Testumgebung

Levelgenerator

Michael Ulko 5830648

Team 12

Als Testumgebung habe ich die in Eclipse integrierte Konsole benutzt und habe mir dort alles ausgeben lassen.

Meine Aufgaben im Überblick waren:

- Labyrintherzeugung mittels FloodFill-Algorithmus
- Türenplatzierung im Raum
- Monsterplatzierung im Raum
- Itemsplatzierung im Raum
- Mind. 2 verschiede Raumtypen

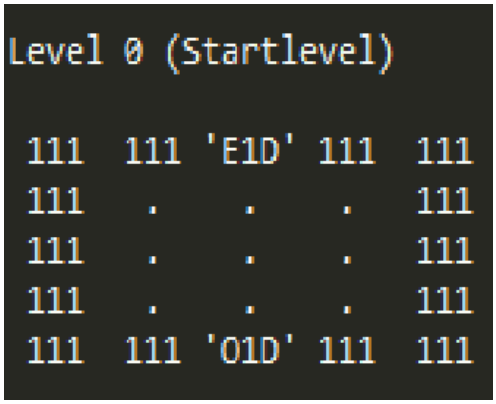
Unser Spiel hat 67 Knoten. Jedem Knoten ist eine ID zugewiesen (0-66) und jeder Knoten enthält einen Level. Jeder Level ist vom Typ Tile[][].

Also: Unser Spiel hat 67 Levels. Unter diesen finden sich folgende Raumtypen:

- Startlevel x1 (Knotennummer: 0)
- Boss Level x3 (22, 44 und 66)
- Shop Level x12 (4, 6, 17, 19, 26, 28, 39, 41, 48, 50, 61, 65) und
- Levels mit Labyrinthen x51 (restliche knoten)

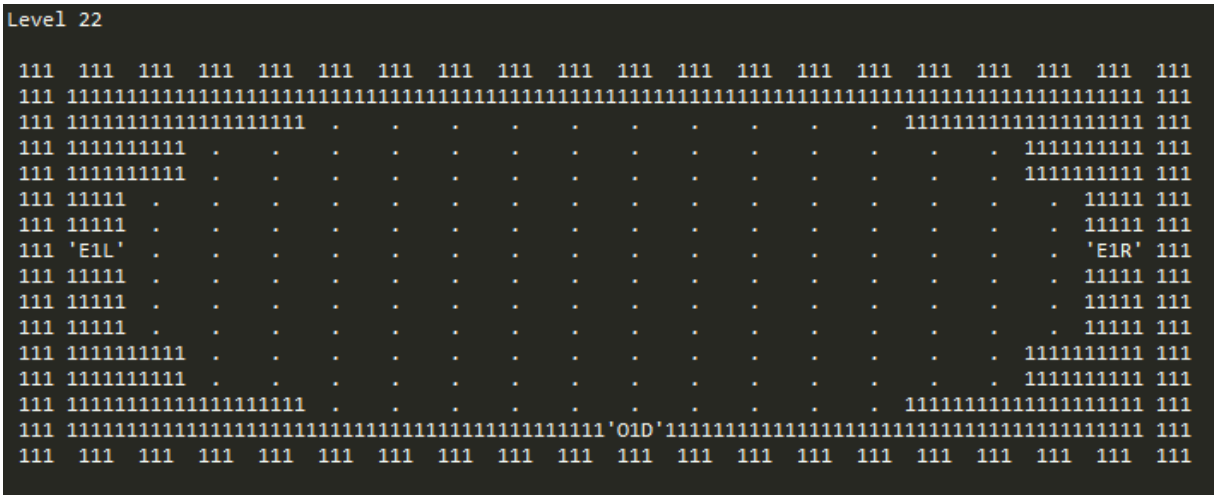
Ich gehe jetzt alle Raumtypen durch und erkläre stichpunktartig, wo, wie und was funktioniert.

Startlevel



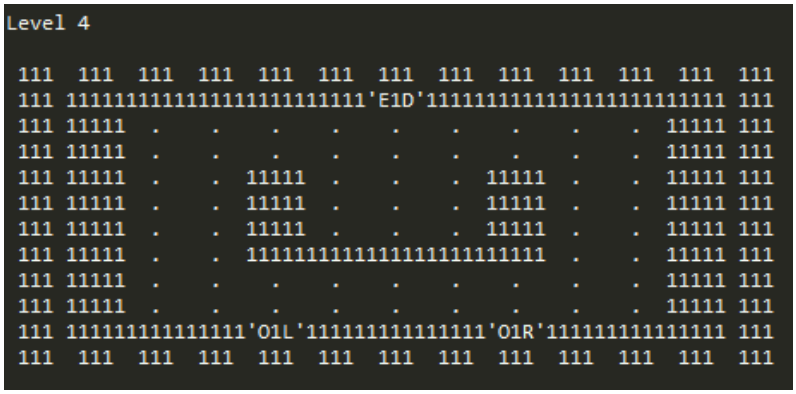
Die Struktur für diesen Raumtypen ist in einer Methode festgelegt. Es gibt nur zwei Türen: Eingang und Ausgang zum ersten Spiellevel. Die Türen werden auch immer an gleicher Position festgelegt.

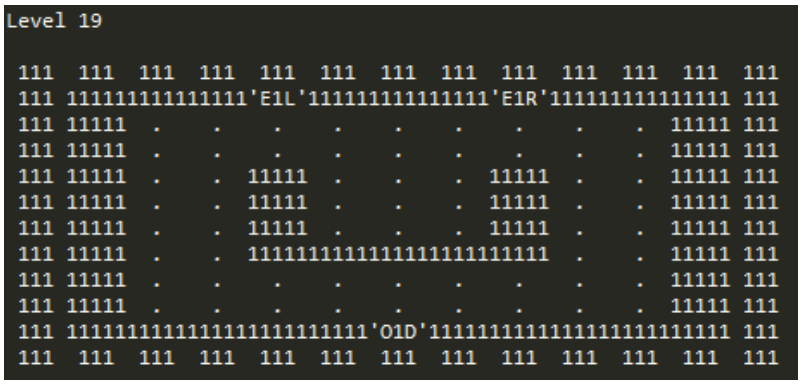
Bosslevel



Die Struktur für diesen Raumtypen ist auch in einer Methode festgelegt. Diesmal gibt es 3 Türen: Rechter Eingang, linker Eingang und einen Ausgang zum nächsten Akt. Die Türen werden auch immer an gleicher Position für alle 3 Bosslevels festgelegt.

Shoplevel

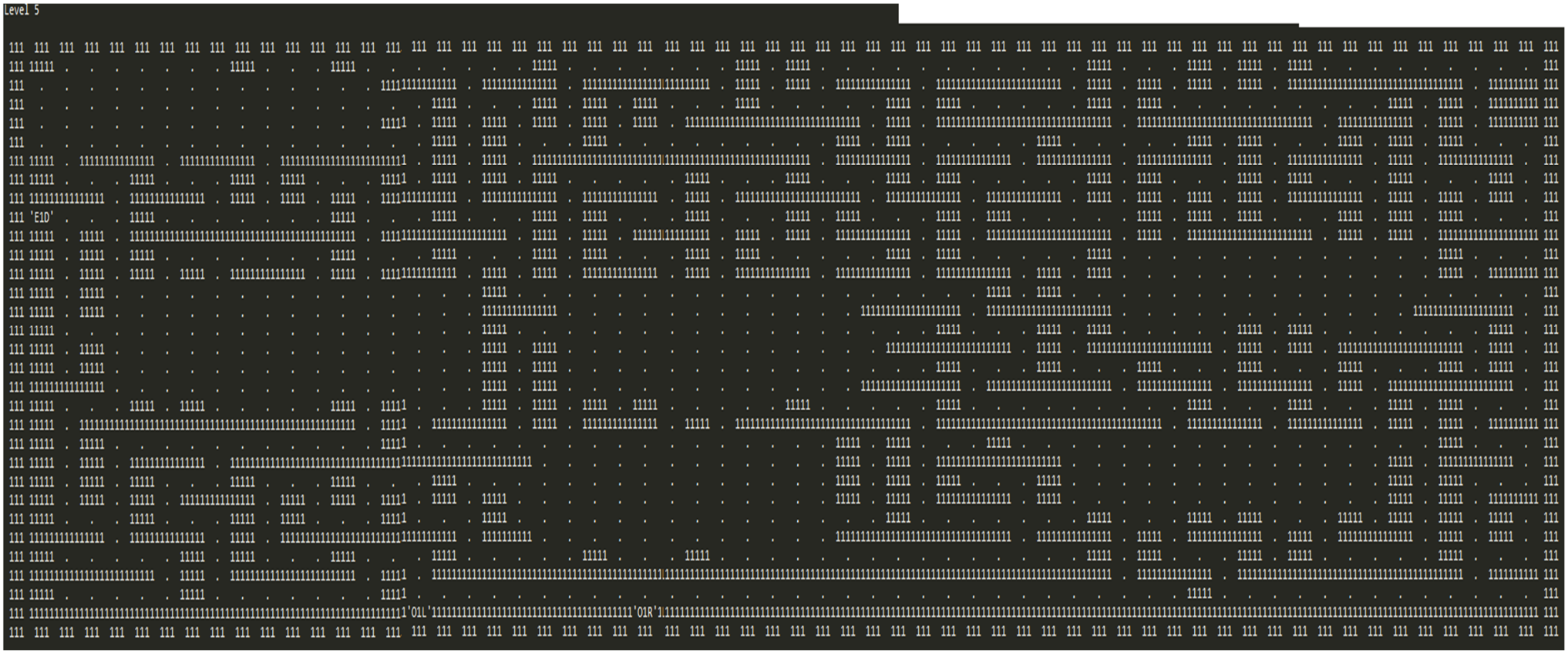




Es gibt 2 verschiede Arten von Shoplevels: Der eine hat 1 Eingang und 2 Ausgänge, der andere hat 2 Eingänge und 1 Ausgang. Das liegt daran, dass die erste Art des Levels in der ersten, die andere in der zweiten Baumhälfte liegt (dazu siehe Hinweis am Ende des Dokuments).

Die Struktur für diesen Raumtypen ist auch in einer Methode festgelegt.

Levels mit Labyrinthen



Bei solchen Räumen kommt der FloodFill-Algorithmus zum Einsatz, um die Wege durch das Steinfels zu schlagen. Die Räume werden per einen Algorithmus zufällig generiert. Die Türen werden entweder unten oder links per Zufall platziert.

Monster- und Itemsplatzierung

```
Level: 64
Art des Charakters: 1
Anzahl Monster im Level: 7
Monster 1 hat folgende Positionen: posX = 1     posY = 49
Monster 2 hat folgende Positionen: posX = 7     posY = 15
Monster 3 hat folgende Positionen: posX = 17    posY = 43
Monster 4 hat folgende Positionen: posX = 23    posY = 14
Monster 5 hat folgende Positionen: posX = 17    posY = 57
Monster 6 hat folgende Positionen: posX = 5     posY = 6
Monster 7 hat folgende Positionen: posX = 4     posY = 47

Level: 64
Anzahl Items im Level: 3
Item 1 hat folgende Positionen: posX = 18       posY = 49
Item 2 hat folgende Positionen: posX = 21       posY = 24
Item 3 hat folgende Positionen: posX = 17       posY = 54
```

Bei der Erzeugung jedes Levels, wenn das Tile[][]-Array fertig ist, starten die Methoden für die Platzierung von Monstern uns danach Items im Level. Dabei werden die Monster bzw. items-Objekte nicht im Array abgelegt, sondern lediglich ihre Positionen ermittelt. Für Speicherung der Anzahl und Positionen werden 2 Vector[]-Arrays der Länge 66 benutzt (= Ein Vector für einen Level außer Startlevel und ein Array für Monster sowie eins für Items).

Hinweis:

- Jeder Level hat eine globale Grenze (anderes Tile-Objekt)
- Um das o.g. zu überprüfen, starte die Main-Methode in der Main-Klasse aus
- Es kann auch sein, dass dabei nicht alles auf der Konsole dargestellt wird. Dafür soll man nur die maximale abbildbare Zeichenanzahl für die Konsole erhöhen.

Baumstruktur (Skizze)