howTo :: Generator Labiryntów (sheadovas/poradniki/howto/howto-generator-labiryntow/)

Wrz 05, 2015 / howTo (sheadovas/category/poradniki/howto/)

Witam Was w kolejnym poradniku z serii *howTo*, w którym zajmiemy się generowaniem losowych labiryntów, o wcześniej podanych wymiarach i co jeszcze warto zaznaczyć: będą to labirynty doskonałe (zaraz sobie wyjaśnimy o co dokładnie chodzi).

Seria *howTo* zazwyczaj składa się z dwóch części: teoretycznej, w której przedstawiam ogólną ideę omawianego zagadnienia oraz część praktyczną w formie wideo (film znajduje się pod tym wpisem, jednak do jego zrozumienia jest wymagana znajomość przedstawionego w wpisie materiału).

Labirynt doskonały

Najprościej mówiąc: jest to taki labirynt, w którym możliwe jest dojście do każdego pola, tzn nie ma komórek, które są niedostępne dla gracza przez zablokowaną ścianę.

Najlepiej to ilustruje poniższy obrazek, po lewej widzimy labirynt, w którym do części labiryntu nie prowadzi żadna ścieżka, po prawej: labirynt doskonały (do każdej komórki labiryntu prowadzi jakaś droga).

perfect-maze

(https://i1.wp.com/www.shead.ayz.pl/wp-content/uploads/2015/09/perfect-maze.png)

Algorytm

Nasz algorytm do generowania labiryntu jest oparty o algorytm DFS (Depth First Search) i jego złożoność pamięciowa to O(n), jednak gdy się nieco pokombinuje można zejść do O(n/4).

Całą procedurę rozpoczynamy od utworzenia siatki komórek (każda komórka jest otoczona przez 4 ściany).

empty-grid

(https://i2.wp.com/www.shead.ayz.pl/wp-content/uploads/2015/09/empty-grid.png)

Dalej pracujemy wg algorytmu:

1. Wybieramy losowo pozycję startową i ustawiamy ją jako komórkę bieżącą..
2. Badamy sąsiednie komórki i zapisujemy do listy tylko te, które posiadają 4 ściany (czyli są nieodwiedzone).
3. Jeśli lista nie jest pusta, to wybieramy jedną z nich w sposób losowy i usuwamy ścianę pomiędzy wylosowaną komórką, a tą którą obecnie się zajmujemy. Następnie przechodzimy do wylosowanej komórki (staje się ona bieżącą komórką).
4. Jeżeli wszystkie komórki sąsiadujące z bieżącą komórką zostały odwiedzone, to cofamy się o komórkę i ustawiamy ją jako bieżącą.
5. Należy powtarzać kroki 2-4, aż do momentu odwiedzenia wszystkich komórek.

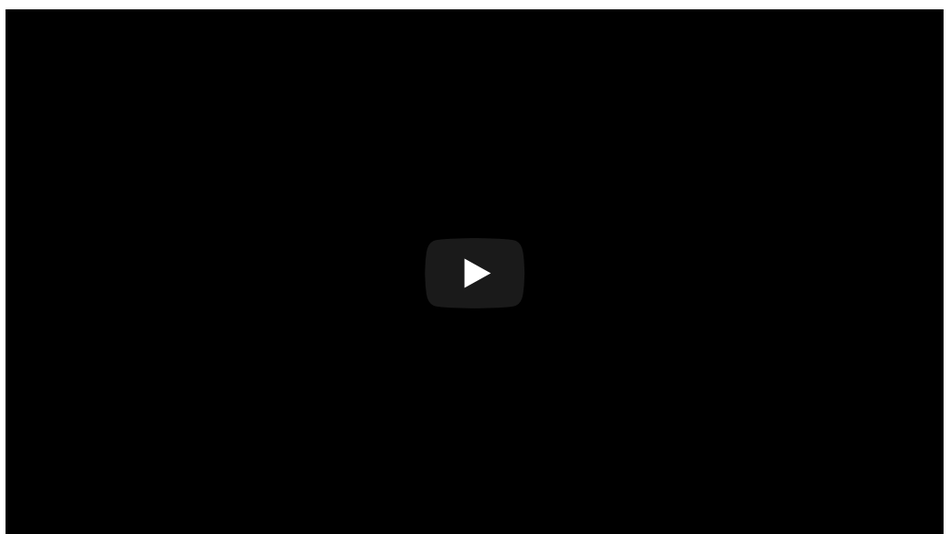
Struktury

Przykładowa struktura do zastosowania w implementacji algorytmu.

```C++
struct Cell
{
 bool visited = false; // flaga odwiedzenia komórki

 // sąsiadujące z komórką ściany
 Wall* left;
 Wall* right;
 Wall* bottom;
 Wall* top;
};
```

Implementacja algorytmu



(http://adf.ly/1QDZpL)

Code ON!