|  |
| --- |
| ODISEE |
| Advanced Applied Programming |
| 3D-Doolhof met werkende fysica |

|  |
| --- |
| Bram Vanhoutte |

Inhoud

[De opdracht 2](#_Toc515034860)

[Het onderzoek 2](#_Toc515034861)

[Het 3D-model 2](#_Toc515034862)

[De fysica 2](#_Toc515034863)

[Versnelling 2](#_Toc515034864)

[Maze algoritme 3](#_Toc515034865)

[Depth-first search 3](#_Toc515034866)

[Recursive devision method 3](#_Toc515034867)

[De praktische uitwerking 4](#_Toc515034868)

[Het 3D-model 4](#_Toc515034869)

[De fysica 4](#_Toc515034870)

[Beweging van de bal 4](#_Toc515034871)

[Collision 5](#_Toc515034872)

[Maze algoritme 5](#_Toc515034873)

[Code structuur 6](#_Toc515034874)

[Maze 6](#_Toc515034875)

[Globals 6](#_Toc515034876)

[Logic 6](#_Toc515034877)

[Besluit 6](#_Toc515034878)

[Referenties 8](#_Toc515034879)

# De opdracht

Het maken van een WPF-toepassing (Windows Presentation Foundation) die een 3D-Model weergeeft waarmee geïntrageerd kan worden. De WPF-toepassing toont een 3D-doolhof met daarin een bal. De gebruiker kan het doolhof doen draaien waardoor de bal door fysica gaat beginnen rollen. Zo moet de gebruiker het doolhof proberen uit te spelen.

# Het onderzoek

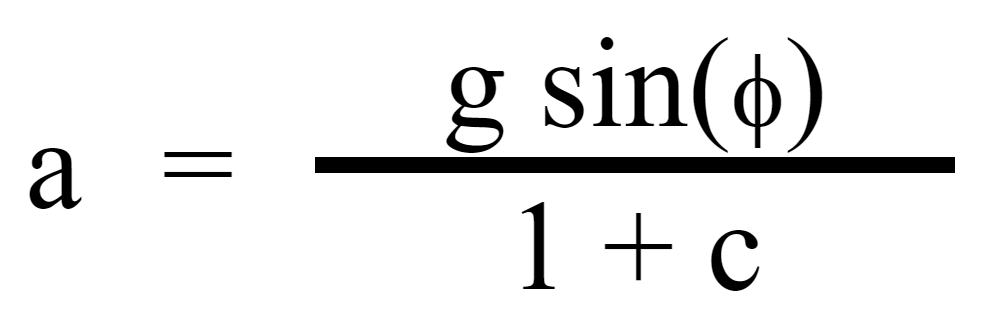
## Het 3D-model

Het begin van het onderzoek is het verstaan van WPF, in tegenstelling tot WinForms, die normaal in de opleiding gebruikt worden. WPF gebruikt dezelfde onderdelen van WinForms zoals: button, label, combobox, … maar elke component kan ook in XAML (Extensible Application Markup Language) aangepast worden. Hierbij worden alle componenten van de view getoond in een markup language. In die weergave kan elk component ook aangepast of toegevoegd worden. De component in XAML dat het tonen van 3D objecten toelaat is “Viewport3D”. Hierin worden alle componenten zoals licht, camera en 3D-objecten toegevoegd. Voor het verstaan van de werking van al die componenten worden tutorials gevolgd met code voorbeelden. [1] [2]

## De fysica

Voor het implementeren van de fysica is het begrijpen van de nodige formule(s) belangrijk. Deze zijn makkelijk te vinden en kunnen aan de hand van code-voorbeelden snel geïmplementeerd worden. Hierover is op de website “’physics.bu.edu” heel wat te vinden. [3]

### Versnelling

De formule die aan de basis ligt het programma is deze. [4]  


Dit is een formule voor de versnelling van een bal op een helling.  
Hier wordt de versnelling van de bal berekend aan de hand van de valversnelling van aarde (g = 9.81), vermenigvuldigd met de sinus van de hoek van de helling. Dit wordt over 1 plus de constante c (getal afgeleid van eigenschappen van object zoals massa, vorm) gezet.

## Maze algoritme

Er zijn voor het project twee algoritmen onderzocht, het “Depth-first search” algoritme en het “Recursive devision method” algoritme. Deze werken op twee heel verschillende manieren.   
Het “Depth-first search” algoritme bleek met bestaande structuur van het programma makkelijker te implementeren. Ook scheen het moeilijker om met het “Recursive devision method” algoritme het juiste pad voor de gebruiker aan de duiden. [5]

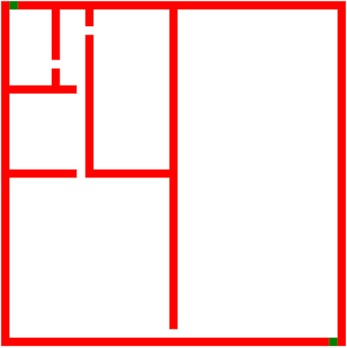
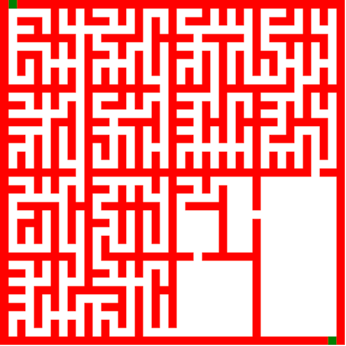
### Depth-first search

Hierbij word het bord verdeeld in een grid. Elke node in dat grid heeft vier muren en 2-4 aangrenzende nodes. Er is ook een stack waar al bezochte nodes opgelegd worden. Als er een node van een stack wordt afgehaald dan is het altijd de laatst toegevoegde.  
Het algoritme begin bij een eerste node. Vervolgens kiest het algoritme me een willekeurige aangrenzende node die nog niet bezocht is. De muur tussen de huidige node en de gekozen buur-node wordt weggehaald, de huidige node wordt op de stack gezet en de huidige node wordt als bezocht aangeduid. Dan wordt de gekozen node de huidige node en begint dit proces opnieuw.

Als er geen node geen onbezochte buren heeft dan begint het algoritme met backtracking. Hierbij worden bezochte nodes van de stack afgehaald. Deze worden dan een voor een voor onbezochte buren nagekeken. Als een node die van de stack afgehaald is een onbezochte buur heeft. Dan stopt het algoritme met backtracking en gaat het verder zoals ervoor.  
  
Als het algoritme bij backtracking heel de stack is afgegaan, en dus de stack leeg is, dan is het programma klaar en zijn er dus geen onbezochte nodes meer. Hieruit komt dan een gegenereerd doolhof.

### Recursive devision method

Bij deze methode start men met een leeg bord met enkel zijmuren. De ruimte wordt dan in twee gedeeld door een muur met één opening. De twee ruimte worden nog eens door twee geeld door muren met één opening. Zo blijft het algoritme de ruimtes door twee delen tot er geen ruimtes meer zijn maar enkel gangen. Dit zou dan een doorloop baar doolhof moeten uitkomen.

# De praktische uitwerking

## Het 3D-model

Het grootste probleem was het 3D model maken. Hiervoor was XAML leren gebruiken noodzakelijk, dit nam veel tijd in beslag. Na het verder begrijpen, kond met verschillende aspecten van 3D objecten zoals: materiaal, kleur, locatie, grootte, texture, triangelindices, ... geëxperimenteerd worden. In XAML wordt de onderkant van het doolhof als een rechthoek aangemaakt. Deze wordt als “DiffuseMaterial” aangemaakt, dit is een volle kleur. Als materiaal type voor een 3D object zijn er ook “SpecularMaterial” (deze wordt gebruikt om een glans effect op een 3D object te krijgen) en “EmissiveMaterial” (deze zorgt dat het 3D object ook een emissive material wordt). Elke 3D object bestaat uit TriangleIndices, elk vlak op een balk heeft 2 driehoeken die het vlak zichtbaar maken. Deze moeten worden gedefinieerd aan de hand van de hoeken van de driehoeken die overeenkomen met de hoeken van de balk.

De bal van het doolhof is ook in XAML gegenereerd, aan het XAML object wordt ook een naam gegeven zodat deze in code aangesproken kan worden. De bal bestaat uit “DiffuseMaterial” om het een kleur te geven en heeft ook een “SpecularMaterial” laag om het een glanzend effect te geven. De muren van het doolhof worden in code gegenereerd. Dit is niet enkel voor simpliciteit, maar maakt het mogelijk om ze dan aan passen voor de toekomstige opdracht waar we een willekeurig doolhof-patroon gaan moeten generen.

Het 3D-model kan worden gedraaid aan de hand van de pijl- en WASD-toetsen. Het programma bevat twee timers. De eerste timer is de logica timer die om de 10ms de input van de gebruiker registreert en als gevolg de hellingsgraad van het doolhof verandert. Deze timer berekend ook om de 10ms de fysica van de bal.   
De tweede timer is de grafische herlaad timer, deze herlaad de zichtbare draai van het doolhof en de locatie van de bal aan 60FPS(Frames Per Second).

De fysica

Beweging van de bal

De basis van de beweging van de bal is de snelheid. Dit wordt bijgehouden in twee variabelen; de snelheid op de x-as en de snelheid op de y-as. Deze snelheid wordt in cm/s bijgehouden. Om deze snelheid toe te passen op de bal, zal die altijd omgezet moeten worden van cm naar pixels in het coördinatenstelsel van mijn “Viewport3D”. Dit gebeurt door het met factor 37.795275590551178 te vermenigvuldigen. [6] Elke keer als de logica-timer de berekeningen aanroept, wordt de snelheid van de bal op de x-as zowel als op de y-as herrekend.

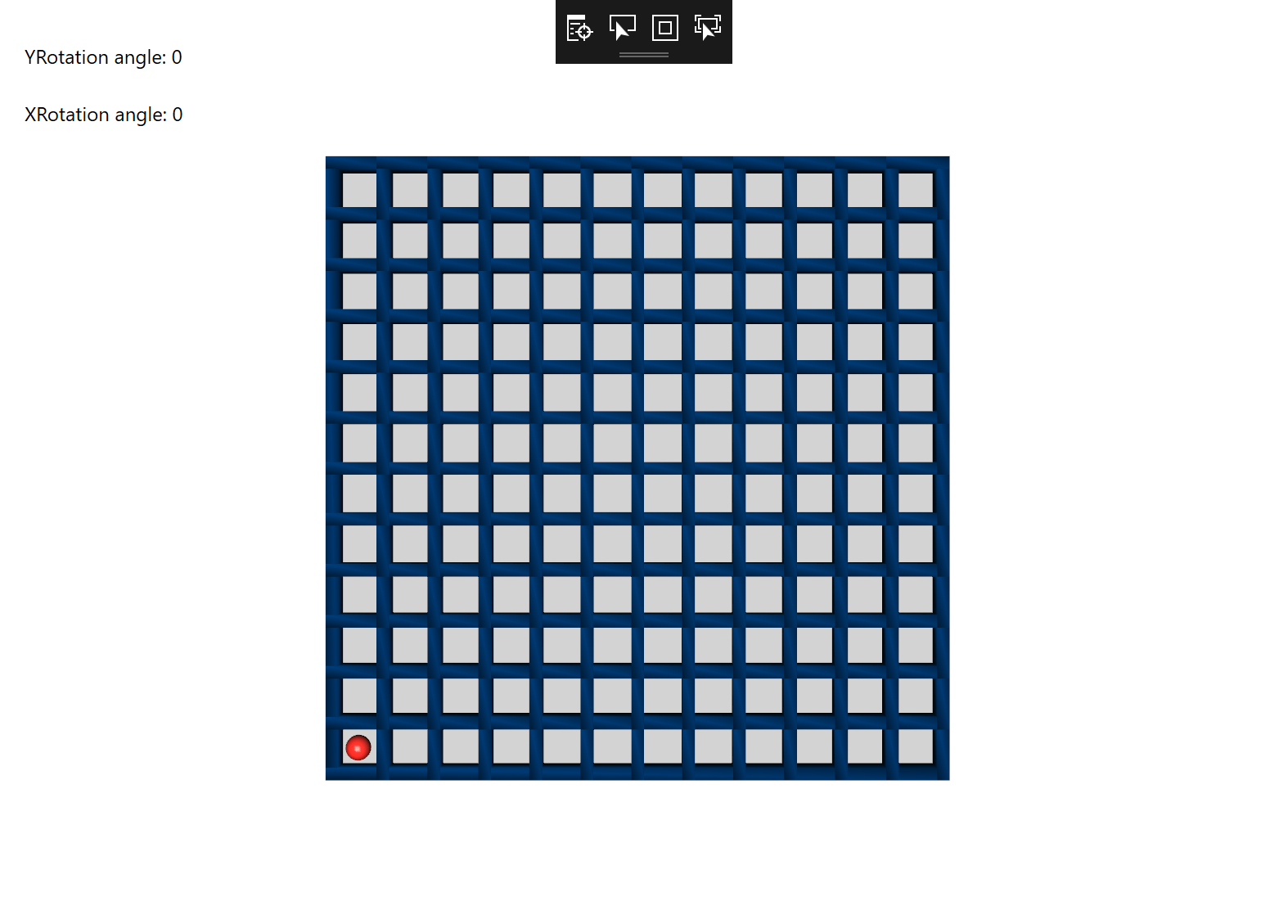
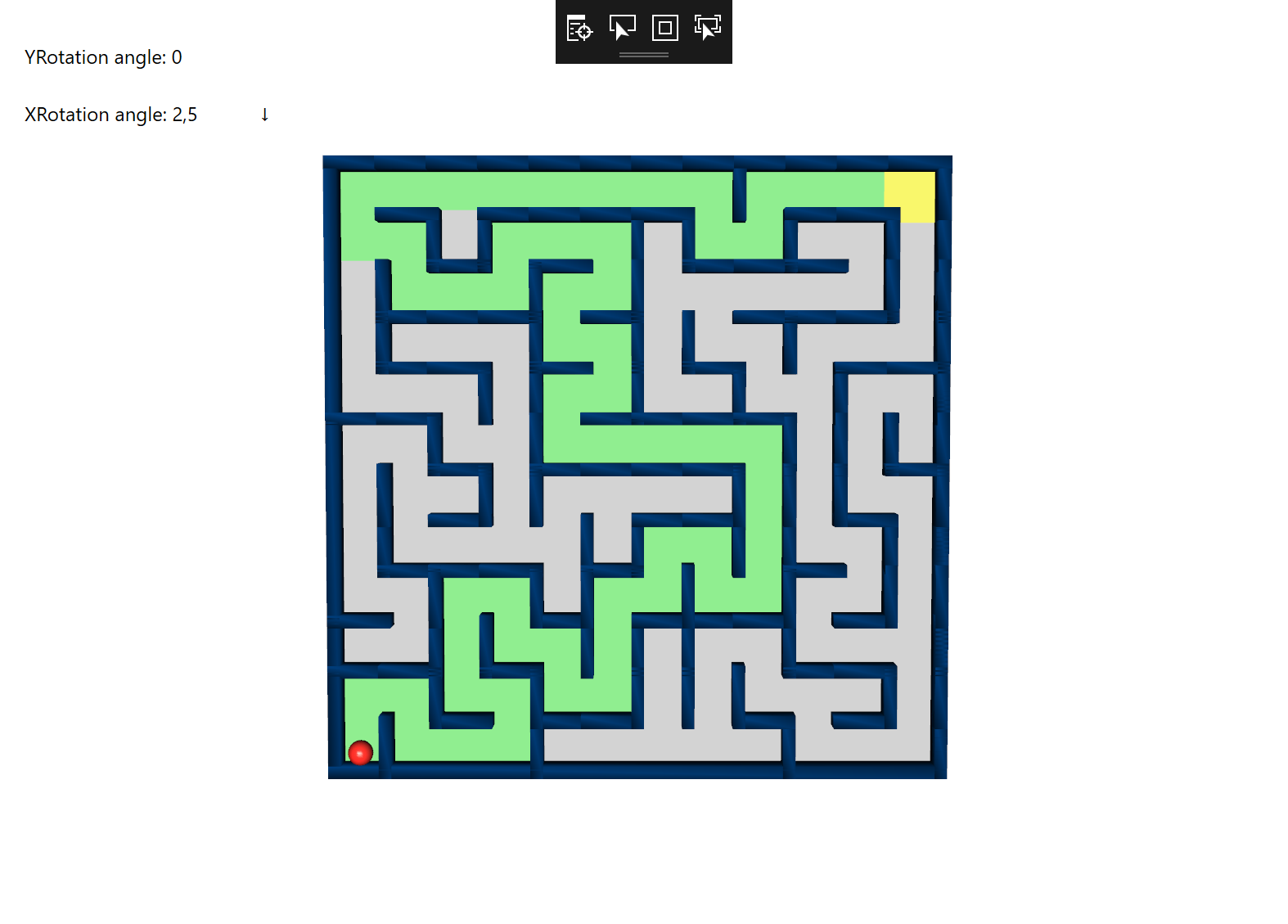
De werking van de snelheidsberekening, die hierna uitgelegd gaat worden, gebeurt voor zowel de snelheid op de x-as als die op de y-as.  
De berekening van de snelheid gebeurt door eerst de huidige hellingsgraad van het doolhof te nemen, en aan de hand daarvan de versnelling te berekenen. Deze versnelling wordt dan op de vorige snelheid toegepast om de huidige snelheid te bekomen.  
De voorgaande positie van de bal wordt dan genomen, en aan de hand van de huidige snelheid van de bal en de tijd die verstreken is sinds de laatste berekening(10ms) wordt de huidige positie van de bal berekend. Dit wordt vervolgens in een variabele gestoken. Als de graphics-timer de zichtbare locatie van de bal verandert, haalt deze de locatie van de bal, gehaald uit de voorgaande variabele.

Collision

De bewegende bal zal, als die tegen een muur komt, moeten stoppen. Dit heet collision. In deze toepassing zal elke keer als de logica-timer berekeningen uitvoert de locatie van de bal vergeleken worden met elke muur in het doolhof. Als het blijkt dat de volgende locatie van de bal met de huidige snelheid in een muur terecht zal komen, dan gaat de huidige snelheid in die richting gehalveerd en geïnverteerd worden (snelheid op x-as of y-as). Dit zorgt voor een bots-effect op de bal.

## Maze algoritme

Alle berekeningen gebeuren in de klasse MazeGenerator. Deze heeft een lijst van alle nodes en een lijst van alle muren als property. Vervolgens heeft de klasse ook nog een stack voor de bezochten nodes als property.   
  
De methode VisitNode doet alle berekeningen van het algorithem. De methode heeft als parameter een Node die bezocht wordt genaamd “visit”. Van de node visit wordt dan de bezoekbare buur-node gezocht deze word dan recursief met dezelfde methode VisitNode terug aangehaald om zo het proces te herhalen. Ook als er geen buur-nodes zijn zal de methode de vorige node van de stack halen en deze et VisitNode aanhalen. Dit gebeurd tot de stack leeg is.  
Als de rechtberboven node de volgende node is word de stack op dat moment opgeslagen. Dit is dan het uiteindelijke pad naar het einde.  
Ten slotte worden de nodes en de muren gerenderd.

Maze generation

# Code structuur

De solution bestaat uit 3 projecten. Globals, Logic en Maze.

## Maze

Dit is een WPF-project waar het uiteindelijke doolhof gegenereerd word. In XAMP wordt enkel de bal gegenereerd.

In maze word de Mazegenerator-klasse aangeroepen om het doolhof te generen en de Physics-klasse heeft methodes die gebruikt worden voor de fysica van de bal.

## Globals

Het project Globals bevat twee klassen Wall en Node. Wall bevat alle informatie over een muur. De punten van het 3D object, het centrum, de breedte, de hoogte, …. De klasse Wall bevat ook het 3D object van de muur die gerenderd zal worden in het WPF-project.

Node bevat ook alle 3D-informatie over een Node en houd ook bij of deze al bezocht is.

## Logic

Bevat twee klassen die al het rekenwerk van het programma doen.

# Besluit

De opdracht was een goede manier om te leren werken met WPF 3D-game componenten.

Het gebruik van WPF laat een applicatie developer toe om een mooiere user interface voor zijn/haar programma te maken. Dit is belangrijk om een eigen stijl aan een applicatie te kunnen geven en zorgt dat de developer niet gelimiteerd is tot windows components.

Het tonen van 3D-objecten in WPF gaat, en is goed voor simpele toepassingen zoals deze. Voor grotere toepassingen wordt bijvoorbeeld openGL gebruikt. Dit maakte het moeilijker om documentatie te vinden voor 3D-objecten in WPF. Dit was vooral weten hoe, meer dan denkwerk.   
De fysica implementeren is een wat moeilijkere opdracht. Hierbij komt buiten de fysische formules ook veel logisch denken bij te kijken. Dit vergt veel zoekwerk. Maar eens de nodige formules werkende zijn, gaat het verder uitwerken van de fysica (zoals botsingen, weerstand, ...) veel vlotter.

Voor het uitwerken van het algoritme ging een groot deel van het werk in het bestaande programma herstructureren zodat de implementatie van het “Depth-first search” algoritme vlot zou verlopen.   
De uiteindelijke werking van het programma is goed verlopen en het pad wordt duidelijk aangeduid. Dit hele project heeft ook meer inzicht in de werking van path finding AI gegeven.

# Referenties

|  |  |
| --- | --- |
| [1] | S. Sexton, "Sean's stuff," [Online]. Available: https://stuff.seans.com/2008/08/13/drawing-a-cube-in-wpf/. |
| [2] | G. Arien, "Geert Ariën's Programming Portfolio," [Online]. Available: https://geertarien.wordpress.com/2014/12/06/3d-maze/. |
| [3] | D. Andrew and L. Ali, "Physics lecture demonstrations at Boston University," [Online]. Available: http://physics.bu.edu/~duffy/. |
| [4] | A. Duffy and A. Loewy, "A Race: Rolling Down a Ramp," [Online]. Available: http://physics.bu.edu/~duffy/sc527\_notes06/race.html. |
| [5] | "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Maze\_generation\_algorithm. |
| [6] | CodeWarrior, "Stackoverflow," [Online]. Available: https://stackoverflow.com/a/10232354. |