

# Game Development

## Kernmodule 3 AI

Mick Teunissen

Inhoud:

- Beschrijving Gameplay
- Beschrijving AI en implementatie
- Beschrijving gebruikte systeem
- Behaviour Tree
- UML
- PMI

### Gameplay

Shinobi striker is een top down actie game waarbij de speler als ninja de samurai uit een dorp moeten zien te verslaan. Je krijgt hulp van een mede ninja die rookbommen kan gooien om de speler een opening kan geven om te vluchten.

Het hoofddoel is om alle samurai in het drop uit te schakelen. Dit kan de speler doen door shurikens naar de enemy samurai te gooien tot hp bar op 0 eindigt en de samurai is verslagen. Als alle 6 de samurai zijn uitgeschakeld heeft de speler gewonnen. Hebben de samurai er in geslaagd de hp van de speler tot 0 te brengen heeft de speler verloren en is het spel afgelopen.

### AI en implementatie

Mijn Feedback van de vorige keer was dat ik geen Panda Behavior tree/plugin mocht gebruiker om mijn states in te maken. Dit heb ik bij deze dus aangepast. De hele plugin is weg en ik heb een behavior tree gemaakt met behulp van de starter project die op canvas stond aangegeven.

Ik heb 2 soorten AI in de game verwerkt. De enemy samurai en de allied ninja. Om alle verschillende states en behaviors te managen heb ik een Behavior Tree gebouwd met als inspiratie van het voorbeeldproject dat op canvas was aangegeven "BehaviorTreeExample". Ik heb gekeken wat elke functie deed en hoe de nodes waren opgebouwd en met elkaar communiceren. Toen heb ik het zelf toegepast aan mijn eigen project.

Ik moet wel eerlijk toegeven dat na lang testen en proberen het me niet is gelukt om de samurai te laten reageren op de smokebomb van de ally. Unity gaf een hoop Null reference errors en terwijl ik het duidelijk gedefinieerd had maargoed helaas werkte Unity niet mee en heb ik het na lang testen maar gelaten voor wat het is. De go back to patrol if in smoke functie van de enemy samurai er dus niet in verwerkt.

Ik heb een simpele game loop in de game toegevoegd met een menu, win en lose condition.

## Gebruikte systeem

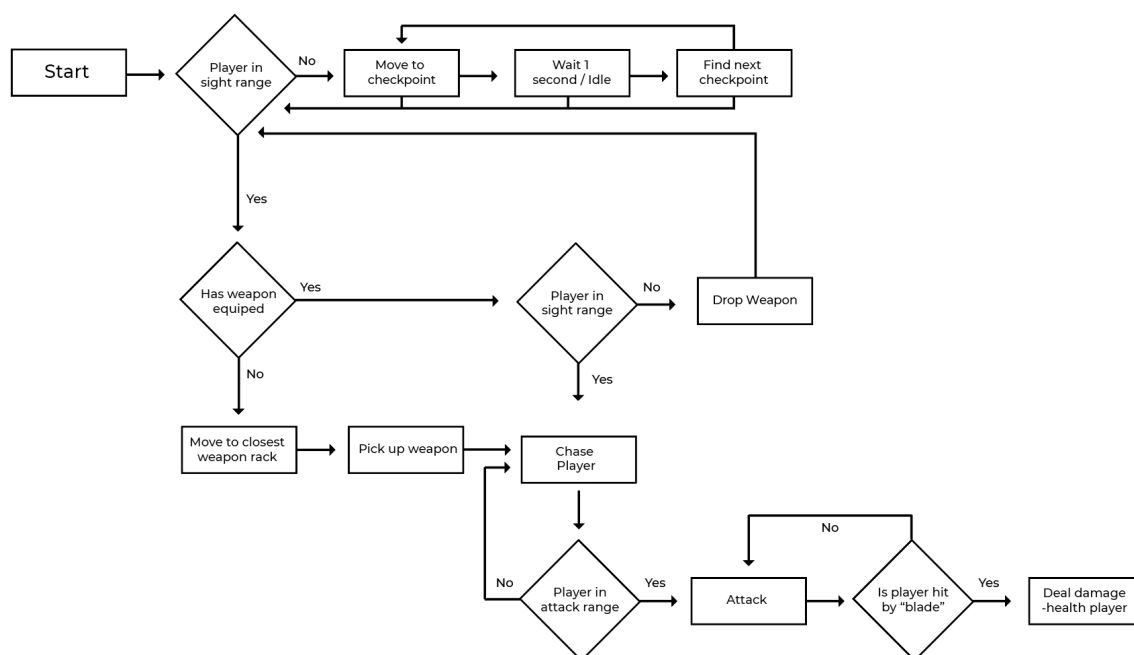
Voor het AI Pathfinding algoritme heb ik A\* pathfinding gebruikt. Ik heb de zelfde code hiervoor gebruikt als die in opdracht 2. Na de lessen terug te hebben gekeken en wat tutorials op internet te hebben gevolgd van Sebastian Lague vond ik A\* goed onder de knie te krijgen. Ik heb een aantal tutorials van Sebastian Lague letterlijk gevolgd om een begin te maken aan het A\* algoritme. Daarna heb ik de code met eigen behaviours er aan toe te voegen goed kunnen uitbreiden.

De Samurai en de Ninja hebben allebei een ander behaviour script met verschillende states. De codes lijken best op elkaar maar ik heb toch voor 2 aparte scripts gekozen om het overzicht in het project te bewaren. In de behaviour scripts staan onder andere de Health, Movement, Actie, Patrol functies beschreven. In de behavior tree maak ik gebruik van een Selector/Composite, Sequence, en Condition nodes. Deze zijn uiteraard nodig om alles goed te laten werken.

Voor de patrol functie van de samurai maak ik gebruik van een waypoint systeem zodat er consistente patrouille wordt aangehouden en dit de gameplay verbeterd in plaats van een bijvoorbeeld random position.

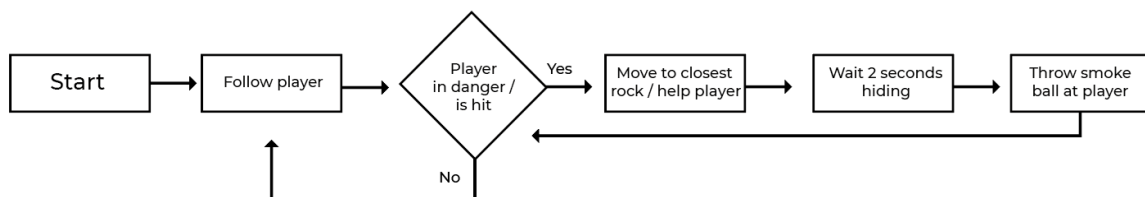
## Behaviour Tree

### Samurai/Enemy state diagram:



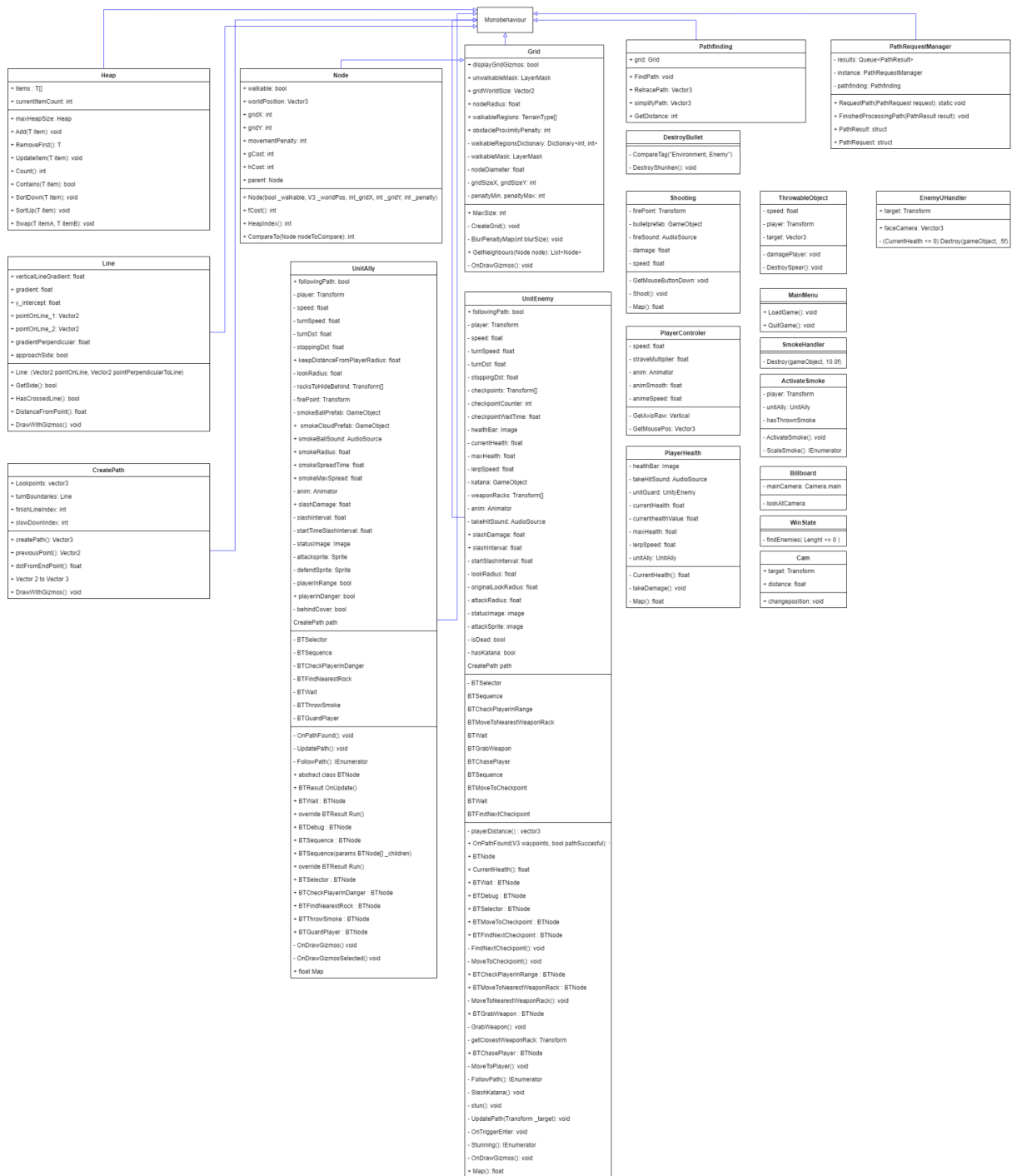
State	Purpose	Type of interaction
Patrol	Samurai patrols from waypoint to waypoint	world and AI
Grab weapon	Samurai goes to closest weapon rack when in sight of player and grabs weapon	World, player and AI
Follow player	Follows the player while in sightrange	Player and AI
Attack player	Attacks the player when in attack range	player and AI

### Ninja/Ally state diagram:



State	Purpose	Type of interaction
Follow player	Follows the player around on range	Player and AI
Find closest cover / rock	Ninja finds cover to closest rock	World and AI
Protect Player / smoke	After in range of cover throws smoke at the player to distract the samurai	Player and AI

# UML



## **PMI**

Voor en nadelen gebruik van een behavior tree die onder andere gebruik maken van en Selector/Composite, Sequence, en Condition Nodes.

### **Plus**

- De ai kan gemakkelijk en efficiënt de weg vinden met A\*
- Het is overzichtelijk, je kan zo snel alles states nagaan
- Je kan gemakkelijk alle states ordenen

### **Min**

- Met te veel states kan het onoverzichtelijk worden
- Soms clashen states met elkaar en wordt debuggen een stukje lastiger

### **Interessant**

- Het was interessant om zelf vanaf het begin een behavior tree op te bouwen. In plaats van een plugin te gebruiken.
- Ik heb veel gespeeld met de running, succes en failed states.
- Soms lopen bepaalde states in elkaar vast. Bij mij kwam het voor dat de samurai naar de weapon rack wou lopen. En het vervolgens uit de range van de player liep en daarna weer terug ging naar zijn patrol route waar ik als de player weer stond. Hierdoor ging de samurai weer terug naar de weapon rack. Het was dus een eindeloze loop waar ik een oplossing voor moest bedenken.