Procedural Generated Online Virtual Worlds

Content generation in online virtual worlds

Jos Hoebe, jhe310@few.vu.nl

Begeleider: ***Prof. dr. Anton Eliens***

Tweede lezer: ***Dr. R. Siebens***

Samenvatting

In dit onderzoek is gekeken naar het ontwikkelen van virtuele werelden. Ten eerste werd er geconstateerd dat de hoeveelheid 3D content in games enorm is en dat daarmee de productietijd en -kosten ook erg groot zijn. Vervolgens werd er een mogelijke oplossing gegeven voor dit probleem in de vorm van procedural generation. Er bestaan verschillende vormen van deze techniek. Toepassingen werden gevonden bij het genereren van bomen, textures en terrein. Maar ook game content en animaties kunnen worden gegenereerd met behulp hiervan.

De werking van deze technieken is gebaseerd op fractal systems uit de natuur. Dit zijn recursieve patronen waarvan de vorm wordt gegenereerd door kleine versies van deze vorm. Fractals kunnen worden beschreven met behulp van L-systemen. Dit zijn grammatica uit de logica die deze recursieve eigenschap kunnen afbeelden. Wanneer men deze grammatica toepast met behulp van instructies is het mogelijk een vorm te genereren met behulp van enkele instructies. Ook het gebruik van een schijnbaar willekeurig patroon kan gebruikt worden. Het genereert steeds hetzelfde patroon wanneer men dezelfde input geeft. Dit maakt het geschikt voor terrein generatie in multiplayer. Vervolgens werd er een plan opgesteld om met behulp van Perlin ruis terrein te genereren en deze in te kleuren met behulp van texture tiling. Dit plan werd geïmplementeerd in de Unity3D engine.

Voorwoord

De afgelopen jaren heb ik mij bezig gehouden met het maken van games. Hierbij heb ik persoonlijk de ervaring opgedaan van het lange ontwikkeltraject. Het grote probleem bij het ontwikkelen van games is het maken van de enorme hoeveelheid aan content. In dit onderzoek ga ik daarom uitzoeken welke elementen verantwoordelijk zijn hiervoor en zal kijken naar procedural generation als oplossing.

Ik ben zelf niet 100% overtuigd van de mogelijkheden van procedural generation. Vooral omdat het maken van o.a. Levels de nodige aandacht van level designers vragen om het spel leuk en uitdagend te maken. Ik weet niet of procedural generation deze dingen kan bieden. Bovendien weet ik niet of de resulaten hiervan wel realistisch zullen zijn omdat het nabootsen van effecten die in de echte wereld een rol spelen mogelijk een groot probleem kan zijn.

Samen met Mike Hergaarden werk ik al tijden aan online spellen[1]. Hiervoor hebben wij recentelijk de professionele licentie van de Unity3D engine gekocht. Voor dit project willen we hiermee een online virtuele wereld bouwen. Deze wereld bestaat uit eilanden en een natuur die op de aarde moet lijken. Het is de bedoeling dat de virutele wereld oneindig is je oneindig kan doorvaren. Daarnaast moet het in staat zijn om voor alle spelers dezelfde wereld weer te geven. Mike zal zich daarbij bezig houden met het netwerk gedeelte van het project. Ik zal mij richten op het genereren van de omgevingen in deze wereld. Hiervoor wil ik eerst onderzoeken welke mogelijkheden ik daarbij heb.

1 Website: http://www.verdun-online.com

Inhoudsopgave

Inleiding

"To develop games is a huge process, a couple of years ago it wasn't that huge. In the 80s and early 90s, four people could sit in a basement and create a game that sold a lot. Now, creating a game is about the same complexity as creating a large Hollywood movie." – Erling Ellingsen (Game Developer bij Funcom)

Al jaren is er een sterke groei van de game industrie[1]. De geproduceerde games zijn vaak complex en bieden grote werelden aan waarin men kan rondlopen. Dit heeft een aantal nadelige gevolgen voor de ontwikkeling van dergelijke spelen. Een recent voorbeeld van zo'n game is het spel Grand Theft Auto (GTA) IV[3] van Rockstar games. Dit spel speelt zich af in een stad gelijke New-York. De ontwikkeling die zo typerend is voor deze tijd van game ontwikkeling is de toenemende hoeveelheid content van een spel zoals GTA..

De enorme toename van de kracht van de cpu's en gpu's* heeft er voor gezorgd dat de mogelijkheden op grafisch gebied sterk zijn uitgebreid. Dit zien we terug in GTA IV. Deze 3D graphics zijn vandaag de dag de standaard voor het weergeven van 3D op de 'next-gen' platformen (Microsoft's XboX 360, PC en Sony's Playstation 3). Het maken van deze 3D content die geschikt is voor deze 'next-gen' platformen is zeer veel complexer en vereist voor een enorme wereld enorm veel werk van artiesten. Het ontwikkelen van een spel zoals GTAIV kost een grote hoeveelheid man uren en heeft daarmee een zeer hoge productie kosten.* Deze kosten zijn dus voor een groot deel te wijten aan de content.

Nog een goed voorbeelden is het recent verschenen "Age of Conan" van het Noorse Funcom[2]. Elk onderdeel van de wereld is met de hand door artiesten vormgegeven. Rond de 200 man heeft zich 4 jaar lang bezig gehouden met het bouwen van deze wereld. De kosten bedragen ongeveer 200 miljoen.

Er zijn echter alternatieven beschikbaar, om het probleem van de steeds maar groeiende en complexer wordende virtuele werelden aan te pakken. Een mogelijke aanpak kan zich op meerdere stadia in het ontwikkel proces voordoen.** Ten eerste kan dit gedaan woorden op het gebied van objecten, de creatie hiervan is tijdrovend en er zijn er veel van nodig om een wereld te vullen. Nog een mogelijke implementatie zou plaats kunnen vinden op de schaal van een level/wereld zelf, waarbij automatisch het landschap wordt vormgegeven. Ook op gebied van content en gameplay kan het handmatige werk gedeeltelijk uit handen gegeven worden. De huidige toepassingen en technieken zullen worden behandeld. Daarna zal een feitelijke implementatie worden gegeven van een selectie van deze voor het bouwen van een virtuele wereld.

Procedural Generation

Het principe van procedural generation zit in stappen (procedures) om iets te genereren. Dit staat in groot contrast met de methodes waarbij hetgeen dat moet worden gebruikt van te voren wordt gemaakt en statisch wordt opgeslagen. Hier worden de stappen opgeslagen die nodig zijn om tot het eindresultaat te komen.[5] Er zijn verschillende manieren om bepaalde element procedureel te genereren, ik zal eerst voorbeelden geven van het gebruik van procedural generated content in games van nu.

Waar wordt procedural generation in gebruikt?

Procedural generation heeft een aantal toepassingen in games vandaag de dag. Maar zijn nog lang niet de standaard bij het maken van games. Als we globaal de content lijst aanhouden die we ook in een virtual world zouden verwachten, dan zien we dat er al een paar goede technieken aanwezig zijn om deze procedureel te genereren.

Vegetatie: Speedtree

Speedtree[13] is een verzameling van software met als doel het genereren van bomen en vegetatie. Er kan gewerkt mee worden in de verschillende fasen van het game ontwikkeling proces. Het is namelijk een CAD (Computer Aided Design) programma waarmee bomen kunnen worden gegenereerd en als losse objecten in het level worden

* Wet van Moore

** Zie appendix "Game development" voor een overzicht van het ontwikkel process blz. 12

gestuurd. Het is ook een plug-in binnen de 3D software. Maar naast deze oplossingen is het een stuk software dat in de game zelf, de vegetatie kan genereren en weergeven. Speedtree maakt onder ander gebruik van fractal procedures om de bomen te genereren.

Procedural Textures

Ook het genereren van procedural textures[4] kan in meerdere fasen van het ontwikkelproces. Er zijn een aantal programma's beschikbaar die procedureel textures kunnen genereren. Deze textures kunnen dan op objecten of in het landschap worden verwerkt met de daarvoor beschikbare tools. De procedure om een bepaalde texture op te bouwen wordt gebruikt om deze te generen en ofwel op te slaan als asset voor een game, ofwel direct ingame gebruikt. Gezien de enorme grootte van ruwe texture assets in games is het directe gebruik van procedural textures een goeie oplossing. Hiermee vermindert de benodigde opslagcapaciteit enorm, en is bovendien daarom bijzonder geschikt voor grote online projecten waarbij men game data over het internet verstuurt.

Procedural animatie

Animation is, zoals we al gemerkt hadden, een groot aspect van het ontwikkelen van een game. Hierbij wordt gebruik gemaakt van handmatige animatie van characters of motion capture technologie. De characters worden echter in-game blootgesteld aan situaties die niet van te voren kunnen worden geanimeerd. Daarom zijn er technieken bedacht om de animatie procedureel te genereren tijdens het spelen. In het spel worden allerlei invloeden uitgeoefend op een character en die worden doorberekend om zo bewegingen aan te brengen. Bijvoorbeeld het lopen of scheef terrein, of het vallen van een rand. In feite niets anders dan het berekenen van de positie van het skelet op basis van wat er gebeurt in de omgeving, de acties van de speler en de positie van het skelet. Een voorbeeld hiervan is het Euphoria[10] systeem dat dit aspect van een spel voor rekening kan nemen. Euphoria legt de lat nog hoger door ook Artificial Intelligence toe te voegen waardoor characters op een natuurlijke manier reageren op de invloeden uit de virtuele wereld.

Content Generatie

Ook op gebied van feitelijke content die gebruikt word voor de gameplay is er spraken van procedurele generatie. Hierbij worden delen van de verhaallijn of de doelen van een verhaal gegenereerd op basis van de input van de speler en overige factoren.

Een goed voorbeeld is het recent verschenen Far Cry 2[12] van ubisoft. Het maakt gebruik van procedural generated content om de verhaallijn vorm te geven. Er worden dynamische gesprekken met NPC's gegenereerd waardoor de spelervaring en het verhaal zich ontwikkelen. Hierdoor zijn de mogelijke manier om het spel uit te spelen (van begin tot de conditie als het spel af is gelopen) een stuk groter dan bij de vooraf uitgestippelde scripts.

Terrain Generation

Ten slotte is er de mogelijkheid om terrein te genereren met behulp van procedural generation technieken. Terrein staat aan de basis van een level dus via terrein zou het dus mogelijk zijn om een heel level vorm te geven. Er zijn een aantal programma's op de markt die het generen van een landschap als taak hebben, met vaak zeer realistische uitkomsten als gevolg. Men berekent met behulp van formules de erosie over het landschap, om ze tot een uiteindelijk resultaat te komen. Daarbij is het mogelijk om de textures (ondergronden) van het landschap ook op het landschap te plaatsen. Bij het plaatsen van deze textures word rekening gehouden met de natuur en de berekende erosie. Terrein word meestal opgeslagen als een 2D texture waarop elke hoogtepunt van het terrein beschreven is als 1 pixel met waarde 0 tot 1 om de hoogte aan te geven. Deze kunnen dan vanuit de programma's naar een game engine worden geëxporteerd om daar gebruikt te worden. FarCry II gebruikt ook procedural generation om landschappen mee te generen in de editor, om vervolgens door de level designers verder te worden uitgewerkt. Een voorbeeld van een programma om terrein me te genereren is Grome[11]. Grome is software die in meerdere fasen van het game ontwikkel proces kan worden gebruikt. Er is een editor die een level designer in staat stelt van te voren een level te genereren, te bewerken en vervolgens in het spel te gebruiken. Daarnaast heeft grome de mogelijkheid te worden geïntegreerd in bestaande leveldesign programma's met behulp van de SDK. Grome heeft een zeer sterke set aan algoritmen om terrein te genereren en deze met behulp van erosie technieken

een realistische vorm te geven. Ook is het mogelijk om textures op een procedurele manier door het programma automatisch op het landschap te schilderen. Hierdoor ontstaat een natuurlijker effect dan wanneer het met de hand gedaan zou worden door een level designer.

De werking van procedural generation

Sommige procedural generation technieken zijn gebaseerd op een aantal principes uit de logica en algoritmen. Ik zal hier een overzicht van geven, zodat deze gebruikt kunnen worden bij het genereren van elementen van virtual world.

Fractale systemen

Fractal systems[6] zijn wiskundige patronen die natuurlijke geometrische vormen beschrijven. Het kenmerkende element van fractal systems is dat er terugkerend patroon in de vorm terug is te vinden. Fractale geometrische vormen zijn te beschrijven door middel van een recursief algoritme dat een bepaald patroon in kleinere schaal terug laat keren op een deel van vorm. Hierdoor ontstaat per recursieve iteratie een nieuwe laag van detail. Het is daardoor mogelijk procedureel een complexe vorm te genereren met een recursief algoritme. Dit maakt het extreem geschikt voor de natuurlijke terugkerende patronen die we zien in terreinen, bomen of natuurlijke textuur.

L – Systemen

L systemen[6] zijn grammatica die o.a. Fractale systemen kunnen beschrijven. Het principe schuilt ook hier in het recursieve karakter. Een grammatica bestaat uit letters en delen van woorden. Door letters te herschrijven als delen van een woord ontstaat een gedetailleerder woord. Doordat de woorden uit instructies bestaan is het mogelijk om bijvoorbeeld de fractale patronen in de productieregels van de grammatica te zetten. In deze productieregels staan vervolgens de verwijzing naar de productieregel zelf. Hierdoor ontstaat een recursieve relatie en is het mogelijk om het fractale systeem te implementeren.

Perlin ruis

Nog een techniek om een natuurlijke vorm van willekeurigheid na te bootsen is een vorm van ruis. Het zogenaamde Perlin ruis[8] is een pseudo-random algoritme om een willekeurig patroon te genereren. Dit patroon kan vervolgens gebruikt worden om textures een natuurlijke willekeurige uitstraling te geven, of in terrein de hoogte te genereren.

Perlin ruis bestaat uit een aantal stappen. De eerste stap is om in een bepaald bereik waarden te genereren op basis van een aangegeven waarde (parametrisch) (De input waarde wordt ook wel de “seed” genoemd). Vervolgens wordt er een functie berekend die door deze waarden heen loopt. Hierdoor is het mogelijk om voor elke waarde een uitkomst te berekenen met de functie (interpolatie). De laatste stap van het algoritme bestaat uit het combineren van verschillende functieresultaten tot een. Hierbij worden de textures over elkaar heen gelegd. Het idee is dat als je fijne ruis met grove ruis combineer, er een middenweg ontstaat die bruikbaarder is voor gebruik in textures en terrein. Het combineren van de verschillende lagen met verschillende seeds geeft steeds een ander resultaat, maar wanneer alle seeds hetzelfde zijn is het eindresultaat hetzelfde. Het combineren van de verschillende lagen gaat op basis van de amplitude en de frequentie. Het variëren hiervan noemt men de persistance. Met perlin ruis is het dus mogelijk om een schijnbaar willekeurig patroon te genereren die reproduceerbaar is. Dit maakt het uitstekend geschikt voor textures en terrein.

Voronoi Cellen

Een aanpak die wat meer richting de biologie nijgt zijn de zogenaamde vonoroi diagrammen[5]. Deze diagrammen doen sterk denken aan biologische cellen. Het principe achter het genereren van dergelijk diagram is eenvoudig. Eerst neemt men een aantal punten, vervolgens verdeelt men de ruimte waarin deze punten zich bevinden onder deze punten. Hier wordt telkens precies de middellijnen tussen deze punten als grens genomen. Dit levert een patroon met een cellenstructuur op. Het gegenereerde patroon is uitermate geschikt voor textures uit de natuur die op celstructuren zijn gebaseerd zoals lichaamshuid of boomstammen. Het kan echter ook worden gebruikt voor artificiële patronen zoals agrarisch landschap of een wegenstructuur.

Tiling (tegelen)

Ten slotte is er nog een techniek die al tijden in games wordt gebruikt, het zogenoemde tiling[5]. Hierbij neemt men een set van textures en overlay textures. Er wordt op basis van factoren die invloed hebben op het gebruik van de texture bepaalt welke tiles er worden gebruikt en welke overlay er wordt geplaatst. Hierdoor ontstaat er door enkel gebruik te maken van onderdelen een nieuwe texture (bijvoorbeeld voor de ondergrond van terrein), die schijnbaar uniek is en niet van te voren door level designers worden ontworpen en geplaatst.

Procedural Generation Design

We hadden eerder al bekeken hoe op de huidige manier gewerkt wordt aan een virtuele wereld. Nu is het tijd om vast te stellen welke element onze virtuele wereld gaat bevatten. We gaan kijken op elke elementen we de accenten leggen en een realistisch doel vormen voor dit project. Hier volgt de requirements engineering voor de virtuele wereld.

Content in de virtuele wereld

De virtual world die wij op het oog hebben speelt zich voornamelijk af in de natuur. Uiteraard word er ook de mogelijkheid open gelaten om een stedelijke omgevingen te maken. Ik zal mij in dit project dan ook voornamelijk richten op het genereren van een omgeving in de natuur. De virtuele wereld wordt opgesplitst in stukken (levels). Hierin kan de speler zich vrij bewegen.

Mijn focus zal daarom op een aantal zaken liggen: Terrein generatie, Textures op het terrein aanbrengen en objecten plaatsen op het terrein (vegetatie en artificiële objecten). Dit zal ik implementeren met behulp van procedural generation technieken.

Terrein Generatie

In de omgeving (level) van een virtual world kan een terrein gezien worden als een 3D plane waarbij met behulp van een 2D texture map de hoogte wordt gegenereerd. Deze texture map heeft zwart-wit waardes om de hoogte aan te geven (zwart = min, wit = max). Het genereren van deze 2D texture is dus genoeg om een stuk terrein te genereren. Terrein generatie voor onze virtuele wereld moet in staat zijn om verschillende soorten landschappen te genereren, en moet bovendien deze landschappen kunnen reproduceren.

De fractale eigenschap van terrein zorgt ervoor dat de techniek die het terrein moet genereren de natuurlijke willekeurigheid moet kunnen nabootsen. Het perlin ruis is daarvoor extreem geschikt vanwege het feit dat deze techniek de mogelijkheid geeft de ruis in 2D vorm te gieten. Bovendien is met bepaalde seeds (input) mogelijk telkens hetzelfde terrein te genereren.

Voor terrein generatie is het dus nodig om eerst de benodigde stappen voor het het perlin noise algoritme te implementeren. De eerste stap is het genereren van willekeurige getallen om deze vervolgens te gebruiken in een 2D formule die benadert wordt door een cosinus functie. Met deze functie is het mogelijk om een 2D texture te genereren. Deze texture word als input aan het gedeelte van het programma te geven dat verantwoordelijk is voor het daadwerkelijk berekenen van het terrein als object in het level. Verder is belangrijk dat de module die er voor zorgt dat het terrein wordt gegenereerd een interface heeft waarin de eigenschappen van een terrein als parameter kunnen worden meegegeven. Voor het terrein willen we in ieder geval in staat zijn sommige parameters te beïnvloeden. Ten eerste de maximale hoogte, ten tweede het hoogte verschil en ten slotte de heuvelig-heid (frequentie van hoogteverschillen). Met deze parameters zijn we in staat om verschillende soorten landschappen te genereren (bergen, woestijn, heuvel etc.)

Textures plaatsen

Textures op terrein is een iets ander verhaal. Textures op het terrein moeten de verschillende soorten ondergrond op het terrein weergeven. Deze ondergrond is afhankelijk van het type terrein, en van de hoogte gegevens van het terrein. Zo zal bij een woestijn een zanderige texture worden gebruikt als basis terwijl er bij een berglandschap eerder voor gras of steen zou worden gekozen.

De techniek die het meest in de buurt komt is gebruik maken van tiling. Hierdoor is het mogelijk om meerdere textures als overlay te gebruiken over andere textures om zo een combinatie en overgangen te creëren. Op die

manier kunnen vervolgens de bergen met steen en de plattere gedeelten met gras of sneeuw worden “beschilderd”.

Deze techniek moet worden gekoppeld aan het terrein genererende gedeelte. Door gebruik te maken van de informatie over het type terrein en de gegevens van de gegenereerde 2D hoogtekaart moeten keuzes worden gemaakt over het plaatsen van textures op bepaalde punten.

Implementatie in Unity3D

Eerst beschrijf ik unity3D als platform om de virtual world in te maken. Ik zal het ontwerp voor de game die ik samen met Mike bouw geven. Vervolgens geef ik mijn implementatie van het genereren van een virtual world in Unity3D.

Unity3d als platform

Voor het bouwen van de virtual world is een goede basis nodig. Hiervoor gebruiken wij de Unity3D engine. Deze engine is geschikt voor het maken van virtuele werelden en ondersteunt alle aspecten van de klassieke manier om deze te maken. Het heeft de mogelijkheid om terrein weer te geven met een 2D hoogtekaart, en hierop textures te schilderen. Unity3D ondersteunt javascript en C# als programmeertalen. Hiermee zal ik bovenstaande ontwerpen voor het genereren van terrein en de textures hierop implementeren.

Implementatie

Het implementeren van de terrein generatie bestaat uit het verbinden van de perlin ruis generatie met de terrein en texture functies van unity3D. Vervolgens zijn er een aantal variabelen die instelt kunnen worden en die invloed hebben op het genereren.

De perlin ruis bestaat uit een binnen unity beschikbare classe die perlin ruis genereert. Deze klasse heeft een functie die op basis van de persistance, aantal octaves en de hoogte functievariabelen instelt. Deze functie kan worden gebruikt voor het genereren van een noise map. De waarden die men als input aan deze noise functie meegeeft hangt af van het coördinaat op de hoogtekaart en van de meegeven seed. Deze seed telt als het ware op bij de functie input. Als het ware het startpunt van een functie, een verschuiving. De noise functie kan gezien worden als een continu 2D vlak berekent door een functie. De hoogtekaart is als het ware een stuk hiervan dat de grootte heeft van de resolutie van de hoogtekaart. Naast de seed input kan de hoeveelheid octaves worden ingesteld, dit regelt de hoeveelheid detail. In de praktijk is het zo dat hoe meer octaves, hoe meer noise textures met steeds lagere resolutie over elkaar heen worden gelegd. Hierdoor ontstaat een mix van hoog en laag detail. Dit kan worden gebruik om de heuvelig-heid te berekenen. Daarnaast kan de persistance worden ingevoerd, deze regelt de sterkte van de opeenvolgende octaves en hun invloed op het eindresultaat.

Door de hoogtekaart te doorlopen en voor elke x,y coördinaat met behulp van de ruis functie een hoogtewaarde te bepalen berekenen wij de complete hoogtekaart. In de implementatie is het de bedoeling eilanden te vormen waar de speler naar toe kan. Om er voor te zorgen dan de randen van de eilanden onder water liggen is er een overlay texture gemaakt met de resolutie van de hoogtekaart. Deze overlay heeft zwarte randen en word langzaam wit naar het midden. Als de hoogtewaarde van de gegenereerde kaart hoger (hoe hoger hoe witter) is dan word het gemiddelde genomen tussen de hoogtewaarde en de overlay waarde. Het water niveau ligt op de helft van de hoogte. Er dient bij het instellen van de terrein hoogte dus rekening gehouden te worden met het waterniveau om te voorkomen dat het eiland in zijn geheel onder water ligt.

Het genereren van textures is vrij eenvoudig als men zich enkel met op de hoogtekaart baseert. Ten eerste moet er gebruik worden gemaakt van een aantal basis textures. Omdat dit gedeelte niet ondersteunt word in de API zijn we gelimiteerd tot 4 textures voor het terrein. Steen, zand, gras en sneeuw. Het is nu makkelijk regels op te stellen en de hoogtekaart pixel voor pixel langs te lopen en zo de zogenaamde splatmap te maken. De splatmap is een 2d texture met daarop 4 kleuren (rood, groen, blauw en de alpha waarde) De waarde van een pixel bepaalt welke texture er ligt. Rood voor zand, blauw voor rots enzovoorts. We kunnen bepalen of ergens rots moet komen door te kijken naar de steilheid van het terrein op een bepaalt punt. Unity heeft hier een aantal functies voor en we bepalen dat er rots moet komen als een bepaalt punt een helling heeft van 55 graden. Overige regels zijn bijvoorbeeld: Sneeuw boven een absolute hoogtewaarde van 90% en zand onder de 45%, daartussen grasland. Het is duidelijk dat op dit vlak de mogelijkheden en regels erg groot zijn. Er kan veel worden uitgebreid en gebruikmakend van geografische wetten kunnen hier foto-realistische resultaten worden gegenereerd.

Evaluatie

Nu we meerdere aspecten van procedural generation als oplossing voor content in games hebben onderzocht zijn we in staat om hierover uitspraken te doen. We benaderen het van meerdere kanten die onderzocht zijn.

Inzetbaarheid

We hebben gezien dat het game development proces zich in meerdere fasen voltrekt. Er bestaan dus meerdere mogelijkheden om procedural technieken te gebruiken. Tijdens het onderzoeken zijn we toolset zoals Grome en Speedtree tegengekomen. Beide zijn inzetbaar zowel ingame als oplossing voor het realtime genereren van content al ook in de ontwikkel-pipeline als externe software voor het maken van bomen of terrein. De enige vorm van procedural generation die enige vorm van standaard heeft weten te verwerven is op gebied van animation. Het Euphoria systeem of soortgelijke software zijn bijvoorbeeld al in GTAIV en andere AAA titels[1] ingebouwd. Hiermee is echter nog maar een deel van het content probleem opgelost en er zal nog veel moeten gebeuren tot ook texture en terrein generatie een zelfde verspreiding zal genieten.

Daarnaast kunnen we vraagtekens zetten bij de feitelijke resultaten van content generation. Bij de procedural storytelling die in Far Cry 2 zit verwerkt wordt bijvoorbeeld aangemerkt dat de verhalen en missies eentonig worden. Bovendien bestaat het gevaar dat men als het ware door heeft dat de verhalen of content in het algemeen automatisch word gegenereerd wat als speler het speelplezier kan verminderen. Men heeft als het ware de truc achter de verhalen en content door en dit geeft het gevoel doelloos bezig te zijn.

Resultaat van implementatie

De implementatie van procedural generation in unity3D is prima mogelijk. De principes achter de technieken zijn niet lastig. Het grootste probleem van de implementatie was het gebruik van de niet gedocumenteerde functies die texture en terrein bewerkingen deden. Hierdoor is het maken van de functies een uitzoekwerk. Unity bleek gelukkig wel een ingebouwde perlin noise classe te hebben die ik kon gebruiken, dus is het niet nodig om zelf deze te programmeren.

De implementatie heeft geen object plaatsing en algoritmen om bomen te plaatsen op het terrein. Dit heeft als grootste oorzaak dat normaliter men de functies om deze te plaatsen en bewerken alleen gebruikt in de editor en niet ingame. Deze functies zijn dus ook niet gedocumenteerd en pogingen om deze te gebruiken stranden bij gebrek aan tijd. Een ander aspect wat mogelijk nog voor verbetering vatbaar is, is het gebruik van erosie technieken waarbij het gegenereerde terrein als het ware een post processing ondergaat. Hierin worden de geografische berekening uitgevoerd op de hoogtekaart om zo de natuurlijke erosie te simuleren.

De implementatie geeft overtuigend weer dat procedural generation goed inzetbaar is bij het maken van levels voor een virtuele wereld. Het gemak waarbij men de ruis kan gebruiken voor het terrein en deze schijnbare willekeurigheid verder met textures en andere terrein gerelateerde effecten uitbreid is zeer overtuigend.

Toekomstbeeld

In de toekomst gaan we zeker meer zien van procedural generation in games. Gezien de huidige toepassing van procedural animation zal dit ontwikkelaars zeker overtuigen van de kracht hiervan. Er bestaat ook een duidelijke noodzaak waar jaren van ontwikkelen met een groot team nodig is om een grote virtuele wereld op te bouwen. Dit is met procedural technieken niet nodig en daarom is het voor de grote gamestudios een goede optie om te investeren in het ontwikkelen van deze technologieën. Er is immers als een grote basis aan informatie beschikbaar over de principes achter de technieken. Bovendien blijkt uit de implementatie in unity3D dat het feitelijk bouwen van een systeem dat bijvoorbeeld landschappen maakt helemaal niet lastig is. Met een klein budget en een team is het al mogelijk om de basis voor een spel te leggen.

1 Bekend voorbeeld is Force unleashed: http://www.lucasarts.com/games/theforceunleashed/

Referenties/Bronnen

1]Global Entertainment and media practice, video games industry preview, Pricewaterhousecoopers

http://www.pwc.com/extweb/industry.nsf/docid/8CF0A9E084894A5A85256CE8006E19ED#video

Een korte impressie van de groei van de game indrustrie, vooral in asia is er op het gebied van massive online veel groei.

2] Age of conan development, Age of Conan development team

http://aoc.wikia.com/wiki/Game_Development_Translation

Ontwikkelaars van het MMO spel Age of Conan hebben hier een overzicht gemaakt van de ontwikkeling van het spel. De enorme ontwikkel tijd is een prominent onderwerp.

[3] GTA IV: Building a brave new world, Hilary Goldstein

http://xbox360.ign.com/articles/863/863028p1.html

Interview met ontwikkelaar van GTA IV. Het detail van de stad waaraan men heeft gewerkt blijkt uit dit interview erg groot te zijn. Uit het interview blijkt precies het probleem van de content.

4] Procedural Textures, Ryan Garside

http://www.bit-tech.net/gaming/2006/11/09/Procedural_Textures_Future_Gam/1

Al een tijdje oud, maar dit interview laat heel goed zien wat de kracht van procedural texturing is. Het introduceert ook een aantal voorbeelden van spellen die hiermee werken.

5] A survey of procedural techniques for city generation, George Kelly, Hugh Mccabe, institute of technology, Dublin

http://www.gamesitb.com/SurveyProcedural.pdf

Dit artikel geeft een zeer goed overzicht van alle procedural generation technieken. Daarnaast geeft het een implementatie voorbeeld van hoe een stad procedureel gegenereerd kan worden met behulp van de technieken.

6] An introduction to fractals, Paul burke, University of western australia

http://local.wasp.uwa.edu.au/~pbourke/fractals/fracintro/

Een van de vele artikelen van Burke over procedural generation. Hier behandelt hij fractals. De oorsprong van fractals en de toepassing met l-systems komen aan bod.

7] Declarative modelling of virtual worlds, Ruben M. Smelik, TU Delft

http://graphics.tudelft.nl/Game_Technology/Smelik

Delfts onderzoek naar procedural modeling van virtuele werelden. Hier word goed weergegeven uit welke elementen een virtuele wereld bestaat. Vooral het plaatsen van wegen is een intressant aspect van de implementatie.

8] Perlin Noise, Hugo Elias, Shadow Robot Company

http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

De complete werking van Perlin Noise uitgelegt. Elke stap van het opzetten van een functie om deze te genereren

en de achtergrondinformatie.

9] Standford Virtual Worlds Group, Standford University

http://vw.stanford.edu/

Groep binnen de standford univesiteit die ondzoek doet naar de virtuele werelden. Zij hebben onder andere de tree generator " dryad" gebouwd (http://dryad.stanford.edu/). De website bevat artiekelen over het genereren van bomen en netwerk aspecten van virtual worlds.

10] Euphoria

http://www.naturalmotion.com/euphoria.htm

Website van eurphoria. Bevat een demonstratie van de software die dynamische animaties verzorgt in games. Er wordt aangegeven dat de techniek een vervanging moet zijn voor de motion-capture en handmatige (keyframe) animaties.

11] Grome editor

http://www.quadsoftware.com/index.php?m=section&sec=product&subsec=editor

De website van de landschapseditor grome. Op deze website staan een aantal videos die het gebruik van de editor uitleggen. Daarnaast kan er op de website een gratis trial versie van het programma worden gedownload.

12] Far Cry II

http://www.gametrailers.com/player/38618.html

Een voorbeeld van de level editor van het spel Far Cry 2, stukken van het terrein kunnen procedureel gegenereerd worden als de speler het type landschap definieert (zoals bij de onderzoeker uit delft). De video geeft ook een mooie inpressie van de hoeveelheid objecten die nodig zijn om een level mee vorm te geven.

Website game: http://farcry.uk.ubi.com/

13] Speedtree

http://www.speedtree.com/

De officiele site van de speedtree software. De toepassingen van speedtree staan op de hoofdpagina. Er staan veel spellen in de lijst van toepassingen. Er is ook een library van type bomen die men met dit programma kan genereren.

Appendix

Een overzicht van het game development proces

Om een duidelijke beeld te krijgen van de oplossingen die bestaan voor de groeiende hoeveelheid benodigde content in games is het zaak eerst het game ontwikkelingsproces te beschrijven. Ik bekijk de productie van een virtual world en geef aan welke type content benodigd zijn. Hierbij neem ik de huidige manier van ontwikkelen. Ik gebruik hierbij de kennis opgedaan in eerdere practica multimedia* en game-engines/games zoals unity3D en Age of Conan[2].

Content in Virtuele werelden

Het ontwikkelen van een virtual world bevat enkele stappen. Ik zal proberen aan de hand van benodigde content een overzicht te geven van de stappen die worden gemaakt. Ik verdeel de content eerst in een aantal categorieën om vervolgens aan te geven hoe deze in het spel gebracht worden.

Een virtual world bestaat in de eerste plaats uit een omgeving (environment). Deze omgeving bestaat uit het landschap, en de artificiële objecten. Landschap bestaat doorgaans uit het terrein, ondergrond, de undergrowth (gras, struiken) , bossage en weergave van de lucht. Het artificiële gedeelte bestaat uit wegen, nederzettingen en losse objecten.

Er zijn nog meer aspecten van game ontwikkeling dan enkel de virtual world zelf. Naast de pure 'statische' omgeving heeft de wereld ook characters en spel content/regels. De karakters bestaan uit de speelbare (avatars) en de niet speelbare NPC's (non-playable characters). De NPCs vormen samen met de 'content' (de regels) de verhalen en zorgen voor de spelervaring.

Dit is de content zoals gerepresenteerd in het spel zelf. Hier gaat echter een ontwikkelingsproces aan vooraf, waar 3D/2D/Geluid artiesten aan werken. Het op een rij zetten van de ingrediënten voor een virtuele wereld maken het dan ook niet heel verassen dat een spel zoals Age of Conan jaren duurt voor het al deze content in het spel representeert.

Het ontwikkelproces

In het ontwikkelproces van games met virtual worlds maakt men gebruik van engine gerelateerde editors en third party editors. Deze laatste zien we terug bij het maken van objecten in het landschap (ondergrond texturen, struiken, bomen, losse objecten en gebouwen). Men maakt deze content in 3D programma's. Na het vormgeven importeert men deze in de engine om gebruikt te worden in de eigen editor voor het maken van de landschappen en vormgeven van de werelden.

Dergelijke editors hebben de optie om terrein vorm te geven met een verfkwast om zo de hoogteverschillen op het landschap te schilderen. Daarnaast is men in staat om de ondergrond een kleur te geven. De geïmporteerde content kan vervolgens op het terrein worden geplaatst om zo een game level te vormen. Op deze manier is door de level designers die deze editor bedienen de complete wereld van Age of Conan tot in elk detail gemaakt**. Er werken dus eerst artiesten in third party editors om de objecten te maken, om ze vervolgens aan de level editor te geven die de rest van de ontwikkeling van een level/wereld op zich neemt.

Eenzelfde productie ondergaan ook de characters, met toevoeging van animaties. Deze animaties worden in 3D programma's gedaan en/of met behulp van motion capture technieken*. Na het animeren worden de characters in het spel geïmporteerd.Naast de visuele content wordt er ook veel tijd besteed aan storytelling. De verhalen worden eerst verzonen en opgeschreven en vervolgens omgezet in bruikbare gameplay. Deze gameplay hangt meestal samen door middel van scripts. Aan deze scripts worden eventueel geluiden en overige content toegevoegd (zoals teksten).

Knelpunten

Het is duidelijk dat met de toename van de grootte van virtual worlds het werk om deze te maken ook sterk toeneemt. Bovengenoemde ontwikkelmethoden zijn nu misschien nog mogelijk met een stad ter grootte van New-York, of een beperktere versie van Hyboria in Age of Conan. Wat als we de hele Verenigde Staten als virtuele wereld willen hebben? Of in die lijn de hele wereld? We dienen wel te accepteren dat een werkelijke 1 op 1 relatie van de werkelijke wereld waarschijnlijk nooit mogelijk is vanwege de eindige complexiteit. Een accurate representatie is wel mogelijk, wanneer men gebruikt maakt van principes uit de natuur en menselijk handelen daar in. Op de manier doorwerken zoals boven beschreven is hiervoor geen optie, hiervoor is een andere aanpak vereist.

Het efficiënter maken van het creatie proces kan dan op verschillende vlakken worden aangepakt. We stellen ons doel bij en maken het minder ambitieus, we richten ons op het maken van een virtuele wereld met overeenkomstige verschijning zoals we die kennen van onze planeet.

We hebben dan de mogelijkheid om verschillende processen te automatiseren. Omdat we nog steeds de handmatige controle over het finale product willen hebben is het niet per se nodig om het level design werk te automatiseren. Mogelijke aanpak kan zich dus op meerdere stadia in het ontwikkel proces voordoen. Ten eerste kan dit gedaan woorden op het gebied van objecten, de creatie hiervan is tijdrovend en er zijn er veel van nodig om een wereld te vullen. Nog een mogelijke implementatie zou plaats kunnen vinden op de schaal van een level/wereld zelf, waarbij automatisch het landschap wordt vormgegeven. Ook op gebied van content en gameplay kan het handmatige werk gedeeltelijk uit handen gegeven worden.

Hiervoor bestaan een reed technieken, deze technieken vallen onder procedural generation. We nemen nu een kijkje naar de huidige toepassing van dergelijke technieken en bekijken deze onder de motorkap. Vervolgens kijken we naar hoe wij die in de generatie bij het genereren van onze virtual world kunnen toepassen.