

Odisee Hogeschool

Gebroeders de Smetstraat 1, 9000 Gent

Advanced Applied Programming Project 1:

Opdracht 3D

Steven Impens

Elektronica-Ict

2017-2018

Inhoudsopgave

[Codefragmentenlijst 4](#_Toc512345823)

[Figurenlijst 5](#_Toc512345824)

[Inleiding 6](#_Toc512345825)

[1 Onderzoek 3D 7](#_Toc512345826)

[1.1 Windows Presentation Foundation 7](#_Toc512345827)

[1.2 3D Scene in WPF 7](#_Toc512345828)

[1.2.1 Camera 7](#_Toc512345829)

[1.2.2 Licht 8](#_Toc512345830)

[1.2.3 3D objecten 8](#_Toc512345831)

[2 Uitwerking 3D 9](#_Toc512345832)

[2.1 Controls 9](#_Toc512345833)

[2.2 3D Scene 10](#_Toc512345834)

[2.2.1 Spelbord 10](#_Toc512345835)

[2.2.2 Belichting 12](#_Toc512345836)

[2.2.3 Camera 12](#_Toc512345837)

[2.2.4 Muren 13](#_Toc512345838)

[2.3 Kantelen Spelbord 15](#_Toc512345839)

[2.4 Simpele animatie van de bal 16](#_Toc512345840)

[3 Onderzoek game fysica 18](#_Toc512345841)

[3.1 Game loop 18](#_Toc512345842)

[3.2 Wrijving 18](#_Toc512345843)

[3.3 Snelheid en Versnelling 19](#_Toc512345844)

[3.4 Botsing 19](#_Toc512345845)

[4 Uitwerking game fysica 20](#_Toc512345846)

[4.1 Game loop 20](#_Toc512345847)

[4.2 Wrijving 21](#_Toc512345848)

[4.3 Versnelling 22](#_Toc512345849)

[4.4 Botsing 22](#_Toc512345850)

[5 Besluit 23](#_Toc512345851)

[Literatuurlijst 24](#_Toc512345852)

# Codefragmentenlijst

[Codefragment 2‑1: UI controls 9](#_Toc512345853)

[Codefragment 2‑2: spelbord in XAML 11](#_Toc512345854)

[Codefragment 2‑3: Binding 11](#_Toc512345855)

[Codefragment 2‑4: Lichtbronnen 12](#_Toc512345856)

[Codefragment 2‑5: Camera 12](#_Toc512345857)

[Codefragment 2‑6: Aanmaken Driehoeken 13](#_Toc512345858)

[Codefragment 2‑7: Aanmaken kubus 14](#_Toc512345859)

[Codefragment 2‑8: Bord rotatie 15](#_Toc512345860)

[Codefragment 2‑9: Rotatie animatie van de bal 16](#_Toc512345861)

[Codefragment 4‑1 20](#_Toc512345862)

[Codefragment 4‑2: implementatie wrijving 21](#_Toc512345863)

# Figurenlijst

Figuur 1‑1: Camera [3] 7

Figuur 1‑2: Tetrahedron [5] 8

Figuur 2‑1: Coördinatensysteem 10

Figuur 3‑1: Vectoriele voorstelling wrijving [8] 18

Figuur 4‑1: wrijving over een helling [9] 21

# 

# Inleiding

Het doel van dit project is het ontwerpen van een 3D spelbord waarbij een bal zich moet navigeren door een doolhof. De muren worden programmatisch toegevoegd. De bal kan navigeren aan de hand van interactie in de gui die het spelbord laten kantelen. Het is dus eerst de bedoeling een 3D geheel te ontwerpen waarna de controle van de 3D objecten en de animatie van een bal. Er wordt gebruik gemaakt van het WPF-raamwerk en C# in Visual studio. In het eerste deel is onderzocht wat WPF is en hoe een 3D scene is opgebouwd. In het tweede deel hoe alle nodige elementen worden geïmplementeerd.

# Onderzoek 3D

## Windows Presentation Foundation

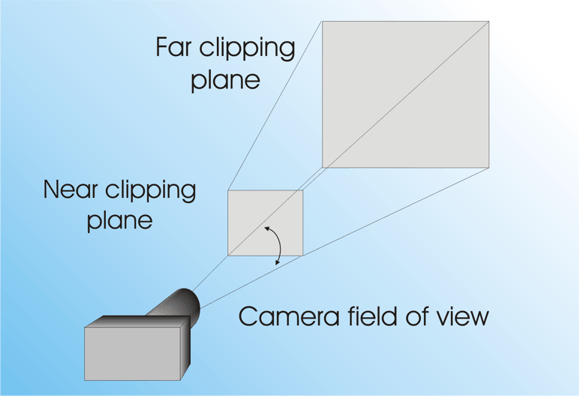
WPF kort voor Windows Presentation Foundation is een grafisch systeem voor het renderen van userinterfaces. WPF maakt gebruik van DirectX, een bundel van Application Programing interfaces. Deze APIs worden toegepast voor multimedia en games op het Windows platform. WPF heeft veel verschillende features waaronder Data Binding voor de interactie en manipulatie van data en Animaties voor het laten bewegen van objecten. Er wordt gebruik gemaakt van Extensible Application Markup Language, XAML. Hiermee is het mogelijk om delen van een programma te definiëren zonder code te schrijven maar deze methode is niet noodzakelijk sinds alles ook met code kan worden opgebouwd. [1] [2]

## 3D Scene in WPF

Een 3D scene in WPF heeft enkele onderdelen nodig. Een viewport waarin alle 3D objecten in bestaan, dit is een soort virtueel universum. Om de objecten die zich bevinden in een vierport te zien is er een camera en een lichtbron nodig. De camera kan enkel de zichtbare delen van opgelichte 3D objecten zien.

### Camera

Een camera in WPF heeft een minimum en maximum bereik (Far clipping en Near clipping) alsook een Field of View (Figuur 1‑1: Camera [3]). Deze eigenschappen zorgen voor de zichtbaar ruimte. Indien een object uit deze zone ligt dan zal deze niet meer zichtbaar zijn. De camera heeft ook een positie in de viewport en een kijkrichting. Het is ook noodzakelijk om de bovenkant mee te geven zodat de camera weet welke kant de bovenkant is.



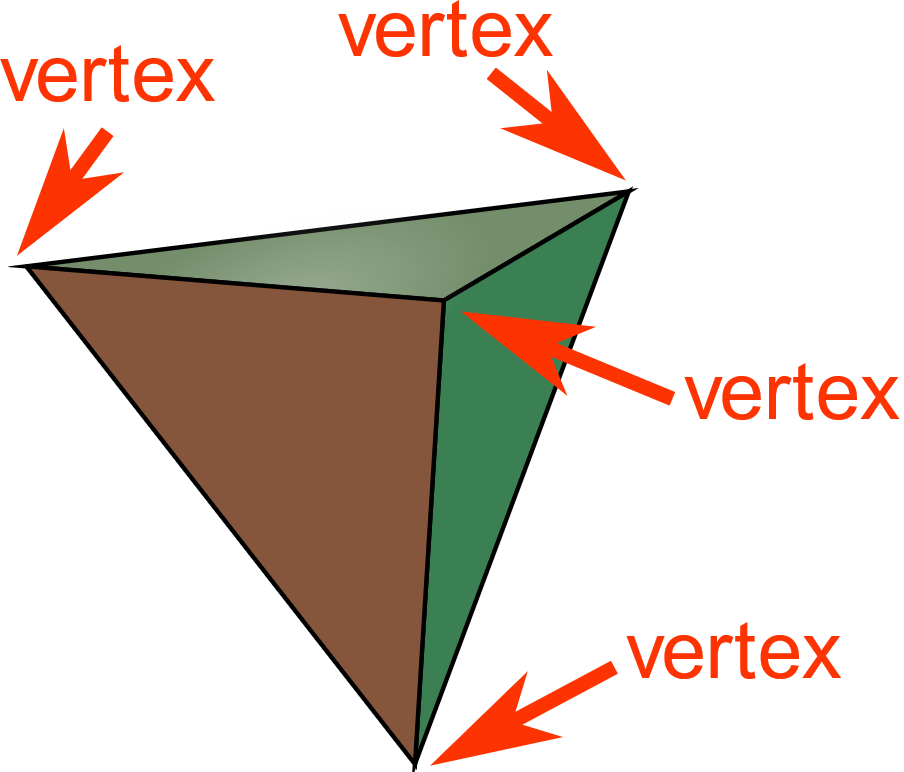
Figuur 1‑1: Camera [3]

### Licht

Lichtbronnen zorgen ervoor dat objecten worden opgelicht en dus zichtbaar zijn. Zonder licht is er ook geen reflectie en kan de camera niets opnemen. Licht is dus noodzakelijk voor een 3D scene. WPF bevat vier soorten lichtbronnen. AmbientLight zorgt voor een lichtinval op alle objecten ongeacht locatie of oriëntatie, alle vlakken worden opgelicht. In tegenstelling tot een DirectionalLight die geen locatie heeft maar wel een richting, alle objecten worden opgelicht vanuit een ingestelde directie. Er bestaan ook meer specifiek lichtbronnen, de PointLight en de SpotLight. De eerste is net zoals een lamp en zal dus rond de bron verlichten in een gespecifieerd bereik alsook vermindering van sterkte naargelang de afstand. De tweede is een gerichte lichtbron in de vorm van een kegel.

### 3D objecten

Alle 3D objecten worden beschreven door een verzameling van driehoeken. De software die de objecten weergeeft kan de kleur van elke driehoek berekenen aan de hand van hun materiaal en de lichtinval van alle lichtbronnen. De oppervlakte van een 3D object genaamd een mesh bestaat uit vertices. Dit zijn de punten waaruit de driehoeken worden gevormd. Figuur 1‑2: Tetrahedron is een 3D object bestaande uit vier vertices die samen driehoeken vormen tot een geheel. Enkel de voorkant van een mesh wordt weergegeven. Om een driehoek weer te geven zijn er 3 coördinaten nodig en hun relatie met elkaar. Hun relatie moet in een tegenwijzers zin gedefinieerd worden anders zal de achterkant als voorkant gezien worden wat de driehoek dus niet zichtbaar maakt. Dit is vooral opvallend in volledige objecten. Door een verkeerd ingestelde driehoek zal er een gat ontstaan in het object. [4]



Figuur 1‑2: Tetrahedron [5]

# Uitwerking 3D

## Controls

De interface wordt opgedeeld in twee delen namelijk een plaats voor de 3D scene waar het spelbord zich in bevindt en een plaats voor de bedieningselementen die zorgen voor het kantelen van het bord. Hiervoor wordt gebruik gemaakt van een Grid element. De interface kan zo ingedeeld worden als een tabel waarbij de hoogte van een rij wordt ingesteld.

Sinds dat het bord in 2 kanten moet draaien zullen er 2 elementen nodig zijn. Hiervoor zijn sliders een goede keuze. De sliders worden in het midden afgesteld zodat ze de waarde van nul graden hebben. Indien een slider naar links wordt verschoven zal een negatieve hoek ingesteld worden, rechts een positieve hoek. Een slider zal zorgen voor de rotatie rond de x-as terwijl de andere zorgt voor een rotatie rond de z-as. Beide sliders worden in een StackPanel gestoken, hierdoor kan hun positie makkelijk worden uitgelijnd. Het panel krijgt ook een rij nummer, namelijk 0 zodat dit gedeelte als eerste wordt getoond in de interface. De viewport zal de 3D scene bevatten met het spelbord.



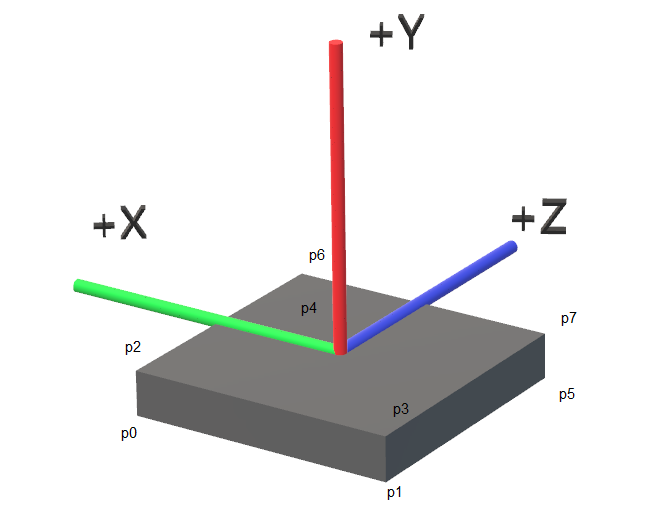
Codefragment 2‑1: UI controls

Beide sliders krijgen een “ValueChanged” attribuut, als de slider wordt gebruikt zal de bijhorende functie aangeroepen worden. De functie zal dan de code uitvoeren om het bord te kantelen.

## 3D Scene

### Spelbord

Het spelbord is een 3D object en heeft een Mesh nodig. Een ModelVisual3D zal content moeten bevatten namelijk een GeometryModel3D, hierin wordt niet enkel de geometrie van het bord in gedefinieerd maar ook het materiaal ervan. De Mesh van het spelbord is beschreven in het MeshGeometry3D element. Zoals beschreven in het onderdeel 1.2.3 3D objecten moet een mesh bestaan uit vertices en hun positie. Een bord met een lengte en breedte van 100 en een dikte van 2 is opgebouwd uit 8 vertices en 12 driehoeken.



Figuur 2‑1: Coördinatensysteem

Punt 0 zal in de “Positions” attribuut op de eerste plaats komen en heeft als coördinaten (50,-2,-50), punt 1 met (-50,-2,50) op de tweede plaats gescheiden door een spatie. Om het midden van het bord op het nulpunt te krijgen zal het bord spreiden van 50 tot -50 op de x- en z-as. De gebruikte positie verdeling is weergegeven op Figuur 2‑1: Coördinatensysteem.

Na het toevoegen van de punten moeten ze verbonden worden voor het maken van driehoeken. De driehoek van punt 0, 1 en 3 en die van punt 3, 2, 0 vormen samen 1 vlak. Deze punten moeten tegenwijzers zin toegevoegd worden anders kan het zijn dat een deel van het bord niet juist weergegeven wordt en er dus een opening zal ontstaan op de fout ingeven combinaties. De combinaties worden toegevoegd aan de “TriangleIndices” attribuut.

Sinds dat WPF automatisch het voorste vlak kiest kan het zijn dat de belichting niet correct op het bord neerkomt dit kan verholpen worden door de “Normals” attribuut. Door elk punt een vector te geven van (0,1,0) hierdoor richt elk vlak naar boven waar er een lichtbron is.

Het is nu nog nodig om een materiaal toe te voegen. Om een afbeelding als textuur te gebruiken is het nodig om “TextureCoordinates” in te geven. Het bovenste vlak moet de textuur krijgen dus de 4 bovenste punten moeten overeenkomen met de coördinaten voor de afbeelding. Punt 2 krijgt een coördinaat van (0,0), punt 3 (0,1), punt 6 (1,0) en punt 7 (1,1). De andere punten krijgen (0,0). Er is alsook een “BackMaterial” aanwezig om ervoor te zorgen dat de onderkant van het bord niet doorzichtig is.

Alles bij elkaar krijgen we een 3D object die behoord tot de content van een “ModelVisual3D”.



Codefragment 2‑2: spelbord in XAML

Het is ook mogelijk om de Positions en TriangleIndices via code toe te voegen aan de XAML. Er moet dan gebruik gemaakt worden van “Data Binding”. Dit is een feature van WPF waarbij er verwezen wordt naar een functie in de C#-code. De functie geeft dan de juiste waardes mee. Sinds dat het spelbord statisch is en niet moet veranderen is het handiger om deze feature niet te implementeren. In Codefragment 2‑3 is Binding toegepast waarbij FloorPoints3D en FloorPointsIndices de functies zijn.



Codefragment 2‑3: Binding

### Belichting

Alle 3D objecten moeten verlicht zijn om ze te kunnen zien. Om dit te verwezenlijk is gebruikt gemaakt van een “AmbientLight”. Die zorgt ervoor dat alles verlicht wordt ongeacht de positie en oriëntatie van het 3D object. Om dit licht niet te uitbundig te maken is een donkere kleur beter. Voor een meer gefocust licht op het bord te krijgen is een “DirectionalLight” van toepassing. Deze heeft een lichtere kleur zodat er duidelijk meer natuurlijk licht aanwezig is. Dit soort moet ook een richting krijgen namelijk (0,-1,-1). Hierdoor zal het licht schijnen naar beneden recht op het spelbord.



Codefragment 2‑4: Lichtbronnen

### Camera

De camera moet gericht zijn op het spelbord. Om dit te verwezenlijken krijgt de camera een positie boven en ver van het bord. Sinds dat het bord in het nulpunt ligt moet da camera in een negatief Z-coördinaat liggen (0, 120, -120) om in een positieve richting te kijken. Door deze coördinaten te gebruiken ligt het bord in het midden en volledig in de zichtbare regio. Het is noodzakelijk om een “UpDirection” mee te geven anders weet de camera niet wat boven en onder is.



Codefragment 2‑5: Camera

### Muren

De muren van het spel worden dynamisch toegevoegd aan de 3D scene via C#-code. In de XAML word een “ContainerUIElement” toegevoegd die tevens ook een naam krijgt met de “x:Name” attribuut. Hierdoor kan de container aangesproken worden vanuit de code.



Sinds dat 3D objecten bestaan uit driehoeken is er een methode nodig die deze kan aanmaken. In de Triangle methode wordt een MeshGeometry3D aangemaakt van 3 punten, hun relatie en de Normals. De methode geeft een Model3DGroup terug met een GeometryModel3D van een driehoek en een materiaal.



Codefragment 2‑6: Aanmaken Driehoeken

De Cube methode zal de nodige driehoeken aanvragen en samenvoegen tot een geheel die de kubus vormt. Aan de hand van de meegegeven coördinaten en dimensies worden de nodige punten berekend en de driehoeken aangemaakt. Als alle driehoeken zijn toegevoegd zal er een ModelVisual3D teruggegeven worden die dan kan worden toegevoegd aan een viewport of de ContainerUIElement die voorzien is voor de muren.



Codefragment 2‑7: Aanmaken kubus

## Kantelen Spelbord

De bijhorende functies van de sliders in de interface roepen een functie op met de posities van beide sliders. De functie “ChangeBoardRotation” maakt een transformatie aan die roteert rond 2 assen namelijk de x-as en de z-as. De transformatie wordt toegepast op het spelbord, de muren en de bal. Deze functie is zo geprogrammeerd dat de rotatie van een as niet reset tijdens het roteren op een andere as.



Codefragment 2‑8: Bord rotatie

## Simpele animatie van de bal

Animaties gebeuren aan de hand van storyboards. Een storyboard kan meerdere animaties bevatten. Een animatie is een verandering van een waarde over een gespatieerde tijd. Om de bal te laten rollen moeten er 2 waarden veranderen namelijk een hoek voor de rotatie en een afstand.

Het aantal graden dat de bal draait hangt af van de afstand die hij aflegt. Als de bal een volledige rotatie maakt van 360 graden dan zal er een afstand van 2\*pi\*radius afgelegd worden.



Codefragment 2‑9: Rotatie animatie van de bal

Sinds dat de bal niet in het nulpunt ligt is het zeker nodig zijn middelpunt coördinaten in te stellen anders zal de rotatie gebeuren tegenover de assen van de 3D scene en niet de assen van de bal zelf. Gedurende de translatie animatie updaten die coördinaten zelf. Hierdoor moeten ze niet constant meegegeven worden aan de animatie voor de rotatie.

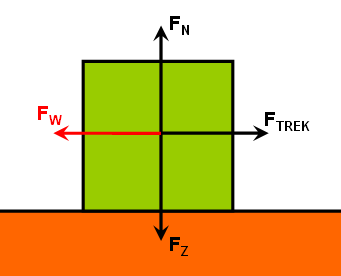
# Onderzoek game fysica

## Game loop

De game loop zorgt ervoor dat om een aantal milliseconden de nodige fysica berekening worden uitgevoerd. Een vooringesteld timer kan de nodige methodes oproepen, deze implementatie wordt ook wel “Fixed Time Step” genoemd. Dit is zeer gemakkelijk te implementeren doordat de berekening stap per stap worden uitgevoerd. Trage computers hebben een groot impact op de snelheid waarbij de timer de berekeningen doorloopt waardoor het geheel trager zal werken. Bij snelle computers is het probleem dat de berekeningen snel gebeuren en de er moet gewacht worden op de nieuwe berekeningen. Een variant hierop is de “Variable Time Step” waarbij de code kan doorlopen worden aan de snelheid van de pc. Deze implementatie bij tragere systemen kan ervoor zorgen dat waardes veel te snel veranderen terwijl de Render loop niet kan volgen. Hierdoor kan het lijken alsof objecten verspringen en helemaal geen fysica wetten volgen. [6] [7]

## Wrijving

Wrijving ontstaat bij de verschuiving van twee oppervlaktes. De wrijvingskracht FW werkt in de tegengestelde richting waardoor er een negatieve versnelling voorkomt, een vertraging. Een object in beweging zal vertragen tot stilstand als er geen andere krachten aanwezig zijn. De grootte van de wrijving hangt af van de normaalkracht FN en de wrijvingscoëfficiënt μ. De normaalkracht is de kracht loodrecht op de oppervlakte. Bij een voorwerp die recht op een oppervlakte steunt zal de normaalkracht gelijk zijn aan de zwaartekracht. De wrijvingscoëfficiënt hangt af van de materialen. [8]



Figuur 3‑1: Vectoriele voorstelling wrijving [8]

## Snelheid en Versnelling

Een voorwerp valt door de valversnelling die gecreëerd wordt door de zwaartekracht. Een snelheid kan berekend worden door de beginsnelheid op te tellen met de versnelling vermenigvuldigd met de tijd. Waarbij a de valversnelling is door de zwaartekracht. Een eenparige versnelling is de verandering in snelheid over een bepaalde tijdsduur.

Met de versnelling en de snelheid van een voorwerp kan een afgelegde afstand berekend worden.

## Botsing

# Uitwerking game fysica

## Game loop

Aan de hand van een timer worden de fysica berekening uitgevoerd. De timer zal om de aantal seconden een methode oproepen die nieuwe positie van de bal bepalen. De timer start de methode in een nieuwe thread waardoor het nodig is om een Dispatcher te gebruiken waardoor de berekeningen kunnen uitgevoerd worden op waardes vanuit de standaard thread. Het interval van de timer bevat het de tijdsduur van een frame in milliseconden. In zo frame wordt de nieuwe positie van de bal berekend. De berekeningen zullen dus dit interval gebruiken als verlopen tijd maar dan in seconden. [9]

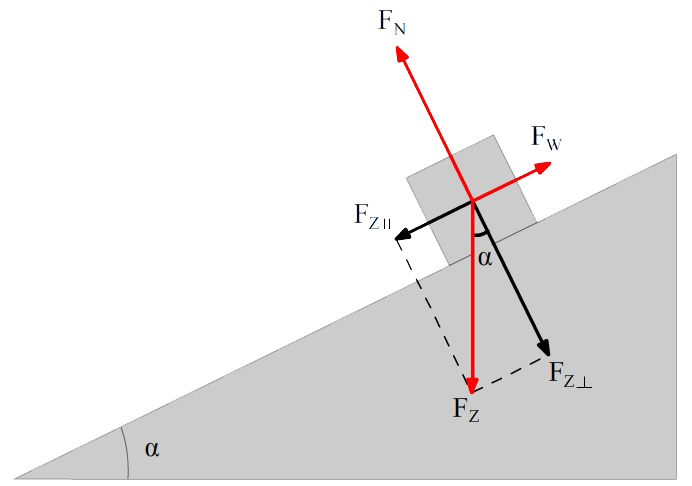


Codefragment 4‑1: Timer voor Game Loop

Een frame bevat de methodes die de afgelode afstand meegeven aan de translatie van de bal. Indien er een object in de weg staat zal da afgelegde afstand op nul worden gezet waardoor de bal niet meer verder zal bewegen.

## Wrijving

De wrijving van de bal wordt berekend aan de hand van de zwaartekracht en de wrijvingscoëfficiënt. Om een realistische wrijving te verkrijgen is gekozen voor een statische coëfficiënt van 0.05 en een dynamische coëfficiënt van 0.025. Dit zijn de waardes van een wrijving tussen metaal en hout. Indien de bal stilstaat wordt gebruik gemaakt van de statische waarde anders de dynamische. Doordat het spelbord gekanteld is zal niet de volledige zwaartekracht in werking zijn op de wrijving van de bal. De richting en grootte wordt bepaald door de cosinus met de hoek van het bord.



Figuur 4‑1: wrijving over een helling [10]

De formule van wrijving wordt: Waarbij



Codefragment 4‑2: implementatie wrijving

## Versnelling en snelheid

De versnelling van de bal gebeurd enkel door de zwaartekracht en de hoek van het spelbord.

Doordat het bord gekanteld is moet de valversnelling berekend worden over een hoek net zoals bij de wrijving maar nu met de sinus.



Codefragment 4‑3: Berekening versnelling

Door de wrijving en de versnelling kan de snelheid van de bal bepaald worden over een frame. De versnelling zal de balversnelling zijn met de wrijving verminderd. Om de acceleratie van de wrijving te verkrijgen word de wrijvingskracht gedeeld door de massa van de bal.



Codefragment 4‑4: Berekening snelheid

Met de versnelling en de snelheid van de bal kan een afgelegde afstand verkregen worden over een tijdeenheid, een frame.



Codefragment 4‑5: Berekening afstand

## Botsing

# Besluit

# Literatuurlijst

|  |  |
| --- | --- |
| [1] | Wikipedia, „Windows Presentation Foundation,” 31 3 2018. [Online]. Available: https://en.wikipedia.org/wiki/Windows\_Presentation\_Foundation. |
| [2] | Wikipedia, „DirectX,” 9 4 2018. [Online]. Available: https://en.wikipedia.org/wiki/DirectX. |
| [3] | M. James, „WPF The Easy 3D Way,” 10 6 2015. [Online]. Available: http://www.i-programmer.info/projects/38-windows/273-easy-3d.html?start=1. |
| [4] | C. Moser, „Introduction to WPF 3D,” 4 7 209. [Online]. Available: http://wpftutorial.net/IntroductionTo3D.html. |
| [5] | MathIsFun, „Vertices,” 2016. [Online]. Available: https://www.mathsisfun.com/geometry/vertices-faces-edges.html. |
| [6] | S. V. Impe, „Game Loops,” 24 11 2015. [Online]. Available: http://svanimpe.be/blog/game-loops-fx. |
| [7] | AmazingThew, „Fixed time step vs. variable time step,” Reddit, 2014. [Online]. Available: https://www.reddit.com/r/gamedev/comments/22k6pl/fixed\_time\_step\_vs\_variable\_time\_step/. |
| [8] | Wikipedia, „Wrijving,” 2017 10 31. [Online]. Available: https://nl.wikipedia.org/wiki/Wrijving. |
| [9] | B. Gideon, „Do C# Timers elapse on a separate thread?,” 20 11 2010. [Online]. Available: https://stackoverflow.com/questions/1435876/do-c-sharp-timers-elapse-on-a-separate-thread/1436331#1436331. |
| [10] | S. P. Dinkgreve, „Ontbinden van krachten,” 2018. [Online]. Available: https://wetenschapsschool.nl/chapter/Kracht\_5\_Ontbinden+van+krachten.html. |