2D Game

Het 2D spel gemaakt door **Nigel Belderbos** en **Simon Stijnen**.

Doelstelling

Het doel van dit spel is om de speler een uitdagende ervaring te bieden waarin zij zo lang mogelijk moeten overleven in een dynamisch gegenereerde wereld. Deze wereld wordt opgebouwd uit verschillende soorten terrein, zoals gras, zand en water. Elk type terrein heeft unieke eigenschappen die van invloed zijn op de manier waarop de speler zich door de wereld beweegt. Gras en zand zijn begaanbare oppervlakken waarover de speler kan lopen, terwijl water ontoegankelijk is en dient als obstakel. De uitdaging wordt verder verhoogd door de aanwezigheid van vijandige monsters die de speler aanvallen. Om te overleven, moet de speler deze monsters ontwijken of ze confronteren met behulp van wapens. Het spel eindigt wanneer de speler wordt verslagen.

Probleemstelling

Tijdens de ontwikkeling van dit project zijn we tegen enkele significante problemen aangelopen. De belangrijkste hiervan is dat we door tijdsgebrek geen implementatie voor de monsters hebben kunnen realiseren. Ons plan was om zombies te ontwerpen die gebruikmaken van het A*-algoritme om het kortste pad naar de speler te vinden. Hoewel het concept in theorie duidelijk was, hebben we het niet kunnen implementeren.

Een ander probleem betrof de werking van de camera. Door de manier waarop onze codebase is gestructureerd, hebben we onbedoeld problemen veroorzaakt die ervoor zorgen dat vijanden niet correct worden weergegeven in relatie tot de speler. In plaats daarvan worden ze rechtstreeks op de XY-coördinaten van het scherm getekend, onafhankelijk van de positie van de speler op de kaart. Deze fout beperkt de uitbreidbaarheid van ons project en maakt het moeilijker om nieuwe features te implementeren. We zijn van mening dat het aanpakken van dit camera-probleem prioriteit heeft, omdat het oplossen ervan de basis legt voor toekomstige uitbreidingen en verbeteringen.

Relevantie in de maatschappelijke context

Ons 2D-spel toont aan dat het mogelijk is om een procedureel gegenereerde wereld te maken met behulp van moderne algoritmen en tools. Procedurele generatie wordt tegenwoordig veel toegepast in de game-industrie, met als doel variatie, herspeelbaarheid en unieke ervaringen te bieden aan spelers. Bekende voorbeelden van spellen die gebruikmaken van vergelijkbare technieken zijn *Minecraft*, *No Man's Sky* en *Terraria*. Door dit concept op kleine schaal te implementeren, dragen we bij aan het begrip en de toepassing van procedurele generatie in een educatieve context.

Onze doelgroep is eenvoudig gekozen: de klasgenoten en docenten voor wie we het project presenteren. Dit zijn mensen die bekend zijn met de basisprincipes van programmeren en game-ontwikkeling, wat ons in staat stelde om ons te richten op de technische aspecten van het project in plaats van de toegankelijkheid ervan voor een breder publiek.

Analyse

Voor de wereldgeneratie hebben we gekozen voor procedurele generatie gebaseerd op Perlin noise. Perlin noise is een algoritme dat pseudo-willekeurige maar natuurlijke en realistische patronen genereert. Dit maakt het ideaal voor toepassingen zoals terrein- en landschapsvorming. In ons spel gebruiken we Perlin noise om de ondergrond te bepalen, bestaande uit gras, zand en water. Het algoritme analyseert een matrix van 3x3 rondom een specifieke positie en kijkt naar de verhoudingen tussen de omliggende typen terrein. Afhankelijk van de hoeveelheid water, gras en zand in de nabije omgeving, genereert het algoritme nieuwe terreinblokken die logisch aansluiten bij hun omgeving. Bijvoorbeeld, als een positie omringd is door veel water, is de kans groot dat er op die positie ook water wordt gegenereerd.

Dit proces zorgt voor een dynamische en organische wereld die elke keer anders is. Spelers kunnen dus telkens een unieke ervaring verwachten wanneer ze het spel spelen. Dit aspect van ons project was bijzonder interessant en uitdagend om te ontwikkelen, omdat kleine aanpassingen in het algoritme grote invloed kunnen hebben op de gegenereerde wereld.

Ons project is vergelijkbaar met andere spellen zoals de *Zelda*-kopie ontwikkeld door ClearCode en moderne overlevingsspellen zoals *Brotato* en *Survivor.io*. Hoewel we niet al deze spellen volledig konden evenaren qua complexiteit, hebben ze ons wel geïnspireerd in onze aanpak.

Voor de technische implementatie hebben we gebruikgemaakt van een combinatie van libraries en tools. We hebben *Pygame* en *Pygame-GUI* gebruikt voor respectievelijk de weergave van de game en de gebruikersinterface. Daarnaast hebben we gebruikgemaakt van *NumPy* om het Perlin noise-algoritme te ondersteunen en voor de eenvoud bij het werken met matrixbewerkingen. Het spel is compatibel met zowel Windows als Linux, en we hebben Python 3.12 gebruikt vanwege de LTS-ondersteuning voor beide platforms. Het project kan eenvoudig worden uitgevoerd door het in te laden in PyCharm en de hoofdmodule (*main*) uit te voeren.

Resultaat

Hoewel we enkele belangrijke doelstellingen hebben bereikt, zijn we er niet in geslaagd om het project volledig af te ronden. Het meest opvallende resultaat is het algoritme voor de procedurele generatie van de wereld. Dit algoritme werkt zoals bedoeld en genereert

telkens een unieke en geloofwaardige wereld met verschillende soorten terrein. Dit aspect van het spel hebben we goed kunnen finetunen, wat ons veel inzicht heeft gegeven in de werking van Perlin noise en hoe we het kunnen toepassen op andere projecten.

Echter, de andere kernelementen van het spel, zoals vijanden en een overlevingsmechanisme, ontbreken nog. De wereld is dus wel dynamisch en divers, maar biedt momenteel weinig gameplay. Dit maakt dat het spel meer aanvoelt als een proof of concept dan als een volledige game.

Uitbreiding

Wat kan je doen als uitbreiding:

- **Vijanden**: Vijanden zoeken het kortste pad naar de speler om deze zo snel mogelijk te verslaan. Deze maken dan ook gebruik van het *A* * algoritme.
- **Buit element**: Monsters laten bepaalde voorwerpen vallen die je kan gebruiken om je zwaard sterker te maken.
- **Kerkers**: De speler kan een kerker binnen gaan waar deze tegen moeilijkere vijanden moet vechten, maar hierdoor krijgt de speler ook een betere buit.
- **Timer**: Een timer die toont hoe lang de speler al overleeft.
- **Leaderboard**: Een scorebord met je beste scores.
- …

Aangezien het een ‘open spel’ is kun in veel richtingen gaan voor uitbreidingen.

Conclusie

We zijn niet in staat geweest om alle doelstellingen van ons project te realiseren. Dit is teleurstellend, vooral omdat we beiden erg gemotiveerd waren en hoge verwachtingen hadden. Toch hebben we waardevolle lessen geleerd en zijn we trots op de onderdelen die we wel hebben voltooid. Het algoritme voor de procedurele generatie van de wereld werkt naar behoren en biedt een solide basis voor toekomstige uitbreidingen. Het was interessant om te experimenteren met Perlin noise en de resultaten te zien van kleine aanpassingen aan het algoritme.

Onze grootste uitdaging was het omgaan met de beperkte tijd. Als we meer tijd hadden gehad, zouden we ons hebben kunnen richten op de implementatie van vijanden en gameplay-elementen. Desondanks hebben we een spel gemaakt dat functioneert en dat ons inzicht heeft gegeven in de uitdagingen en mogelijkheden van game-ontwikkeling.

We hopen dat dit project een goede basis biedt voor toekomstige verbeteringen en uitbreidingen. Het is een proof of concept dat laat zien hoe je met relatief eenvoudige tools en technieken een dynamisch en uniek spel kunt maken. Met verdere ontwikkeling kan dit project uitgroeien tot een volwaardig en uitdagend spel.