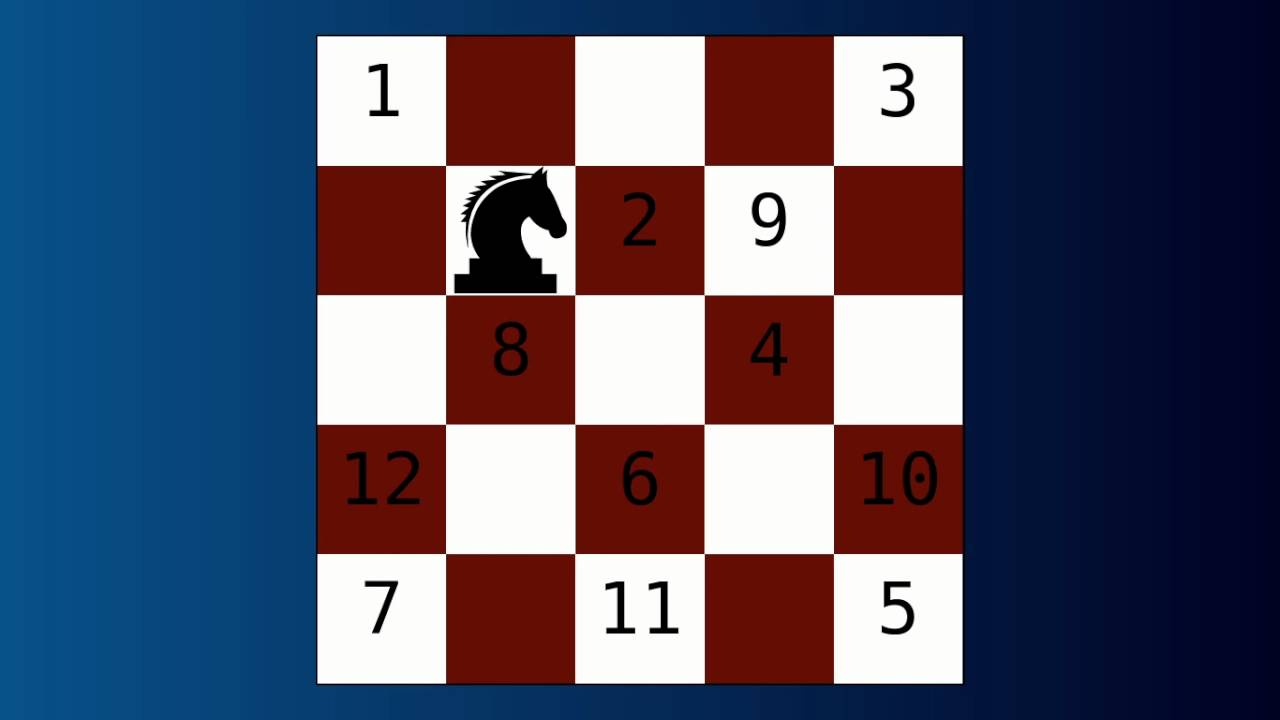
Plan van aanpak  
  
Knight’s tour oplossen met OpenCL  
  
Door: Dion Legierse  
Studentennummer: 2110603

# Inleiding

Voor dit project is het de bedoeling dat er kennis gemaakt gaat worden met de rekenkracht van grafischekaarten. Dit is omdat grafische kaarten heel snel en parallel kunnen rekenen. Twee bekende systemen om te werken met grafische kaarten zijn de OpenGL en OpenCL bibliotheken. Waarbij OpenGL is gericht op grafische berekeningen en een beeld printen op een scherm en OpenCL op het snel uitvoeren van berekeningen van matrixes en vectoren.

Voor dit project is er gekozen voor een implementatie met OpenCL. Dit onderwerp is interessant voor het oplossen van puzzels, specifiek het oplossen van puzzels via brute force. De puzzel die gekozen is om op te lossen is de Knight’s Tour puzzel. Knight’s Tour is een spel dat zich afspeelt op een klassiek schaakbord. Er wordt op het bord één paard neergezet. Het doel is dat het paard, volgens de standaard beweging van het paard, het hele bord afgaat zonder twee keer op het zelfde vakje te staan.

# Doelstelling

Voor het oplossen van de Knight’s Tour puzzel zijn er verschillende doelen opgesteld. Her eerste doel is het oplossen op de klassieke manier via de cpu van een computer. Dit is om de basis te leren begrijpen van het spel en hoe het verloop van het spel geimplementeerd kan worden in code. Het tweede doel is dit probleem vertalen naar OpenCL, waarbij de code van de cpu oplossing als referentie punt wordt gebruikt. Het laatste doel is een programma maken dat alle oplossing berekent voor elk vakje waarbij er gekozen kan worden tussen cpu en gpu of beide.

# Acties

* Installeren intel-sdk voor laptop en amd-pro-drivers voor desktop
* Installeren Rust toolchain in Visual Studio Code
* Test programma maken voor leren basis OpenCL in Rust
* Test programma uitbreiden voor verdieping OpenCL
* Implementeren van Knight’s Tour in de programmeer taal Rust voor cpu oplossing.
* Rust code vertalen naar een OpenCL implementatie.
* Cpu oplosser en gpu oplosser vergelijken.
* Oplossing van zowel cpu als gpu visueel opslaan.
* Programma maken waar keuze gemaakt kan worden tussen cpu, gpu of beide.
* Cpu oplossing multithreaden voor betere vergelijking.

# Benodigheden

* Rust toolchain
* Visual Studio Code
* OCL crate voor rust (OpenCL laag voor Rust)
* pc met gpu
* laptop met gpu

# Tijdspad

* Week 2:
  + Beginnen met het plan van aanpak
* Week 3:
  + Keuze maken probleemstelling voor OpenCL
  + Leren OpenCL
* Week 4:
  + Leren van OCL bibliotheek voor Rust
  + Bedenken oplos algoritme voor cpu
* Week 5:
  + Oplosser implementeren voor cpu
  + Testen cpu oplosser
* Week 6:
  + OpenCL oplosser implementeren in 1 dimensie
  + Cpu oplosser debuggen
* Week 7:
  + Gpu oplosser debuggen
  + Gpu oplosser implementeren in meerdere dimensies
* Week 8:
  + Gpu oplosser debuggen met meerdere dimensies
* Week 9:
  + Grafische opslaan van gevonden oplossingen via html
* Week 10:
  + Plan van aanpak afmaken, verbeteren, resultaten en reflectie toevoegen

# Resultaten

* Cpu oplosser die elke positie kan berekenen voor elk vierkant bord
* Een programma dat een oplossing kan berekenen voor elke positie van elk viekant bord, met de keuze dit te doen via gpu, cpu of beide.
* Een programma wat al het bovengenoemde kan en een gevonde oplossing kan opslaan in html format

# Reflectie

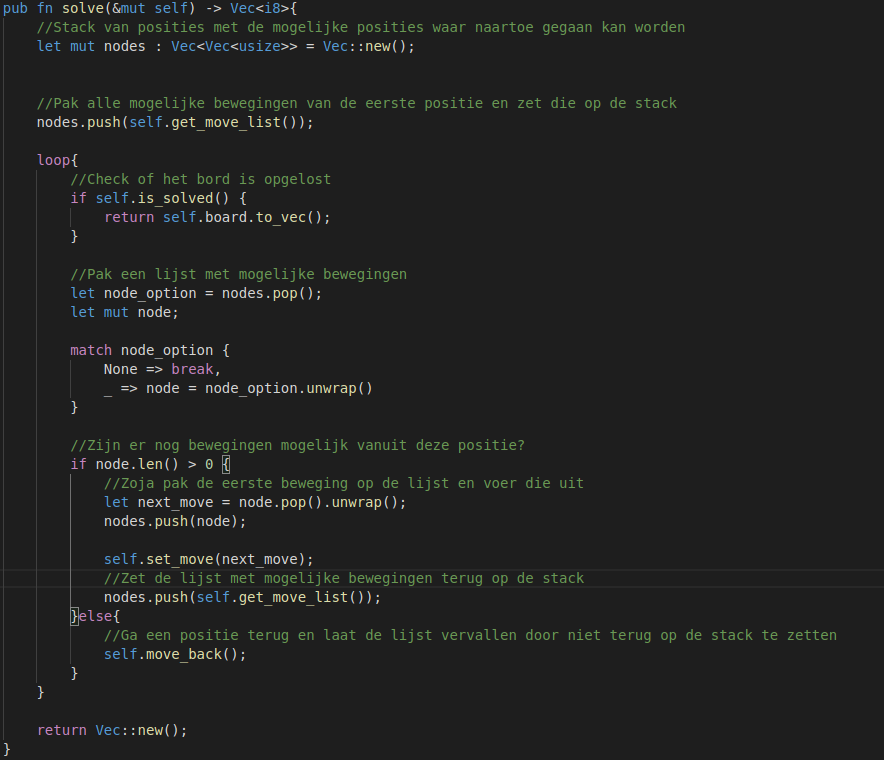
* Opzetten van OpenCL in Rust is goed te leren in korte tijd via de OCL bibliotheek
* Verdieping is mogelijk voor een uitgebreider programma via de OCL bibliotheek
* OpenCL zelf koste wat meer tijd om goed te leren, vooral aanmaken custom buffers en dimensies
* Grote verschillen tussen OpenCL compilers van AMD en intel voorbeeld:

//Bij intel mag dit wel maar niet bij AMD **const** **size\_t** size = **5**;  
**char** buffer[size];

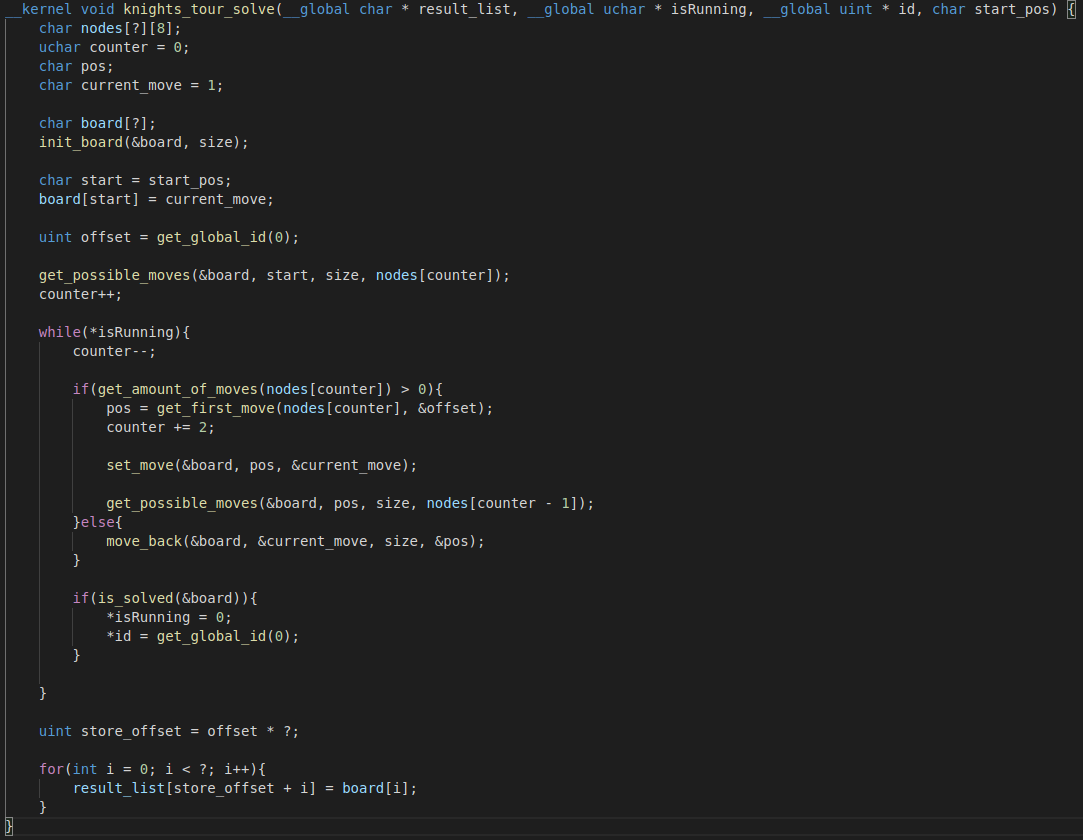
//Bij AMD moet het zo  
 **char** buffer[**5**];

# Bijlagen

## Cpu oplosser code



## Gpu oplosser code



# Bronnen

<https://docs.rs/ocl/0.19.3/ocl/>

<https://www.eriksmistad.no/getting-started-with-opencl-and-gpu-computing/>

<https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf>

https://github.com/DionLegierse/invd\_project