The tale of Roboboi

Ik kan met trots zeggen dat Roboboi vierde (eigenlijk derde als we Valentijn niet meerekenen) is geworden in het spectaculaire toernament! Hij was zeker niet perfect, maar was een waardige strijder en in dit document zal ik uitleggen hoe hij zo ver in de strijd wist te komen.

The origin of Roboboi

Toen ik begon aan deze opdracht had ik nog nooit een Behaviour Tree gemaakt en nog nooit van Robocode gehoord. Na de theorielessen van Valentijn werd het concept van een Behaviour Tree al vrij snel duidelijk voor mij. Vervolgens heeft Valentijn in de les voorgedaan hoe je een Behaviour Tree in praktijk op zou zetten. Hierna had ik er genoeg vertrouwen in dat ik zelf verder kon.

Het duurde echter wel even voordat ik Robocode begon te begrijpen. De limiterende functies waren fijn en overzichtelijk, maar waren zeker even wennen en daarbovenop was het ook enigszins verwarrend dat Behaviour Tree niet helemaal goed samen ging met de manier waarop Robocode opgezet was.

Ik schreef een aantal regels code en probeerde het een en ander uit. Ik kon al vrij snel mijn robotje laten rijden, maar moest toch even rondvragen hoe het een en ander nou precies in elkaar zat (zoals het debuggen van de radar). Ook was het, door een onbekend probleem, onmogelijk voor mij om mijn code te koppelen aan Robocode. Dit zorgde er voor dat ik constant handmatig mijn Robot in het juiste mapje moest slepen na het builden.

Hierna was ik vooral druk aan Project Context aan het werken en heb ik Robocode voor een tijdje laten liggen.

Robocode terug oppakken

Na een tijdje Robocode niet aangeraakt te hebben, zag ik dat een aantal klasgenoten toch wel redelijk geavanceerde robots hadden gemaakt. De deadline begon ook al wat dichterbij te komen, dus het werd voor mij tijd om weer eens aan mijn robocode te werken.

Om op mijn gemak te raken met de functies van Robocode, het gebruiken van de Behaviour Tree en het schrijven van wiskundige formules, ben ik begonnen met het schrijven van een simpel gedrag. Het idee was als volgt: laat de robot naar het midden rijden, laat de robot omhoog kijken en laat de robot heen en weer rijden (zodat hij robots in de buurt kan rammen). Ik vind wiskundige formules in games altijd erg leuk, maar heb er helaas niet enorm veel ervaring mee. Het kostte me daarom ook redelijk wat tijd om mijn robot naar het midden te laten kijken. Maar door deze “opdracht” aan mezelf te geven ben ik veel meer te weten gekomen over hoe robocode nou precies werkt en heb ik leren om te gaan met Behaviour Tree! Een geslaagde opdracht dus.

De code voor het rijden naar het midden.

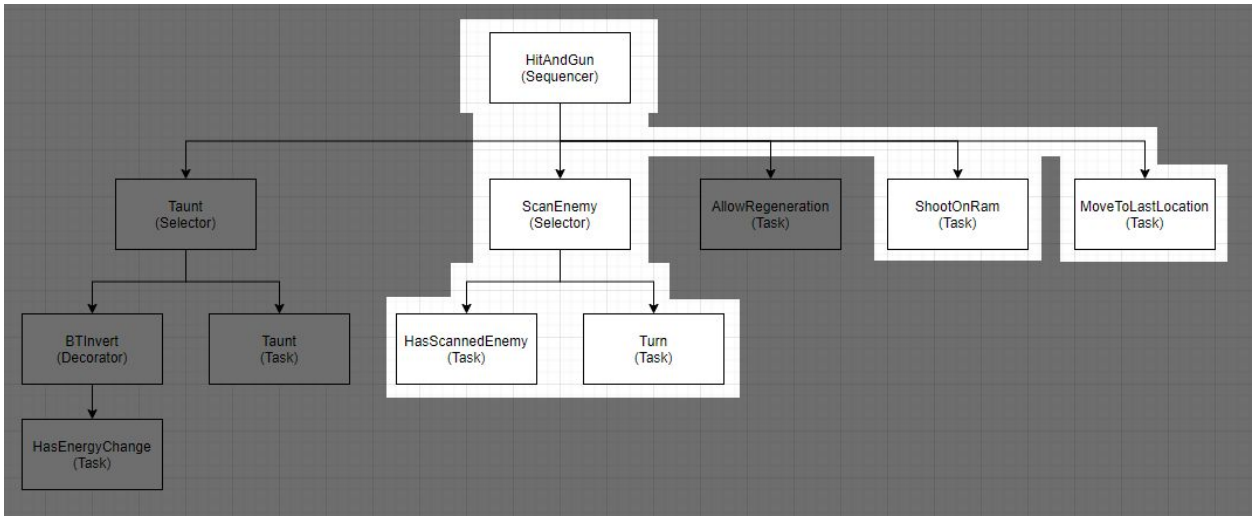
De Behaviour Tree voor dit gedrag.

Het echte gedrag maken

Toen we deze opdracht kregen had ik in eerste instantie het idee om de robot constant in zijn achteruit de laten rijden. Het idee hierachter was om robots die voorspellen waar een andere robot heen gaat te verwarren. Helaas kwam ik er tijdens het maken van mijn oefen gedrag achter dat dit niet zo werkt. Ik leerde van klasgenoten dat het slim was om naar andere robots te kijken voor inspiratie. Ik vond de “RamFire” robot erg intimiderend en merkte dat hij erg veel potjes won. Ik besloot toen om dezelfde tactiek te gebruiken voor mijn robot.

De tactiek is: scan voor een robot, rijd op de robot af, ram hem en schiet zo hard mogelijk een kogel zodra je hem ramt.

Om overzicht te krijgen heb ik voor het eerst in mijn leven van te voren een documentatie gemaakt. Ik heb de Behaviour Tree uitgetekend en ik moet zeggen dat het erg fijn werkte om een beeld te krijgen van hoe alles in elkaar komt te zitten.



Ik begon met 3 belangrijke componenten (zie afbeelding) en heb er later 2 aan toegevoegd.

In het begin kijkt de robot 360 graden om zich heen voor een vijandelijke robot. Zodra hij er een heeft gevonden onthoud hij waar dit was en rijd hij naar zijn laatste locatie. Omdat de robots breder zijn dan de scan straal, en omdat mijn radar automatisch met mijn robot meedraait richting de laatste locatie, ontstaat er uit zichzelf een soort “Lock” op de andere robot, ideaal!

Terwijl de vijandelijke robot gescanned blijft rijdt mijn robot op hem af, totdat hij hem uiteindelijk ramt. Dit vuurt het ram event af en deze sla ik op in de Blackboard. In de “ShootOnRam” task check ik of deze event null is. Als dit niet het geval is schiet ik een kogel naar de locatie van het event. Ik heb getest met verschillende tactieken wat betreft de kracht van de kogel, maar kwam er achter dat voor dit design de beste optie is om gewoon zo hard mogelijk te schieten als in de regels is toegestaan. Wel heb ik later toegevoegd dat mijn robot niet schiet als de vijand onder een bepaald energieniveau komt. Dit heb ik gedaan naar aanleiding van de extra bonus punten die je verdient door een robot te vermoorden door hem te rammen.

Ik heb de robot tegen alle standaard robots getest en was blij om te zien dat hij van zo goed als iedereen won! Zelfs tegen de “RamFire” robot won hij consistent, dit betekende dat ik een betere versie had gemaakt van de “RamFire”! Op dit punt was ik tevreden met het gedrag van mijn robot en wilde ik focussen op het maximaliseren van het aantal punten. Ik ben begonnen met het maken van een gedrag waar mijn robot de vijand niet aanvalt als hij onder een bepaalde energiegrens terecht kwam. Het idee was om de vijand dan weer energie te geven door te wachten tot hij mij aanvalt, om hem vervolgens weer aan te vallen als hij boven een bepaalde energiegrens zou komen. Hiermee zou ik mijn eigen energie om kunnen zetten in punten, best slim vond ik zelf! Ik heb toen ook een energiegrens voor mijn eigen robot toegevoegd zodat mijn robot niet per ongeluk dood zou gaan doordat hij stopt met aanvallen.

Ik heb deze tactiek met verschillende variabele waardes en tegen een hoop robots getest, maar kwam er helaas achter dat deze tactiek mij alleen maar tegenwerkt. Ik kreeg zelfs minder punten dan eerst! Ik zou per tegenstander de waardes van de variabelen perfect moeten instellen wilde ik er iets uit halen en toen besloot ik dat dit niet de moeite waard was om aan door te werken.

Als leuke gimmic heb ik nog een “Taunt” task toegevoegd die de tegenstander met zijn eigen naam belachelijk maakt in de debugger.

Ondanks dat robocode hier niet helemaal de beste uitvoering voor was, vond ik het erg fijn om met een Behaviour Tree te werken. Ik kijk er naar uit om hier in de toekomst vaker mee te gaan werken.

Hieronder is mijn code voor het schieten van een kogel zodra mijn robot een andere robot raakt. Ik heb geprobeerd de rest van de code in dit bestand te zetten, maar het werd erg rommelig en onleesbaar. Als je de rest van de code wilt zien, kan dat op mijn github!

Github

```
public ShootOnRam(BlackBoard _blackBoard, double _power, double _minimumEnemyEnergy = 0)
{
    blackBoard = _blackBoard;
    power = _power;
    minimumEnemyEnergy = _minimumEnemyEnergy;
}
```

```
public override BTNodeStatus Tick()
{
    if(blackBoard.LastRammedRobotEvent != null)
    {
        if (blackBoard.LastRammedRobotEvent.Energy > minimumEnemyEnergy) { blackBoard.Robot.Fire(power); }
        blackBoard.LastRammedRobotEvent = null;
    }

    return BTNodeStatus.Success;
}
```