Rick Ruiten Kerndev3 Artificial Intelligence

Boid concept :
Het concept dat ik had voor de boids was om er een Shooter van te maken waarbij je de boids uit de lucht kon schieten vanaf een fixed positie. Om hier een extra draai aan te geven wilde ik er een sniper game van maken waarbij je moet inzoomen om de boids te kunnen zien. Mijn eerst concept voor de boids was om er een shooter van te maken waarbij je een projectile afschiet waar de boids voor zouden weggaan. Met dit ontwerp ben ik uiteindelijk toch niet verder gegaan omdat het me net iets te veel tijd leek gaan kosten gezien het aantal uur dat we in de boids zouden moeten stoppen.

Procedural Generation Concept :
Wat ik voornamelijk heb gedaan bij mijn ontwerp was het zelf bedenken van een algoritme. Na een beetje puzzelen is het uiteindelijk gelukt en ben ik ook blij met het resultaat. Om hier nog een game van te kunnen maken wilde ik de speler het idee geven alsof die opgesloten zit in een doolhof waar hij de weg niet van weet. Ondertussen wordt de speler opgejaagd door een X aantal spoken die door het doolhof heen bewegen. Ze kunnen wel door de muren heen bewegen, maar niet erdoorheen kijken. Deze keuze heb ik moeten maken omdat ik er al gauw achter kwam dat mijn eigen algoritme niet lekker samenwerkt met alle andere bestaande algoritmes. Hier ben ik te laat achter gekomen en zal ik, zoals al besproken met Valentijn, voortaan eerst op zoek gaan naar de bestaande algoritmes en daar eventueel een eigen versie op maken. De AI die in het level rond dwaalt heeft 3 verschillende states: Move, Wander en Attack. Nadat ik alles heb gemaakt kwam in me op om de states op te vragen door middel van een ander script, zodat ik het main script kan hergebruiken per andere class, maar omdat ik geen andere classes gebruik heb ik dit achterwege gelaten.

Gebruikte Algoritmes

Boids :
Voor bij de boids heb ik de pseudocode gebruikt die we in de les hebben gekregen. Er waren een aantal kleine dingen die niet helemaal werkte zoals ik zou willen, maar de boids zelf deden grotendeels wat ze zouden moeten doen. De kleine foutjes heb ik heel lang overheen gelezen waardoor ik allemaal extra uren erin heb gestopt die niet nodig zijn geweest.

Procedural :
Zoals eerder vermeld heb ik geprobeerd een eigen algoritme te maken waarbij je een lijst afgaat (X positie) en op elke plek in die lijst een nieuwe lijst toe voeg met alle Z posities. Op elke Z positie maak ik een object aan van het type “GridObject” wat een zelfgeschreven class is die alle waardes voor die positie bewaard. Hierdoor krijg je een grid van X * Z en kan je gemakkelijk het object opvragen uit de lijst. Nadat ik hier mee bezig ben geweest kwam Valentijn al gauw met een andere oplossing die ik zou kunnen proberen, maar omdat ik al redelijk ver in het algoritme zat heb ik dit niet geprobeerd.

Omdat ik graag een zo echt mogelijke stad wilde creëren heb ik opgezocht hoe de meeste straten gebouwd zijn. Hieruit kwam dat de meeste straten bestaan uit een X aantal horizontale of verticale huizen waarnaast een weg wordt gebouwd. Nu heb ik niet elke regel van het bouwen van een straat in het programma zitten maar in ieder geval wel de basics. Zo check ik een random waarde of het boven een bepaald getal is en bouw ik op basis daarvan een huis of een weg. Op dat moment wordt de functie aangeroepen om een straat te gaan bouwen. Hierin wordt een variabele bijgehouden van hoeveel huizen er al in de straat staan en op basis daarvan wordt er gekeken naar de kans om een nieuw huis te bouwen. Tevens hebben de huizen 50/50 kans om horizontaal of verticaal gebouwd te worden. Zodra er geen huis meer gebouwd wordt komt er standaard een weg naast de huizen te staan.

Als er ergens een huis gebouwd moet worden krijgt het object in de lijst op die positie een waarde mee. Bij een 1 wordt het een huis, 2 een weg, 3 water en een 4 was om te testen welke huizen er aan het einde verwijderd worden. Is het getal nog 0 dan betekent het dat dit object nog niet aan de beurt is geweest. Het programma loopt voornamelijk door For-loops om te zorgen dat elk object meegenomen wordt bij het berekenen van alle straten en wegen.

Om water te creëren heb ik een beetje moeten spelen met de juiste waardes. Zo kwam ik er al gauw achter dat water een voorkeur heeft om met de stroming mee te gaan en dat ze een enkele keer afwijken van de stroming waardoor je de bochten in je riviertje krijgt. Het water wordt als eerste gecreëerd zodat de wegen en de huizen hier rekening mee kunnen houden.

Om alle overbodige huizen te verwijderen (denk hierbij aan huizen die de wegen blokkeren) heb ik een algoritme geschreven dat bij elk stukje weg om zich heen kijkt of er te veel huizen om hem heen staan. Als dat het geval is dan geeft het alle objecten om hem heen een verhoging van 1 op een specifieke variabele. Zodra deze variabele 4 hoog is zal het object worden verwijderd. Als een object aan de zijkant van het grid zit dan krijgt het object standaard een verhoging van 1 per zijde waar die aan vast zit.

Naast mijn eigen gebruikte algoritmes ben ik gaan achterhalen hoe ik A* zou kunnen gebruiken met mijn algoritme, maar daar kwam ik na 2 - 3 uurtjes puzzelen niet uit en heb dit idee toen weggelegd. Nadat ik het idee had weggelegd bedacht ik me dat ik eventueel een NavMesh kon gebruiken om de enemies over het pad te laten bewegen, maar ik heb nergens kunnen vinden hoe ik de NavMesh kan laten genereren vanaf een bepaald punt. Nu is mij verteld dat de nieuwe Unity 5.6 dit wel ondersteund, maar ook dit heb ik niet geprobeerd.

Uiteindelijk heb ik de keuze gemaakt om de enemies als spook te laten bewegen die door de muren heen kunnen en op zoek gaan naar de speler. De enemies pakken in de Wander state een willekeurige X en Z positie waar ze zich naartoe gaan bewegen. Zodra ze dat punt bereikt hebben doen ze dit nog een keer totdat ze de speler hebben gevonden. Op het moment dat de speler in zicht is gaat het spookje achter de speler aan tot op het punt waar de speler niet meer te zien is. Komt het spookje te dichtbij dan gaat hij automatisch in de Attack state waarin de speler geraakt wordt en een strafpunt krijgt. Bij 5 strafpunten is het spel Game Over en kan je het spel opnieuw spelen met een compleet andere map.

Naast dat ik geen terrain editor wilde gebruiken heb ik toch even gekeken hoe Diamond Square werkt en hoe ik er een terrain mee zou kunnen generaten. Dit was me al vrij snel duidelijk, maar leek me niet geschikt voor het idee dat ik in mijn hoofd had.

Over de kernmodule zelf :

De kernmodule zelf vond ik echt super interessant. Juist door de passie waarmee er gesproken werd en de leuke en leerzame dingen die we in de lessen hebben gezien en gedaan besefte ik me wat je allemaal kan doen met AI. Tuurlijk is niet alles me duidelijk geworden hoe mega groot AI is, maar daarvoor valt er een studie te volgen in AI.
Het enige dat ik zou aanpassen bij de kernmodule voor een volgende keer is dat er iets meer aandacht in de les wordt besteed aan het toepassen van de verschillende soorten algoritmes. Verder echt top lessen !

Dingen die ik nog zou doen :

Ik zou zelf waarschijnlijk nog meer kleine projecten gaan maken waarbij ik de andere algoritmes ga bekijken en hoe ik ze kan gaan verbeteren. Door een slecht begin van mij aan het huiswerk en een verkeerde beginstap heb ik veel dingen niet kunnen doen die ik graag wel had willen leren dit blok. Mede door het enthousiasme van Valentijn ben ik geïnteresseerd geraakt in Artificial Intelligence en zal hier dus nog zeker mee door gaan.

De boids game zou ik niet veel meer aan toevoegen, omdat het meer een kleine oefening is geweest dan echt daadwerkelijk een enorm spelconcept. Het is leuk om het een keer te oefenen en te zien hoe het werkt, maar daar blijft het voor mij ook bij.

Bij de procedural Generation had ik graag nog maar regels aan mijn stad vastgesteld waardoor ik uiteindelijk een heel land zou gaan kunnen bouwen. Daarnaast had ik willen leren hoe ik een stad vanuit het centrum opbouw in plaats vanuit de buitenranden. Dit gebeurt tevens ook niet in het echte leven.
Wat algoritmes betreft had ik graag nog even naar de A* willen kijken en ook daar een klein project voor maken zodat ik door heb hoe dat algoritme werkt en daarna toepassen aan mijn eigen city generator zodat ik niet een spookje hoef te maken maar daadwerkelijk een vijand die op zoek naar je is.

Bronvermelding :

Boids pseudocode
http://www.kfish.org/boids/pseudocode.html