Implementatie Beschrijving

Niels Vriezen – AI Development Blok 3

Buddy

A* Pathfinding:

Voor dat ik aan het ontwerpen begon wist ik al dat ik zelf de pathfinding wilde schrijven. Dus ik heb totaal geen tutorials gebruikt en in eerste instantie ook geen psuedocode. Ik heb alleen het artikel die in Valentijn's presentatie stond gebruikt, dat het concept alleen uitlegde en meer niet.

Hiermee was ik al een heel eind gekomen en gebruikte mijn kennis van de A* Pathfinding die ik in Blok1 gebruikt heb om het te visualiseren. Ik gebruikte OnDrawGizmos om een grid te tekenen waarmee ik kon zien wat het veld is waar de AI kan lopen.

Nadat ik het systeem geschreven had, bleken er al snel wat zaken is te zijn. Om inzicht te krijgen in wat er mis ging, werd ik gedwongen om het te debuggen met breakpoints. Echter kwam ik er zelf nog steeds niet goed uit. Het duurde ongeveer 2 weken voordat het me duidelijk werd dat er iets mis was met de berekening van zowel de FScore als de omrekening van Gridposition naar Worldposition. Hierbij had ik wel de hulp nodig van Valentijn en Vincent.

Dit bleek wel een moment dat alles de goede kant op ging. Er waren nog steeds problemen waardoor er geen goed pad uit kwam. Dit heb ik weten op te lossen door kubussen te tekenen op de plekken van de gridnodes met OnDrawGizmos. Ook door bollen te spawnen op de plekken waar het pad zou moeten zijn. Hierdoor kon ik het probleem met de worldposities oplossen en werkte de pathfinding eindelijk.

Om een object de A* te laten gebruiken kan men simpelweg het pad opvragen met GetPath(beginPositie, doelPositie) en dan krijgt men een lijst van Vector3 terug. Dit bevat dan elke Node van het pad.

Uiteindelijk zijn er nog wel wat problemen. Ondanks dat de AI met de pathfinding altijd op de bestemming zal aankomen, gaat dit niet via het kortste pad. Dit is dus nog iets waar ik naar kan kijken. Ook is het systeem nog heel langzaam in het berekenen. Er is een grote FPS drop en de gehele game staat daardoor voor bijna een seconde stil.

Behaviour Tree:

Voor de implementatie van de Buddy had ik besloten om GOAP te gebruiken. Echter nam A* pathfinding te veel tijd in. Dus ik moest keuzes maken en besloot toen om het met een Behaviour Tree te doen. Voor de Behaviour Tree zou ik BehaviourBricks gebruiken. Echter begreep ik het systeem niet volledig. Ik dacht toen beter af te zijn als ik zelf het Behaviour Tree systeem zou schrijven.

Dit heb ik dan ook gedaan. Ik ben begonnen met het opzoeken van documentatie over het Behaviour Tree systeem. Ik vond toen al snel wat pseudo code. Die heb ik gebruikt om Composite nodes te schrijven. Dit resulteerde in de Sequence en de Selector nodes. Deze gebruik ik ook veel in de Buddy AI.

Het zijn nu allemaal MonoBehaviours, omdat BaseNode, de base class van elke node, inherit van MonoBehaviour. Echter inheritten de composite nodes ook nog van de CompositeNode class. Dit om bepaalde variabelen standaard te maken in deze type nodes. Denk bijvoorbeeld aan runningNode, die de child node opslaat als deze state.running returnt.

De entry point van het systeem is de BehaviourAgent class. Eigenlijk is dit gewoon een RepeatNode, alleen heet het anders. Hier geeft men aan welke node als eerste aangeroepen moet worden. Dit moet echter wel een CompositeNode zijn. Echter zou ik niet in kunnen zien waarom iemand een andere type Node zou willen gebruiken op die plek.

Boss Fight

Behaviour Tree:

De Boss AI gebruikt alleen de Behaviour Tree en geen pathfinding. De beweging zit gebakken in de specifieke Nodes van de Boss.

De Behaviour Tree van de Boss is een stuk simpeler dan die van de Buddy. Eigenlijk gebruikt het alleen een Sequence Node met daaraan de specifieke Boss nodes zoals BossAttack of Dive. Echter moet er een mogelijkheid zijn om hier uit te breken zodra de speler de Boss raakt. Echter kan er een check in de states gezet worden waarin de Boss geraakt kan worden.

Nog een verbetering voor dit systeem zou zijn dat niet alle Nodes op het object geplaatst worden. Nu lijkt het een rommeltje. Het zou ideaal zijn als men alleen een behaviourAgent op het object hoeft te plaatsen of het zelfs via een eigen script zou kunnen aanroepen. Dan is het een stuk overzichtelijker in de inspector.