# 27-09-2014

Vandaag en gister heb ik een echt begin gemaakt met de software die op de pc moet komen. Ik heb het project opgezet en de mappenstructuur aangemaakt.

Ik ben begonnen met wat basic classes maken op alle data in op te slaan die nodig is om het level te genereren. De absolute kern van dit deel van het programma is het *recursive backtracking* algoritme. Na wat rondzoeken op het internet kwam ik er op uit dat dat een algoritme is wat past bij onze eisen. Het is niet heel traag, makkelijk te begrijpen en er is een bestaande C# implementatie van. Toch heb ik geprobeerd het over te schrijven vanuit Ruby, omdat ik het zelf goed wil begrijpen en eventuele performance issues wil voorkomen. Ik zal nog een flowchart maken van het algoritme zodat het misschien wat inzichtelijker is.

Gebruikte bron over het algoritme: http://weblog.jamisbuck.org/2010/12/27/maze-generation-recursive-backtracking

## Uitleg bij de gemaakte bestanden

Ik zal hieronder wat uitleg geven over de bestanden die ik heb aangemaakt.

De solution bestaat op dit moment uit 2 projecten: RecursiveBacktracking en TowerHunterEngine. Het eerste project heb ik niet helemaal zelf gemaakt. Ik heb de implementatie van het algoritme van het internet, en er daarna slechts voor gezorgd dat het zichtbaar is op het scherm. Ik gebruik dit als (werkende) referentie voor ons eigen algoritme.

TowerHunterEngine is het belangrijkste project. Dit bestaat uit de volgende mappen:

* Arduino  
  Hierin komen alle classes die verantwoordelijk zijn voor communicatie met de Arduino
* Content  
  Waarschijnlijk zullen we deze map alleen gebruiken als onze game grafisch interessanter wordt. Hier komen namelijk alle media assets van het programma in. Tot nu toe streef ik ernaar om alles in het geheugen te genereren, omdat dat tot nu toe sneller is. Het is ook makkelijker om snel te bewerken als er iets anders moet.
* Controls  
  Hier komen alle classes in die verantwoordelijk zijn voor de besturing van de spelers dmv de controllers
* Playfield  
  Hier ben ik vandaag voornamelijk aan bezig geweest. Hier staan de classes die het doolhof en daarna de rest van het speelveld genereren. Ook staat hier in opgeslagen waar de torens staan en of ze actief zijn etc. De bedoeling is dat het speelveld eenmalig wordt gegenereerd door de Generator, en dat het daarna reageert op alle acties van de spelers. Alle data staat dus in het Playfield.Field object, wat ook de hele tijd aangepast zal worden. Voor de rest staan er nog wat kleinere classes in als onderdeel van het speelveld.
* Score  
  Hier komt alles wat met scoren en het weergeven van de score te maken heeft.
* Utils  
  Hier staan wat algemene functies van het programma. Een voorbeeld is RuntimeTextures.cs, de class die er voor zorgt dat alle textures in het geheugen worden gemaakt en niet van de hardeschijf worden geladen.

De rest van de uitleg geef ik wel op school, maar hier is de belangrijkste documentatie - Thomas