ZWERGEN CLASH ROAYL

Cluster Cry

Dokumentation

Marvin Binnemann	**5003938**
Lennart Buchtzik	**5009235**
Christian Schemoschek	**5012743**
Leon Hartmann	**5010766**

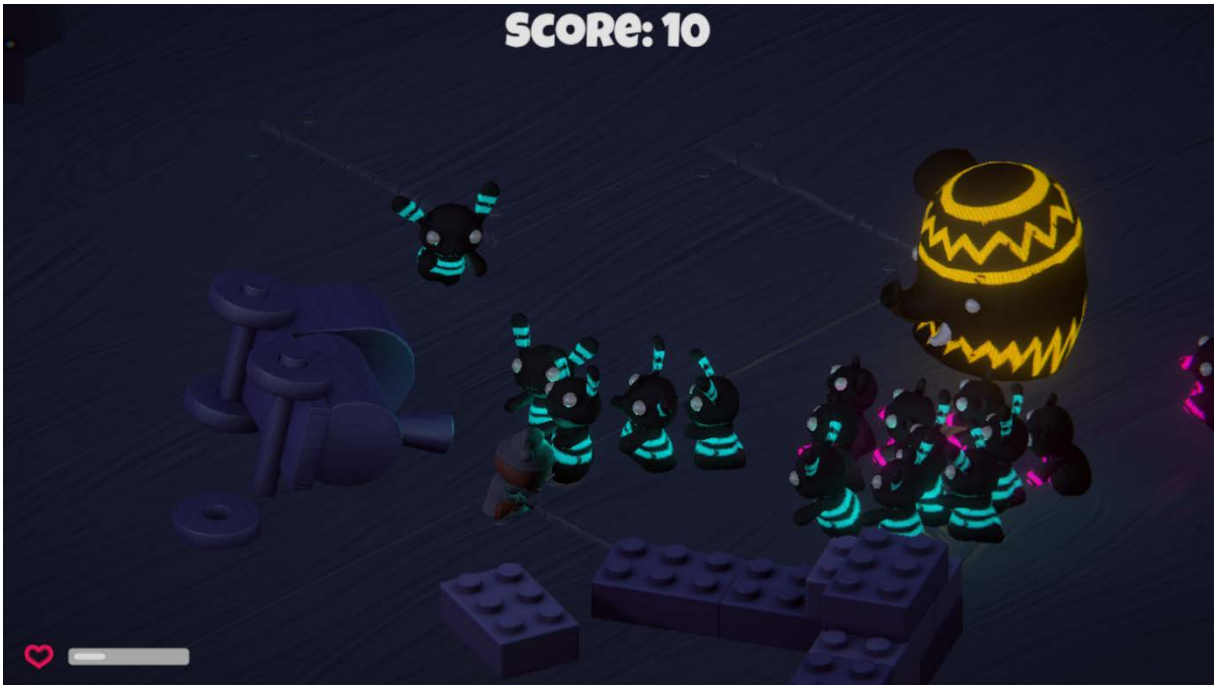
Zwergen Clash Royal

Für das Spiel „Zwergenclash Royal“ wurde die Unity Demo des Survivalshooters erweitert. In dem Survivalshooters schlüpft man in die Rolle einer kleinen Puppe im Schlafzimmer eines Kindes und muss sich gegen gehäkelte Feinde wie den Zombear und das Zombunny behaupten. Die Feinde unterscheiden sich in der Anzahl ihrer Lebenspunkte. Für den Kampf steht dem Spieler ein Lasergewehr zur Verfügung. Für jedes erschossene Monster gibt es Punkte, die den Highscore nach oben treiben. Das Spiel ist vorbei, wenn der Charakter des Spielers stirbt.

Die Idee war es das Setting des Survival Shooters zu ändern und die Spielwelt zu vergrößern. Der Graphikstil des Spiels wurde zunächst in einen „Voxel-Look“ geändert. Der vom Spieler gesteuerte Charakter ist nun ein Zwerg, welcher in einer Fantasywelt überleben muss. Die Fantasywelt ist dabei in kleinere von Felsen umzäunte Bereiche geteilt. Die Weltbereiche sind zufällig aus mehreren 15*15 Meter großen vorgefertigten Tiles generiert. Die Weltbereiche können durch das zwergische Tunnelsystem gewechselt werden, in dem der Spielercharakter die Zwergenmine betritt, die sich auf jedem dieser Weltbereiche befindet. Des Weiteren gibt es einen Tag- Nacht Rhythmus, der die Beleuchtung der Welt über die Zeit verändert. Der Zwerg kann sich mittels seiner Muskete, Granate oder Spitzhacke auf drei verschiedene Arten gegen seine Feinde behaupten, um zu überleben. Feinde sind zum Beispiel Gnome, welche in Dörfern spawnen und Skelette, die allerdings nur Nachts auftreten. Für die Muskete gibt es Munition und auch die Granaten sind in ihrer Anzahl beschränkt, so dass der Spieler sich bei Händlern in der Welt für Geld neue Munition oder Granaten kaufen muss. Des Weiteren kann der Spieler Heiltränke beim Händler erwerben. Geld bekommt er aus Truhen, welche sich in den Gnomdörfern befinden. Diese Truhen können auch Nahrung, Munition, Heiltränke oder Granaten enthalten.

Nahrung ist eine weitere Funktion in Zwergenclash Royal. Nahrung wird autmatisch nach gewisser Zeit konsumiert. Wenn dem Zwerg keine Nahrung zum Konusmieren zur Verfügung steht, nimmt er mit der Zeit Schaden.

Das Spiel ist beendet, wenn der Charakter stirbt.




Score: 40
13:00


Score: 0
20:00

Marvin Binnemann

Die ersten Änderungen des Spiels sollten einfach gehalten werden. Die erste Aufgabe bestand daraus, die Hauptwaffe auszutauschen. Dafür wurde das Waffen-Modell durch ein Musketenmodell0 ersetzt. Im Anschluss folgte der wichtigere Teil: Die Laserschüsse, welche durch einen Raycast mit gelber Farbe dargestellt waren, wurden durch Musketen-Kugeln ersetzt. Diese Kugeln sind eigenständige Objekte und werden am Ende der Muskete instanziiert. Diese bewegen sich dann in einer graden Flugbahn nach vorne, bis sie einen Collider treffen und anschließend gelöscht werden. Gehört der Collider einem Gegner, erleidet dieser schaden. Den Rauch, welcher aus der Muskete kommt, stammt vom „Zombunny“ und modifiziert, um ihn wiederzuverwenden. Das Licht von der alten Waffe blieb unverändert. Im Skript wurde dann die Feuerrate angepasst.

Als nächstes wurde die Granate in das Spiel eingebaut. Sie funktioniert ähnlich wie die Kugel der Muskete, nur dass sie ihren Punkt, wo sie in das Spiel instanziiert wird, über der Schulter des Spielers hat und von der Physik stärker nach unten gezogen wird als die Kugel. Kommt sie mit einem Collider in Berührung, wird ein Partikelsystem abgespielt und Schaden wird an allen Gegnern in einem bestimmten Umkreis verursacht.

Danach kam der Schadens-Rückstoß, welcher für den Spieler und die Gegner eingebaut wurde. Dieser stößt den Spieler oder das Objekt leicht zurück. Bei der Granate wird der Schadens-Rückstoß höher.

Um noch eine Nahkampfwaffe einzuführen, wurde ein Nahkampfskript eingebaut. Sobald der Spieler die Taste „F“ drückt, erleiden alle Gegner in seiner Reichweite vor ihm schaden.

Ein wichtiger weiterer Teil war es Spielsounds und Musik einzubauen. Dabei wurden verschiedene Töne auf freesound.org recherchiert und herausgesucht. Diese wurden dann auf unsere Bedürfnisse zugeschnitten, indem sie gekürzt und verändert wurden. Es wurden Sounds für den Spieler (Schaden, Nahkampfangriff, Schussgeräusche, Granatenexplosion und Todesschrei), den Gnom (Kichern, Schaden, Todesschrei) und den Hintergrund (zwitchernde Vögel im Wald) benötigt.

Dem wichtigstem Feind, dem Gnom, wurden Animationen herausgesucht. Animationen sind sehr aufwendig, weshalb die Animationsbibliothek von mixamo.com genutzt wurde. Hierbei wurden fünf verschiedene ausgesucht (Rumstehen, Angreifen, Rückstoß, Tod und Laufen) und dann ins Spiel integriert. Dafür wurde eine Stake-Maschine erstellt und mit Sinnvollen Übergängen sowie Auslösern eingebaut. Die Animationen mussten dann noch angepasst werden (zum Beispiel Übergänge zwischen den Animationen).

Gnom Dörfer beheimaten Gnome, diese müssen jedoch erst in die Welt instanziiert werden. Dazu wurde das „Enemy-Spawner“-Skript umgeschrieben und in das Dorf in ein Gameobjekt eingebaut. Jedes Dorf hat eine bestimmte Anzahl an Gnomen, die nach einer gewissen Zeit instanziiert werden. Zudem wurde das Bewegungsskript der Gnome so geändert, dass jedes Mal, wenn der Spieler in die Nähe des Gegners kommt, sie ihn automatisch verfolgen. Läuft der Spieler zu weit weg, drehen die Gnome um und laufen zu ihrem jeweiligen Gameobjekt, also ihrem Instanziierungs-Punkt, zurück. In jedem Dorf gibt es außerdem Beute für den Spieler zu holen. Sobald der Spieler die Kiste gefunden und mit der Taste „E“ eingesammelt hat, verschwindet die Kiste und der Spieler erhält eine zufällige Anzahl an Gegenständen.

Zuletzt wurde noch die Anzahl der Items begrenzt, die der Spieler hat. Kugeln, Granaten und Heiltränke, welche gleichzeitig mit eingebaut wurden, sind nun begrenzt vorhanden und können entweder durch das Einsammeln von Kisten oder das Kaufen von einem Händler nachgefüllt werden.

Lennart Buchtzik

Da niemand zu Beginn des Projekts mit Animationen genügend Erfahrung hatte um detaillierte Character, NPC oder Gegner Animationen zu erstellen wurde nach anderen Möglichkeiten gesucht diese in das Spiel zu integrieren.

Animationen

Bei dieser Recherche stieß man auf die Animationsbibliothek mixamo.com von Adobe. Damit die Animationen aus der Bibliothek ohne Fehler funktionieren konnten, mussten die meisten vorher selbsterstellten Modelle noch nachbearbeitet werden. Da in der Bibliothek ein „Autorigger“ genutzt wird, bei dem man die Gelenke im Modell platziert, mussten die Modelle im Voxel Stil möglichst klar abgegrenzte Gelenke haben. Sobald die gesuchten Animationen gefunden waren, mussten diese noch leicht angepasst werden bevor sie integriert werden konnten.

Dann wurde für jede Spielfigur ein zugehöriger Animator-Controller erstellt und Scripts geschrieben/bearbeitet für ihr jeweiliges Verhalten. Später wurden noch Anpassungen an der Position der Prefabs der Waffen während einzelner Animationen des Spielers gemacht.

Animiert wurden so das Player-Model, die Gegner Gnom und Skeleton und der NPC Trader.

Für jedes Modell wurden mehr oder weniger dieselben Basis-Animationen benötigt: Idle, Walk/Run, Fight, Damage-Knockback und Death. Beim Spieler kamen durch seine Waffen noch Shoot und Throw hinzu.

Damage Knockback

Beim Schadens-Rückstoß wird der Spieler oder das Objekt, welches Schaden nimmt, leicht zurückgestoßen. Bei Schaden durch die Granate ist der Rückstoß größer.

UserInterface/Gui

Im Userinterface bzw. über die GUI werden alle wichtigen Informationen gezeigt. Die stündliche Auskunft über die aktuelle Uhrzeit im Spiel wird durch verschiedene Symbole verdeutlicht, eine Sonne für den Tag, ein Mond für die Nacht und eine jeweilige Mischung bei Morgen und Abend.

Mithilfe weiterer Skripte wird die Anzahl der Inventargegenstände von Musketen- und Granatenmunition und von Heiltränken und Nahrung angezeigt, jeweils durch passende Bilder hinterlegt. Außerdem wird über ein Skript angezeigt wann dem Spieler die Fähigkeiten Musketenschuss und Granatenwurf wieder zur Verfügung stehen, nachdem sie verwendet wurden.

Außerdem wird in einer Leiste angezeigt wie viel Leben der Spieler noch zur Verfügung hat.

Christian Schemoschek

World Generator

Open World

Der Spieler kann sich frei in einer Endlos generierten Welt bewegen.

Biome, Tile und Border Prefab-System

Zur leichten Einbindung von neuen Landschaftstypen, Gebäuden, Pflanzen, NPCs und Gegnerspawns wurde ein System von Scripts geschaffen, welches eine Hierarchie an Prefabs aufbaut. Diese können zur Erweiterung der Spielwelt einfach bearbeitet und vervielfältigt werden.

Duplicate Restriction

Um zu verhindern, dass ein Areal der Spielwelt mehrmals TilePrefabs, wie den Spawn und den Minenausgang, in sich generiert, muss der Generator erkennen welche Prefabs nicht mehrfach verteilt werden dürfen.

Auch muss der Generator sichergehen, dass immer ein Spawn, ein Minenausgang, sowie eine Weltbegrenzung vorhanden sind.

Laden und Entladen von Gegnern / NPCs

Gegner und NPCs werden beim Spawnvorgang an ihre Spawns gekoppelt, damit sie beim Entladen der Welt mit entfernt werden.

Pathfinding mit einem dynamischen Nasvmesh

Ein zur Laufzeit generiertes Navmesh ermöglicht es zufällig generierte Welten zu erschaffen, in denen Gegner den Spieler überall verfolgen können.

Spawning und Teleportation / Travel

Beim Wechseln des Spielareals wird sichergestellt, dass der Spieler an die richtige Position befördert, alte Landschaften und Gegner entladen werden.

Inventory

Das Inventory hält die wichtigsten Items des Spielers an einem Ort für den Spieler, und andere Instanzen, wie z.B. das GUI, bereit.

Optimierung und Aufteilung anderer Scripts

Das Waffenscript wurde in Nahkampf, Granatenwurf und Schusswaffenscript aufgeteilt und unnötiger Programmcode wurde gestrichen.

Das Spawnscript wurde soweit erweitert, dass er Gegner in bestimmten Abständen erstellt, diese beim Entladen des Weltareals löscht und dem Gegner einen Ort zum Zurückkehren nach dem Verlieren des Spielers bietet.

Monster sind so angepasst, dass sie den Spieler verlieren und zu ihrem Spawn zurückkehren können.

Leon Hartmann

3D - Modelle
Da recht früh klar war, dass ein neuer Grafikstil für das Spiel verwendet werden sollte, jedoch niemand Erfahrungen im Bereich der 3D-Modellierung hatte, wurde ein detailarmer Voxelstil verwendet. Dafür mussten Modelle erstellt werden. Dazu zählen Spielercharakter, NPCs aber auch alle Modelle der Landschaft, Items und anderer Gegenstände, die es in der Welt zu entdecken gibt.

Tile - Prefabs
Wie bereits in der Einleitung erwähnt, werden die Weltabschnitte aus verschiedenen Tiles generiert. Ein Tile ist 15 x 15 Meter groß und beinhaltet verschiedene Landschaftselemente, Items aber auch Spawner oder NPCs. Für das Spiel wurden mehrere Tiles als Vorlagen erstellt. Sie alle sind so gestaltet, dass sie gut in einen Wald integriert werden können.

Tag – Nacht - Rythmus
Ein weiterer Aspekt des Spiels ist der Tag – Nacht – Rhythmus. Dieser wird durch eine sich langsam wechselnde Lichtintensität und eine rotierende Lichtquelle simuliert, sodass die Schatten mit der Zeit wandern und durch die Helligkeit klar zwischen Tag und Nacht unterschieden werden kann. Zudem wurde der Feind Spawner so modifiziert, dass Feinde nun – wenn gewünscht – nur Nachts spawnen. Zudem sterben nachtaktive NPCs, sobald es Tag wird. Im Spiel kann man dies an dem Skelett – NPC beobachten.

Handel
Interagiert der Spieler mit einem Händler, welcher zufällig in der Welt erscheinen kann (Tile Prefabs), wird das Spiel pausiert und ein Handelsmenü geöffnet. Hier kann der Spieler nun Ausrüstungsgegenstände gegen Geld kaufen. Das Menü besteht aus einfachen Buttons für die verfügbaren Items. Wird ein Button betätigt, wird das Geld vom Konto des Spielers abgezogen, solange der Spieler genug Geld zur Verfügung hat.

Userinterface
Das Userinterface gibt Auskünfte über die aktuelle Uhrzeit im Spiel. Ein Skript berechnet abhängig von der Lichtintensität die Uhrzeit und zeigt sie im Stundentakt an. Des Weiteren wird je nach Tageszeit ein anderes Symbol angezeigt. Der Tag wird durch ein Sonnensymbol angezeigt. Morgens und abends ist eine halbe Sonne zu sehen und Nachts der Mond.
Weitere Skripte zeigen die aktuelle Munitions-, Granaten-, Heiltrank- und Nahrungsanzahl an. Des Weiteren wird dem Spieler über ein Skript angezeigt, wann ihm seine Fähigkeiten (Musketenschuss oder Granatenwurf) wieder zur Verfügung stehen, nachdem er sie ausgelöst hat.

Player Slowdown
Ein weiteres Skript sorgt dafür, dass jede vom Spieler genutzte Waffe einen Slowdownwert besitzt. Dieser Wert beeinflusst die Laufgeschwindigkeit des Spielers, sobald er eine Waffe benutzt. Er wird also langsamer, wenn er die Muskete, Granate oder Spitzhacke benutzt.

Nahrung
Nahrung wird autmatisch nach einer bestimmten Zeit konsumiert. Wenn dem Zwerg keine Nahrung zum Konusmieren zur Verfügung steht, nimmt er mit der Zeit Schaden.