wybierany z wierzchołków osiągalnych z minimalnego drzewa rozpinającego. Wierzchołki te są dodawane do kolejki priorytetowej, gdzie najwyższy priorytet posiada wierzchołek o najmniejszym koszcie dotarcia do minimalnego drzewa rozpinającego. Wykorzystana implementacja nie posiada kolejki, a wybór następuje poprzez wylosowanie jednego z dostępnych wierzchołków. Modyfikacja algorytmu znacząco zmniejsza złożoność obliczeniową algorytmu. Podstawowa wersja algorytmu posiada złożoność od O(|E| + |V|\*log|V|) do O(|E| \* log|V|) (gdzie |E| to liczba krawędzi a |V| to liczba wierzchołków) w zależności od implementacji kolejki priorytetowej na której operacja są najbardziej złożone obliczeniowo. Zrezygnowanie z kolejki zmniejsza złożoność do O(|V|).

Do zaimplementowania algorytmu wykorzystano klasę Wierzchołek która posiada następujące pola:

* czyOdwiedzony – oznacza czy wierzchołek został odwiedzony, czyli jest potencjalnym kandydatem na element krawędzi. Może przyjmować wartości typu prawda fałsz
* listaSąsiedztwa – lista sąsiadujących wierzchołków.

wierzchołkiBrzegowe – lista wierzchołków, do których możemy się dostać z wierxvhlkow odwiedzonych

Implementacja zmodyfikowanego algorytmu Prima w pseudo-kodzie:

krawędzieMDR = pusta lista par wierzchołków

wierzchołkiBrzegowe = pusta lista wierzchołków

obecnyWierzchołek = losowy wierzchołek grafu

obecnyWierzchołek.czyOdwiedzony = prawda

**dla każdego** elementu z listy obecnyWierzchołek.listaSąsiedztwa:

wierzchołkiBrzegowe.Dodaj(element)

**dopóki** lista wierzchołkiBrzegowe nie jest pusta:

obecnyWierzchołek = losowy wierzchołek z wierzchołkiBrzegowe

wierzchołkiBrzegowe.Usuń( obecnyWierzchołek )

obecnyWierzchołek.czyOdwiedzony = prawda

krawędzieMDR.Dodaj ((obecnyWierzchołek , losowy odwiedzony element z listy

obecnyWierzchołek.listaSąsiedztwa) )

**dla każdego** elementu z listy obecnyWierzchołek.listaSąsiedztwa:

**jeśli** element.czyOdwiedzony == fałsz **i** sąsiad nie jest w wierzchołkiBrzegowe:

wierzchołkiBrzegowe.Dodaj( sąsiad )

Labirynt jest reprezentacją grafu o krawędziach znajdujących się w liście krawędzieMDR.