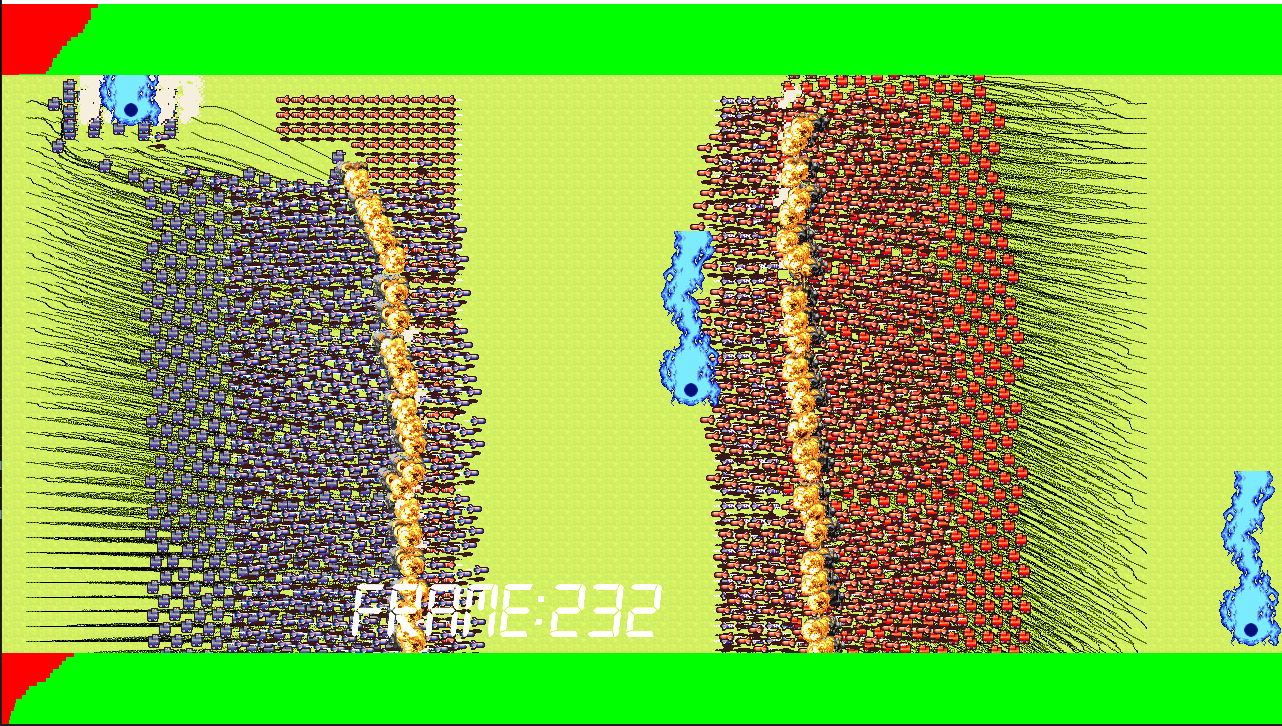
# Machine intelligence Verbeteringen



## Inhoud

[Machine intelligence 1](#_Toc76014771)

[Inhoud 2](#_Toc76014772)

[Samenvatting 4](#_Toc76014773)

[BIG O ANALYSIS 5](#_Toc76014774)

[Big o van de meegegeven code. 5](#_Toc76014775)

[Tank collision detection: 5](#_Toc76014776)

[FindClosestEnemy: 5](#_Toc76014777)

[Update rockets: 6](#_Toc76014778)

[Update Particle\_Beams: 6](#_Toc76014779)

[InsertionSortTanksHealth: 6](#_Toc76014780)

[Toegepaste verbeteringen 7](#_Toc76014781)

[De tank collision detection: 7](#_Toc76014782)

[Draw Sorted Healthbars: 7](#_Toc76014783)

[Update Particle\_Beams: 8](#_Toc76014784)

[Insertion Sort Tank Health: 8](#_Toc76014785)

[Concurrency 8](#_Toc76014786)

## Samenvatting

Eerste run. Zonder optimalisatie.



Algoritmische optimalisaties:

* Toevoegen van een grid. Toegepast op tank collision. Speedup 1.0 naar 1.9
* Healthbars worden getekend met alleen benodigde informatie. Ipv alle objecten binnen Tank. Speedup van 1.9 naar 2.0
* Healthbars Sorteren d.m.v een merge sort. In plaats van de huidige insertion sort. Geen grote speedup.
* Particle beams controleren alleen collision op tanks binnen de opgegeven locatie. Ipv alle tanks op het veld. Sim duurt korter maar niet genoeg voor een speedup.
* Toevoegen van grid op rocket collision speedup van 2.0 naar 2.7

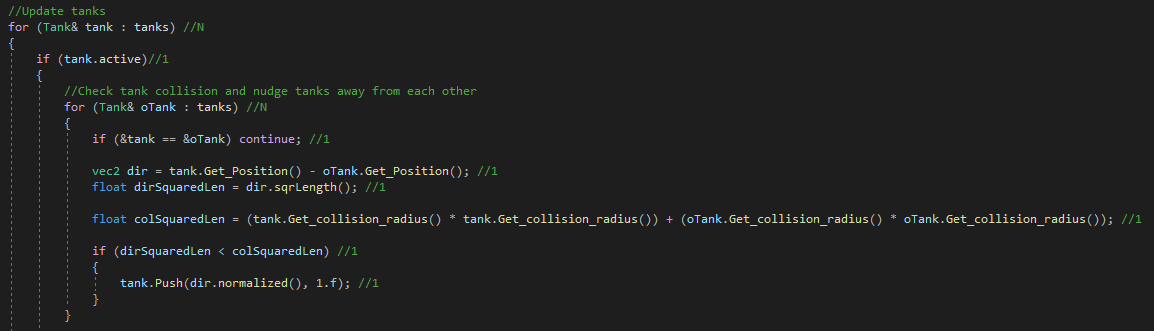
Concurrent optimalisaties:

* Threads worden dynamisch aangemaakt.

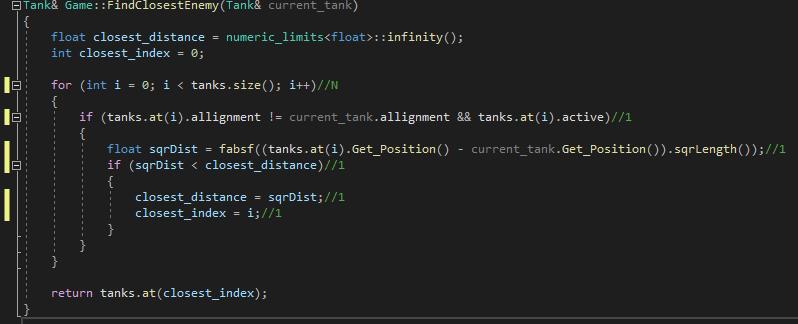
## BIG O ANALYSIS

### Big o van de meegegeven code.

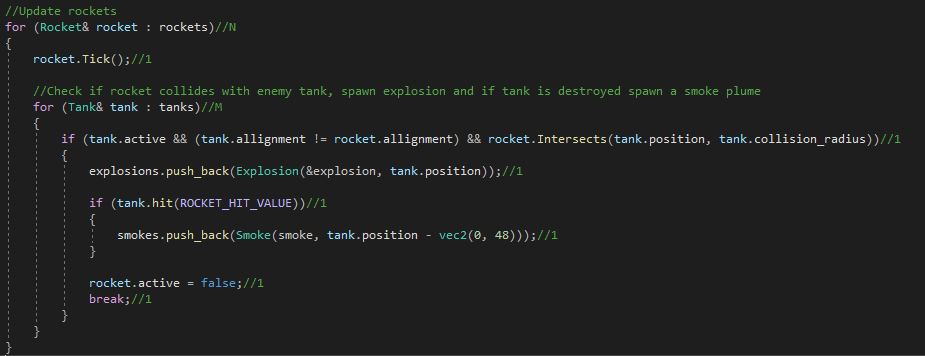
#### Tank collision detection:

O(N6N), O(N^2)

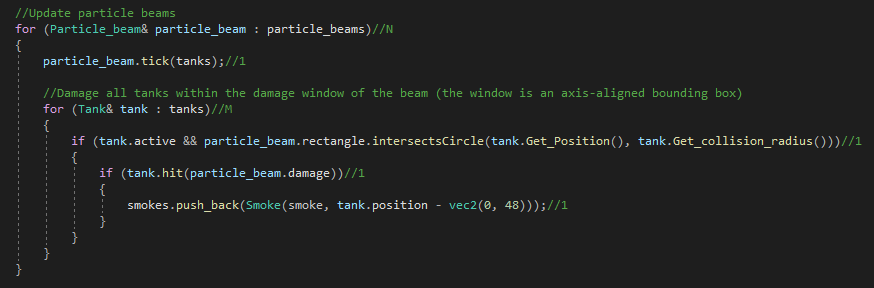
#### FindClosestEnemy:

O(5N), O(N)

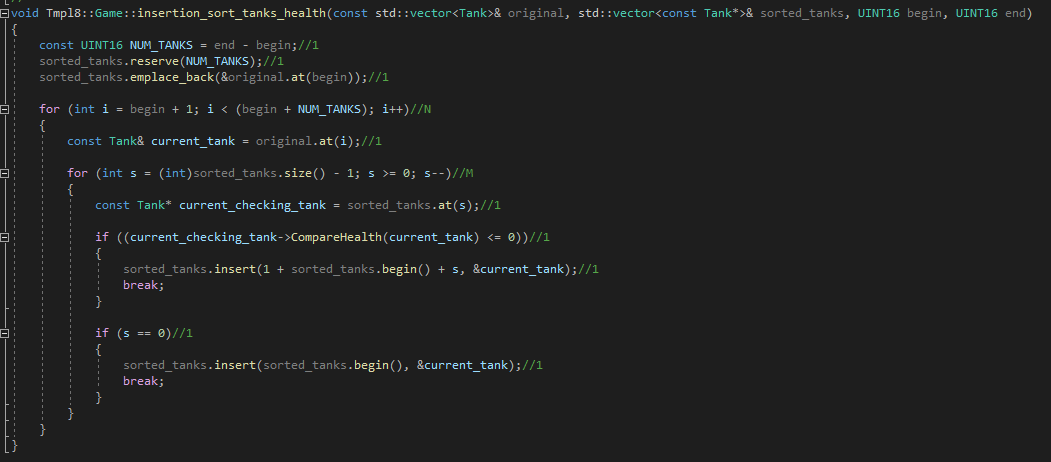
#### Update rockets:

O(N6M), O(NM)

#### Update Particle\_Beams:

O(N3M), O(NM)

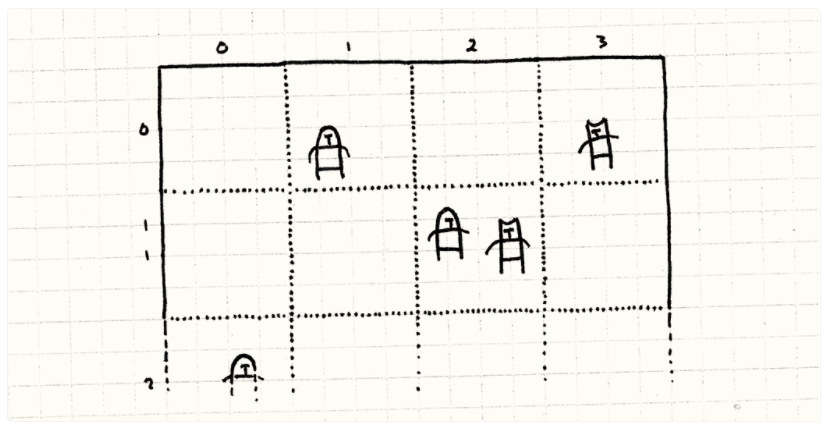
#### InsertionSortTanksHealth:

O(N5N), oftewel O(N^2)

### Toegepaste verbeteringen

#### De tank collision detection met spatial partitioning grid:

De *nested for-loop collision detection* en *particle beam checkt alle tanks.*Deze problemen kunnen opgelost worden door het verkleinen van de ‘n’ in n^2 algoritmen.

Het Spatial partitioning Grid is hier een oplossing voor. Zie Figuur. 

Grid voorbeeld

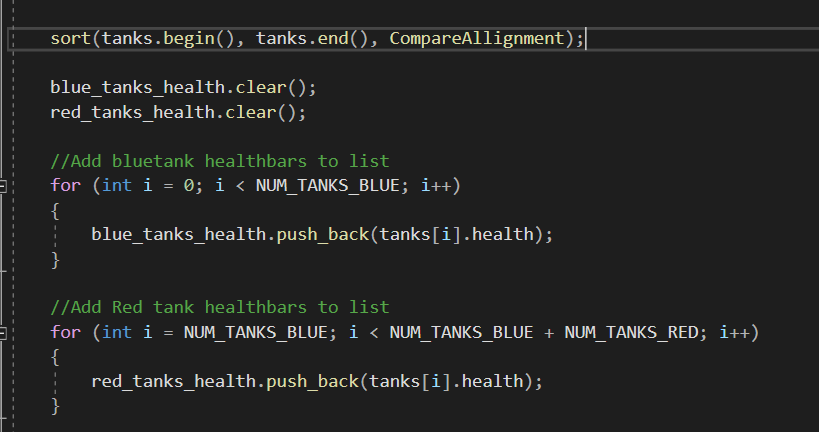
Met een grid verdelen we de tanks over meerdere cellen gebaseerd op de positie van de tank. Een cel bevat een lijst met tanks die dichtbij elkaar staan.   
Collision detection uitvoeren op tanks die niet dichtbij elkaar staan is hierdoor verholpen.

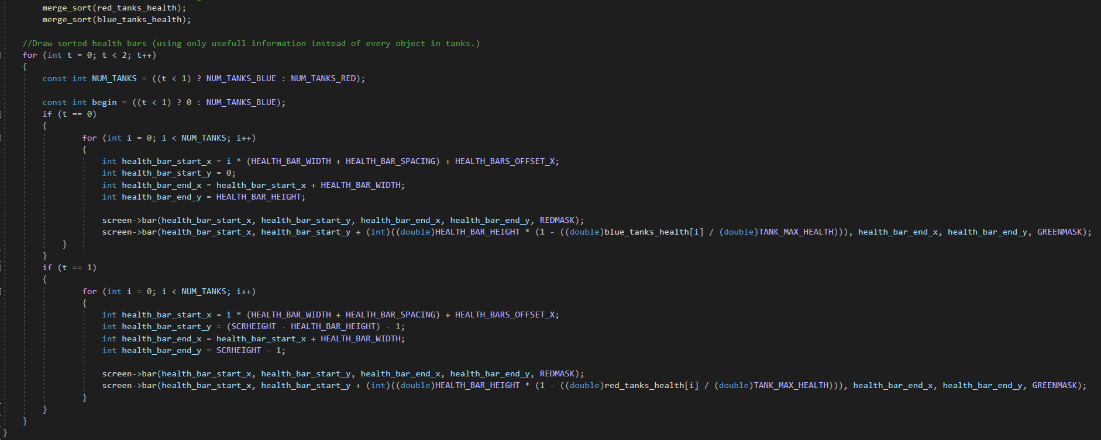
Door de tanks te verdelen over meerdere lijsten is de grote van elke lijst minimaal geworden. Wanneer we op elke cel de n^2 algoritmen los laten zal dit sneller zijn dan het checken van alle tanks in één keer. Een n^2 algoritme met een klein aantal *elements* zal significant veel minder *operations* vereisen dan eentje met een groot aantal *elements.* Dit resulteert in een snellere check van de tanks.

In de broncode zijn er twee files aangemaakt grid.h en grid.cpp. Deze bevatten de code voor een grid class.

In game.cpp wordt een grid object aangemaakt. Hier worden tanks aan toegevoegd. Wanneer een update plaatsvindt wordt er HandleCollision() aangeroepen op het grid object. Omdat de tanks bewegen wordt er na elke tank tick() gecheckt of de tank verplaatst moet worden in de grid. Dit doen we door move(Tank\* tank) op het grid object.

#### Draw Sorted Healthbars:



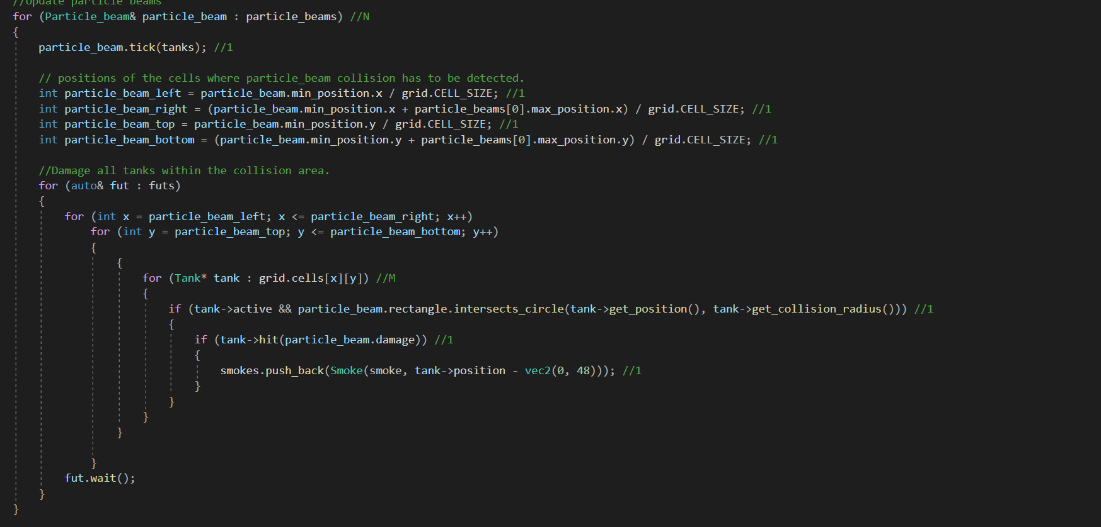


Elke update wordt tanks gesorteerd op alignement. Zo staan alle blauwe tanks in de eerste helft van de vector en de rode tanks op de tweede helft. Deze vectors worden gevuld met int waarden, bestaande uit de health waarden van de blauwe tanks en de health waarden van de rode tanks.

In de functie draw worden deze vectors gesorteerd en dit wordt gebruikt om de health bars te maken.

Hierbij is geen verandering van big O. Wel is dit efficiënter dan het sorteren en gebruiken van vectors gevuld met Tank objecten.

#### Update Particle\_Beams:



Particlebeams controleren alleen op collision op de tanks binnen de opgegeven area.

Ook hier is geen verandering van de Big O. Wel is dit efficiënter omdat er eerst gecontroleerd werd op collision voor elke tank op het veld.

#### Insertion Sort Tank Health:

De insertion sort is gewijzigd tot een merge sort.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Merge gedeelte: O(N)

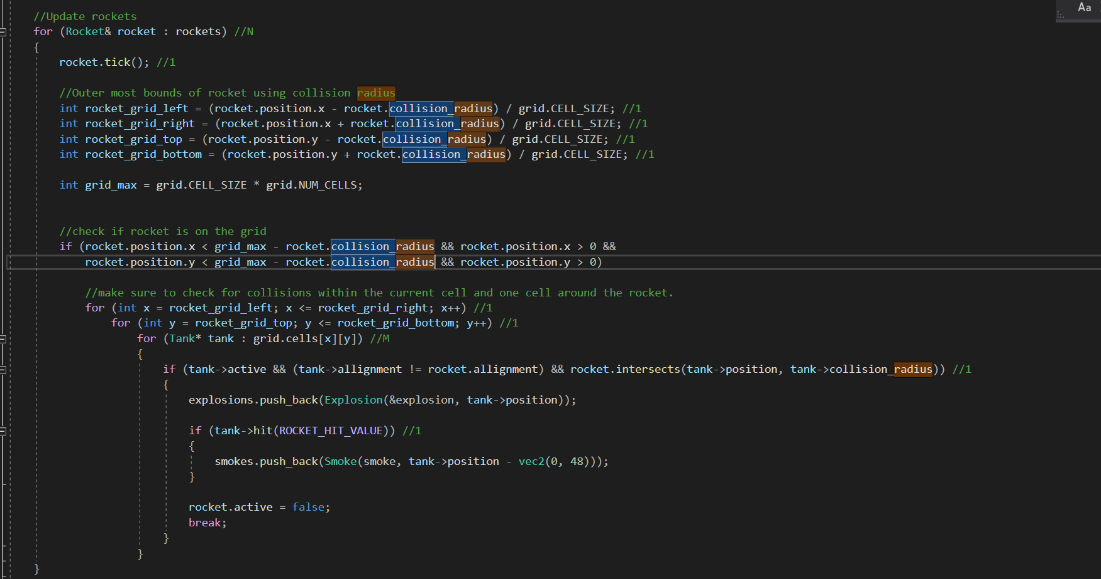
Afbeelding met tekst

Automatisch gegenereerde beschrijving

Divide gedeelte: O(logN)

Merge sort totaal: O(N\* logN)

#### Update Rockets:



## Concurrency