Projekt dyplomowy inżynierski
Gra survival-RPG w Unity 3D
The Mighty Marian - genrowanie proceduralne

Krzysztof Jasiak, 137298
Tobiasz Biernacki, 137249
Dominika Sokołowska, 138635

04.11.2014

1 Generowanie proceduralne grach

1.1 Geneza

1.2 Najczęstrze algorytmy

1.3 Przykładowe zastosowania

2 Generowanie map w grze The Mighty Marian

W naszym projekcie każda rozgrywka polega na przejściu kilku, dziesięciu do pietnastu, poziomów jaskiń. Każda z jaskiń generowana jest niezależnie, przy użyciu algorytmu stworzonego na potrzeby projektu. Generowany obszar, pomimo tego, że w grze wizualnie reprezentowany jest w trzech wymiarach, na etapie generacji traktujemy jako dwuwymiarowy. Mapa składa się z płytek, zwanych także komórkami, które mogą przyjmować dwie wartości, są **podłogą** lub **nicością**. Bohater i wrogowie mogą przebywać i poruszać się jedynie po komórkach podłogi.
Komórki podłogi na rysunkach reprezentowane będą przez jaśniejsze pola, natomiast komórki, po których postacie nie mogą się poruszać, kolorem ciemnym.

2.1 Poprawność mapy

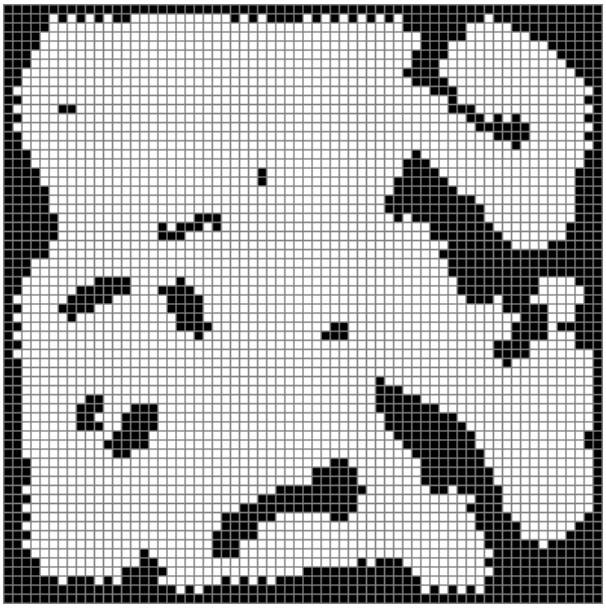
Celem algorytmu jest wygenerowanie mapy, która będąc wystarczająco skomplikowaną, aby gracz mógł się w niej zgubić i jednocześnie spójną, tak, aby wszystkie komórki podłogi, z których składa się mapa były osiągalne przez gracza. Drugi warunek jest konieczny do spełnienia, ponieważ gdyby postać Mariana i drabina umożliwiająca przejście między poziomami gry zostały umieszczone w innych śkładowych spójności"mapy, to skończenie gry byłoby niemożliwe.

To, czy mapa jest spójna (nie istnieją w niej komórki podłogi, do których nie da się dotrzeć z każdej innej komórki podłogi) da się stosunkowo prosto sprawdzić algorytmicznie, to określenie stopnia skomplikowania i zawiłości korytarzy jest już zadaniem, którego w ramach tego projektu nie podjeliśmy się zrealizować. Zdecydowaliśmy się określać w początkowym etapie skomplikowane korytarze i traktować je jako formę, na której działają kolejne kroki algorytmu. Daje nam to gwarancję mapy o porządanym poziomie skomplikowania i podzielonej na logiczne obszary. Ten proces zostanie dokładniej opisany w dalszej części dokumentu.

2.2 Automat komórkowy - gra w życie

W projekcie The Mighty Marian do generowania map został użyty automat komórkowy. Automat komórkowy o odpowiednich parametrach i kryteriach przeżycia komórek pozwala w niewielu krokach wygenerować ciekawe struktury podobne do jaskiń. Niestety to rozwiązanie ma również swoje ograniczenia.

Automat komórkowy to model matematyczny, w którym komórki znajdują się w jednym z określonych stanów. System składa się z pojedyńczych komórek, znajdujących się obok siebie. Każda z komórek może przyjąć jeden ze stanów, przy czym liczba stanów jest skończona. Plansza, na której znajdują się komórki może być w dowolnej skończonej liczbie wymiarów.
Inicjalnie, w czasie $t = 0$, każda z komórek znajduje się w jednym z możliwych stanów. Ich stan w czasie $t = 1$, nowa generacja komórek, określona jest pewną funkcją matematyczną, zwykle zależną od stanu jej sąsiadek.
Na potrzeby tej pracy rozpatrywać będziemy automaty komórkowe w dwóch wymiarach, w których komórki mogą przyjąć jeden z dwóch stanów, $ywa, martwa$. Oczywiście automaty komórkowe mają o wiele więcej możliwych zastosowań i możliwości, są używane chociażby przy symulowaniu ewolucji czy proceduralnym generowaniu tekstur.



Rysunek 1: Spójna mapa o niewielkim poziomie skomplikowania. Otwarty pokój, gdzie gracz nie będzie miał czego odkrywać

Rysunek 2: Mapa niespójna, ale ciekawa pod względem grywalności

2.3 Etapy procesu generowania mapy

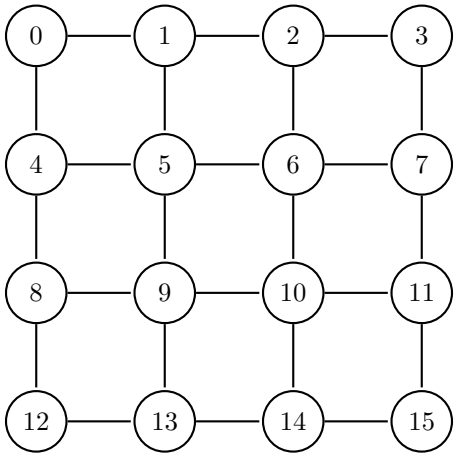
Proces powstawania map można podzielić na pięć etapów. Produkt końcowy każdego z etapów jest danymi wejściowymi dla kolejnego etapu.

2.3.1 Etap labiryntu

Proces generowania mapy rozpoczyna się od stworzenia labiryntu, który określi czy pomiędzy wybranymi pokojami występuje połączenie.
Parametry startowe algorytmu w pierwszej fazie są następujące:
$RoomsX$, $RoomsY$ - ilość pokojów, na które dzielimy przestrzeń mapy

Na tym etapie mapę modelujemy za pomocą grafu prostego ważonego w następujący sposób:
$G(V, E)$ - graf nieskierowany
V - zbiór wierzchołków - pojedyńczy wierzchołek reprezentuje jeden pokój $|V| = RoomsX * RoomsY$
E - zbiór krawędzi - krawędź reprezenuje przejście między pokojami

Program generuje początkowy graf przejść między pokojami. Inicjalnie wszystkie możliwe przejścia między sąsiednimi pokojami istnieją.

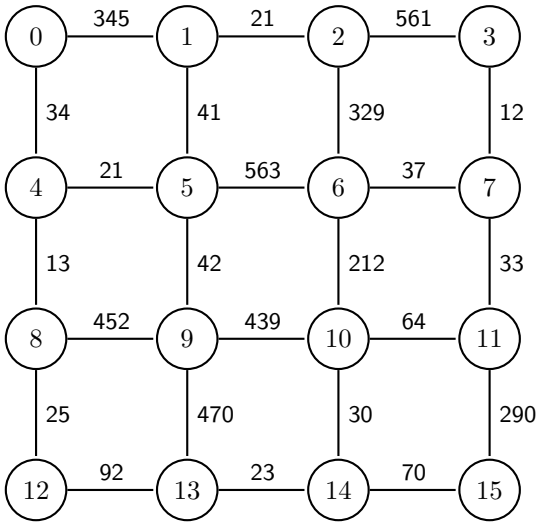


Rysunek 3: Początkowy graf połączeń między pokojami dla $RoomsX = RoomsY = 4$

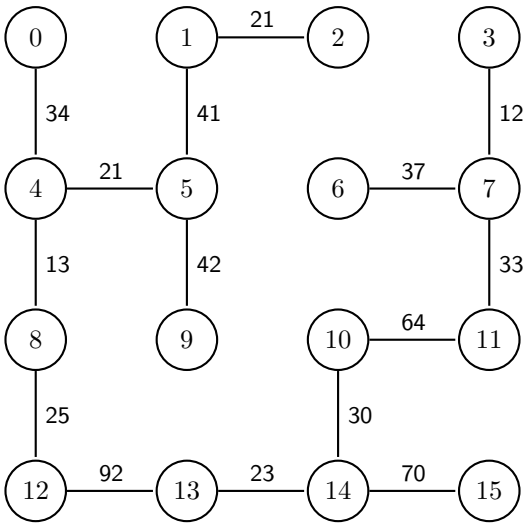
W tak zamodelowanej przestrzeni wygenerowanie labiryntu łączącego pokoje sprowadza się do znalezienia **minimalnego drzewa spinającego** w grafie G. Drzewo spinające grafu jest grafem spójnym i acyklicznym, który zawiera wszystkie wierzchołki grafu oraz niektóre z jego krawędzi. Minimalne drzewo spinające jest drzewem spinającym, którego suma wag krawędzi jest najmniejsza ze wszystkich pozostałych drzew rozpinających danego grafu.. W danym grafie może istnieć może istnieć więcej niż jedno drzewo o tych własnościach. Z punktu widzenia grywalności nie ma znaczenia które wybierzemy, zatem wystarczy wskazać jedno z nich, a wagi krawędziom grafu możemy przypisać losowo. Do uskania minimalnego drzewa spinającego został użyty został algrytm Prima, wybrany ze względu na łatwość implementacji.
Krawędziom grafu G przyporządokwane zostają losowe wagi z zakresu $(3, 660)$. Wierzchołek startowy dla algorytmu również jest wybierany losowo.

Algorytm Prima oparty jest o metodę zachłanną. Można opisać go następująco:

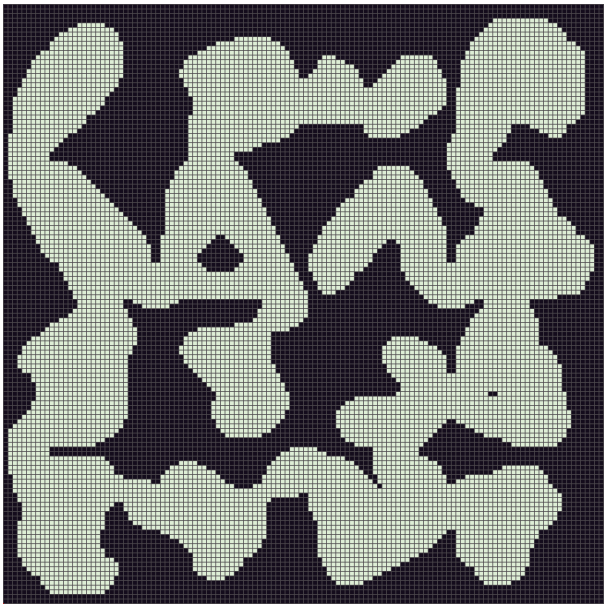
1. Rozpoczynamy od grafu składającego się jedynie z wierzchołka startowego.
2. Krawędzie incydentne do wierzchołka umieszczamy na posortowanej wg. wag liście.
3. Zdejmujemy z listy krawędź o najmniejszej wadze i sprawdzamy, czy łączy wierzchołek wybrany z niewybranym. Jeśli tak, to znalezioną krawędź dodajemy do drzewa spinającego.
4. Dodajemy krawędzie incedentne z nowowybranym wierzchołkiem do posortowanej listy.
5. Powtarzamy kroki 2 - 4 dopóki lista krawędzi nie będzie pusta.



Rysunek 4: Graf przejść z nadanymi wagami



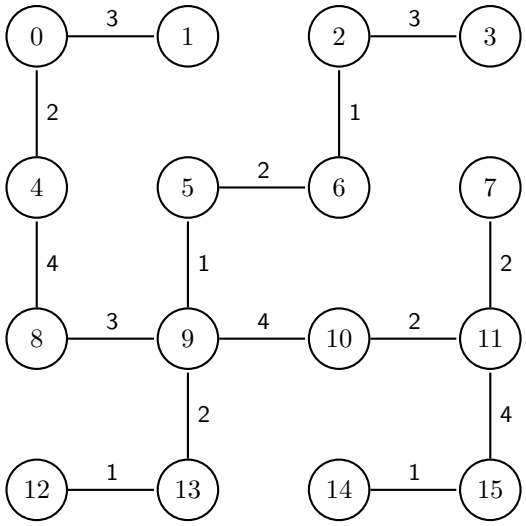
Rysunek 5: Minimalne drzewo spinające znalezione przez algorytm Prima



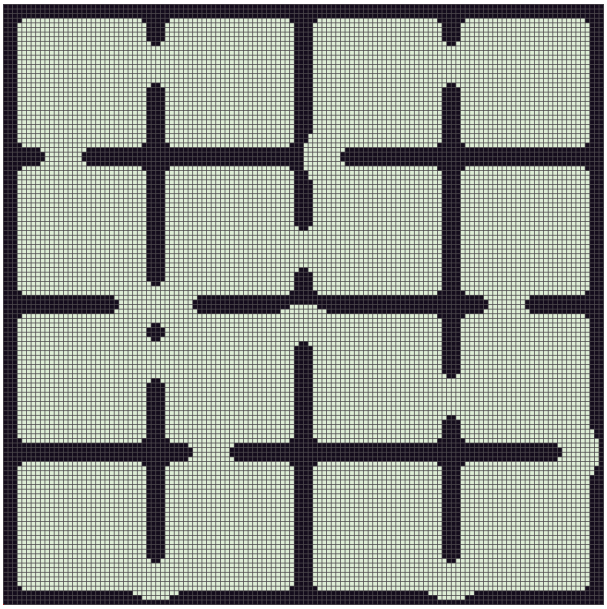
Rysunek 6: Mapa, która powstała dla tych warunków początkowych, wygenerowana przez program The Mighty Marian

2.3.2 Położenie przejść między pokojami

W określonym w poprzednim etapie drzewie spinającym wagi krawędzi zastępujemy losowo wartościami ze zbioru $1, 4$. Ta wartość określa w którym miejscu pomiędzy pokojami utworzone zostanie przejście. Przejście to prostokąt o szerokości $rozmiarpokoju/4$ i długości dwóch komórek. Przejścia są obliczane i umieszczane osobno w każdym pokoju. Dzięki wprowadzeniu różno



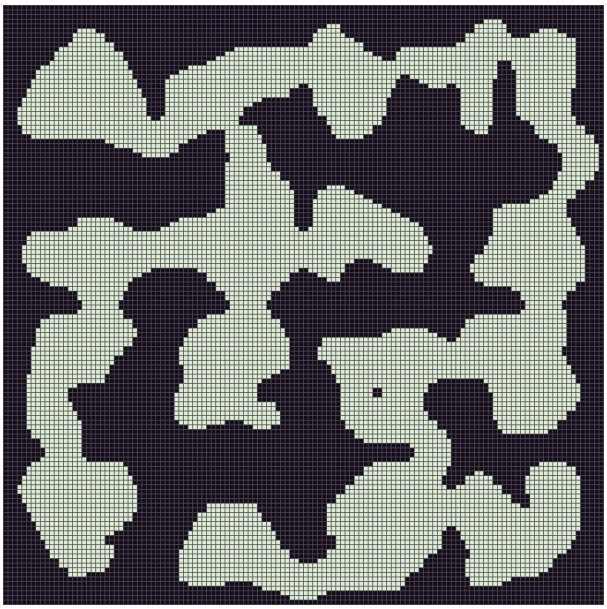
Rysunek 7: Szerokość na jakiej umieszczono przejście



Rysunek 8: Wpływ na wygenerowaną mapę

Już cztery stopnie różnorodności przejść tworzą wrażenie różnorodności i pomagają ukryć przed graczem, to że porusza się po prostu po labiryncie.

Rysunek 9: Przykładowa mapa, w której przejścia umieszczone są na tej samej szerokości



Rysunek 10: Przykładowa mapa w której przejścia pojawiają się w różnych szerkościach