Projekt-Absolvent:	**Leif Gutowski (2375369)**
Leistungspunkte:	**15 CP**
Arbeitsaufwand:	**450 Stunden**
Projektziel**:**	**Konzept für Puzzlemechanik**

1 Einleitung

1.1 Projektbeschreibung

Im Rahmen dieses Projektes entstand eine lebhaftere Darstellung der Lichtform des Hauptcharakters aus "Light of Atlantis". Ziel war es, die starre Sprite-Grafik durch eine flüssige und generative Animation zu ersetzen. Dieses Projektziel wurde jedoch erst während des Projektverlaufs neu definiert.

Das Ursprüngliche Ziel war die Konzipierung von neuen Puzzle-Mechaniken für "Light of Atlantis" in Form von einem Puzzle-Boss und Puzzle-Räumen. Im Kern dieser Konzepte stand eine bereits etablierte neue Spielmechanik, mit der die Kontrolle zwischen verschiedenen Spielfiguren gewechselt werden kann.

In dieser Einleitung wird das Projektglossar mit allen wichtigen Projektspezifischen Begriffen sowie der ursprüngliche Projektzeitplan dargestellt. Außerdem wird das zugrundeliegende Spiel beschrieben und die Punkte erläutern, an denen dieses Projekt ansetzt.

1.2 Projektglossar

In diesem Abschnitt sind alle Projektspezifischen Begriffe aufgelistet.

Licht

Der Main Character in ein Lichtball - kurz das Licht - welcher umherfliegen und Roboter kontrollieren kann. Im weiteren Verlauf beziehe ich mich mit dem Begriff Licht immer auf diesen Lichtball.

Lichtshader

Der Lichtshader ist ein Shader für die Animation des Hauptcharakters, spezifisch entwickelt für das Licht und der Hauptfokus des Projektes.

Posess-Mechanik (Posess/Deposess)

Der Hauptcharakter – also das Licht – kann verschiedene Roboter kontrollieren, indem er sie besetzt. Intern wird dieser Prozess als Posess-Mechanik oder auch einfach als Posess und Deposess betitelt.

Roboter

Die Roboter in Light of Atlantis sind Spielfiguren, deren Kontrolle übernommen werden kann, um bestimmte Fähigkeiten wie Rumlaufen, Springen oder Angreifen zu erlangen.

Bubbles

Die Kugeln, die sich vom Licht durch den Lichtshader ablösen. Da intern von Bubbles und nicht von Blasen geredet wird, wurde der Begriff auch hier als Eigenname verwendet.

Core

Die Kugel, von denen sich die Bubbles lösen und die das Licht darstellt, ist der Core. Da intern vom Core und nicht vom Kern geredet wird, wurde der Begriff auch hier als Eigenname verwendet.

Lavalampen-Effekt

Die Darstellung, wie sich Bubbles vom Licht lösen. Es sieht aus wie Kugeln in einer Lavalampe, die in eine Richtung schwimmen.

1.3 Ursprünglicher Projektzeitplan

In Tabelle 1 ist der ursprüngliche Projektzeitplan dargestellt. Es sei anzumerken, dass die tatsächlichen Projektabschnitte sich ab einem gewissen Punkt stark von diesem Plan unterscheiden.

Abschnitt	Meilenstein	Schritt	Stunden
Einarbeitungsphase	Arbeitsumgebung	Einarbeitung	20
		Workflow-Verbesserungen	38
	Spielewelt	Einarbeitung	38
Konzeptumsetzung	Mini-Prototyp	Entwicklung	50
		Testing	12
	Erste Iterationen	Konzeptausarbeitung	104
		Testing (analog/digital)	12
	Zweite Iteration	Konzeptüberarbeitung	64
		Testing (analog/digital)	12
	Dritte Iteration	Konzeptüberarbeitung	38
		Testing (analog/ digital)	12
Abschlussphase	Finaler Prototyp	Konzeptfertigstellung	38
	Finales GDD im Wiki	GDD-Nachbereitung	12
			450

Tabelle 1: Ursprünglicher Projektzeitplan

1.4 Light of Atlantis

1.4.1 Beschreibung des Spiels

Als Basis für dieses Projekt diente das Spiel „Light of Atlantis“. Dabei handelt es sich um einen 2D-Puzzleplattformer mit Metroidvania-Struktur. Im Fokus stehen die Navigation durch die versunkene Stadt Atlantis und das Lösen von Rätseln mittels verschiedener Spielmechaniken, allen voran das Ein- und Ausschalten des Wassers in einem Raum, wodurch die Level zwischen plattformbasierter Navigation und Unterwassernavigation wechseln.

Ein weiterer großer Fokus ist das Wechseln der Kontrolle verschiedener Roboter mit verschiedenen Funktionen, indem sie besetzt werden (Posess-Mechanik). Dafür kann der Hauptcharakter als Licht im Raum umherfliegen und per Knopfdruck rumstehende Roboter übernehmen. Diese haben verschiedene Fähigkeiten, die zum Lösen der Rätsel benötigt werden.

1.4.2 Projektabgrenzung – Wo setzt das Projekt an

In diesem Abschnitt wird klargestellt, was zum Projekt gehört und was bereits als Basis vorhanden ist. Zu Beginn des Projektes ist die Posess-Mechanik noch besonders neu. Es ist bereits möglich, die Kontrolle zwischen verschiedenen Robotern zu wechseln. Für die technische Umsetzung gibt es Skripte, die die Steuerung des Prozesses ermöglichen (Deposess auf Knopfdruck, Steuerung des Lichts durch den Raum, Posess auf Knopfdruck). Die Darstellung des Lichts ist ein 2D-Sprite, welcher mit leichten Animationen versehen ist, die der Sprite-Grafik ein wenig die Statik nehmen.

An dieser Stelle setzt das Projekt an, indem zu Beginn die Möglichkeiten der Posess-Mechanik erforscht werden sollen. Wie bereits erwähnt, wechselt dieser Fokus im Laufe des Projekts von dem Potenzial der Posess-Mechanik zur Darstellung des Lichts, welche im Zentrum der Posess-Mechanik steht.

Das Projekt wurde allein durchgeführt, es gab jedoch regelmäßigen Austausch mit dem Team von Light of Atlantis in Form von Absprachen und wöchentlichen Meetings, sodass einige Entscheidungen gemeinsam getroffen wurden.

2. Projektbeginn

In diesem Abschnitt wird der Projektbeginn bis zur Entscheidung des Schwerpunktwechsels beschrieben.

2.1 Einlesen in aktuelles Projekt

Da zu Beginn des Projektes der aktuelle Stand des Spiels nicht bekannt war, brauchte es etwas Einarbeitungszeit, um die vorhandenen Spielmechaniken und die Spielwelt von Light of Atlantis zu verstehen. Dafür wurde das Projektwiki, welches vom Team in Notion gepflegt wird, tiefgehend einstudiert. Es wurde vor allem auch Fokus auf den Entwicklungsverlauf gelegt, um Entscheidungen hinter dem aktuellen Stand oder aktuellen Zielen nachvollziehen zu können. Die geplanten Workflow-Verbesserungen (siehe ursprünglicher Projektzeitplan Tabelle 1) wurden nach Beratung meinerseits vom Team übernommen.

2.2 Prototyp – Posess-Mechanik

Um das Potenzial der Posess-Mechanik zu erforschen, wurde sich dazu entschieden einen Prototypen zu entwickeln, der das Wechseln der Kontrolle verschiedener Spielcharaktere aus einer anderen Perspektive beleuchtet.

Mit dieser Intention entstand der Prototyp „Dungeon No Hero“, ein Top-Down-Dungeon Crawler, bei dem die Kontrolle zwischen verschiedenen Charakteren jederzeit gewechselt werden kann. Jeder Charakter kann zielen und schießen und basierend auf ihrer Farbe ist der zugefügte Schaden des Charakters höher oder niedriger (Schere-Stein-Papier-Prinzip).

- Rot fügt Grün mehr und Blau weniger Schaden zu
- Grün fügt Blau mehr und Rot weniger Schaden zu
- Blau fügt Rot mehr und Grün weniger Schaden zu

Nach einem Charakterwechsel, welcher per Knopfdruck passiert und automatisch den aktuell anvisierten Charakter auswählt, ist die Posess-Mechanik so lange blockiert, bis ein anderer Charakter ausgeschaltet wurde. Abbildung 1 zeigt einen Level aus dem Prototypen (siehe auch Anhang 1).

Wichtig zu erwähnen ist, dass das Wechseln zwischen den Charakteren unmittelbar passiert, und es nicht wie bei der Posess-Mechanik in „Light of Atlantis“ ein Licht gibt, welches gesteuert werden kann. Dies wurde bewusst gemacht, da es eine Intention dieses Prototypen war, zu schauen, ob ein schnelleres Gameplay für das Light of Atlantis in Frage kommen könnte, bei dem keine Figur in der Mitte benötigt wird, und stattdessen direkt zwischen Robotern gewechselt werden kann.

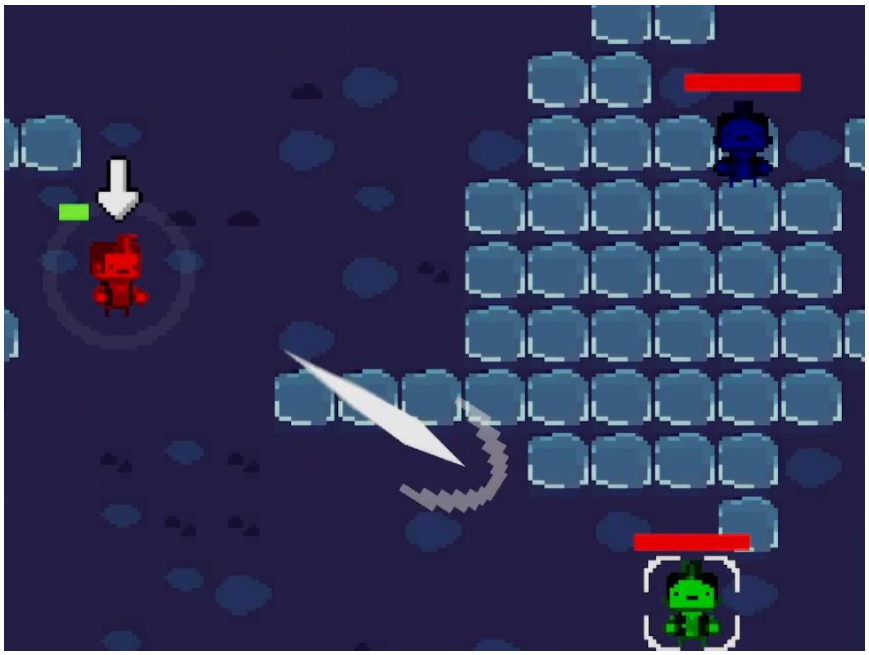

Abbildung 1: Dungeon No Hero: Der Pfeil zeigt auf den aktuell gesteuerten Charakter. Die Weiße Form, die auf den grünen Charakter zeigt, ist ein Angriffsprojektil. Das Weiße Zielkreuz um den grünen Charakter zeigt an, dass auf Knopfdruck in diesen Charakter gewechselt werden kann.

Das Gameplay ist sehr schnell und es erfordert viel Geschick, schnelle Wechsel durchzuführen. Interessanterweise kristallisierte sich heraus, dass bei dieser Art von Posess-Mechanik in Kombination mit dem Zielen und Schießen ein Time-Trial Modus interessant ist. So wird das schnelle Räumen von anderen Charakteren, bis nur noch ein Charakter übrig ist, zum Ziel des Spiels.

2.2.1 Erkenntnisse

Die größte Erkenntnis, die durch diesen Prototypen gewonnen werden konnte, ist, dass sich eine Posess-Mechanik zwar in verschiedenen Spielen gut umsetzen lässt, aber nicht zwangsläufig mit dem gleichen Tempo erfolgen muss. Während „Dungeon No Hero“ eher einem Bullethell-Titel ähnelt, welche in sich viel auf Geschicklichkeit fokussiert sind, ist das langsamere Wechseln mithilfe des Lichts in Light of Atlantis besser auf das Puzzle Genre zugeschnitten. Erste Testsessions im Januar 2024 mit dem Licht haben gezeigt, dass das Licht auf ganz natürliche Weise zum Umschauen benutzt wird. Dabei wird mit Ruhe und Überlegungen vorgegangen, sodass Entscheidungen, wie welcher andere Roboter oder der gleiche Roboter übernommen werden soll, erst im Laufe des Umherfliegens mit dem Licht getroffen werden.

Da „Light of Atlantis“ sich eher als ruhigeres Spiel identifiziert, wurde sich entschieden, das positive Feedback des langsameren und bedachten Wechselns zwischen den Roboter zu bekräftigen, anstatt sich weiter auf ein schnelleres Wechseln zu versteifen.

2.3 Schwerpunktwechsel

Nach Abschluss des Prototypen und nach der Festlegung, das Wechseln in der aktuellen Form weiterzuentwickeln, änderte sich der Fokus wieder auf das Licht. Dabei gingen wir auch auf das Feedback der Playtesting Sessions aus dem Januar 2024 ein.

Hierbei wurde angemerkt, dass sich auf der einen Seite die Steuerung des Lichts besonders gut anfühlt, allerdings auf der anderen Seite die visuelle Darstellung sehr starr ist und dem Licht an Leben fehlt. Die wenigen Animationen der 2D-Sprite-Grafik wurden teilweise gar nicht wahrgenommen, und erst als die Testenden darauf hingewiesen wurden, dass es leichte Bewegungen in der Grafik gibt, konnten diese wahrgenommen werden.

Dadurch wurde sich im Team dafür entschieden, diese nun im Fokus stehende Spielmechanik visuell aufzubessern, wodurch sich auch die nächsten Schritte in meinem Projekt änderten. Ziel war es nun, das Licht mittels technischer Lösungen visuell ansprechender zu gestalten.

3. Lichtshader

In folgendem Abschnitt wird die Entwicklung des Lichtshaders darstellt und die verschiedenen Funktionen erläutert.

3.1 Ausgangssituation & Konzept

Zum Zeitpunkt des Beginns dieser Veränderung war das Licht als 2D-Sprite-Grafik dargestellt (siehe Abbildung 2). Dieser Sprite wurde lediglich durch leichte Verzerrungen transformiert, um das Licht weicher erscheinen zu lassen. Außerdem ist der innere Punkt des Lichts mit einem Shader versehen, welcher wie eine Wasseroberfläche verschiedene Runde Formen durchläuft, um dem Licht Textur zu verschaffen.

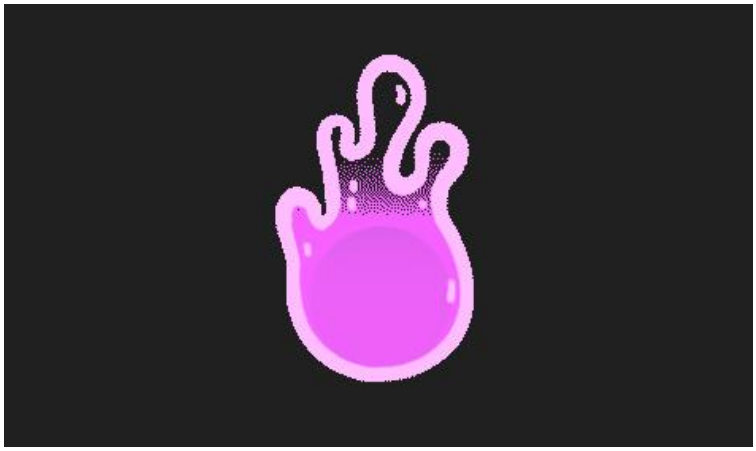

Abbildung 2: Alte Darstellung des Lichts (Ohne den Inneren Punkt).

An den weichen Rundungen des Sprites setzte das Konzept der Veränderung an. Anstatt einer festen Struktur sollten sich einzelne Kugeln (Bubbles) lösen. So sollte das Licht organischer wirken und viel an Statik des festen 2D-Sprites verlieren. Als Inspiration dienten in erster Linie Lava-Lampen und der Charakter Morph aus dem Disneyfilm „Der Schatzplanet“. Da sich der gewünschte Effekt an dem Hochfliegen der Kugeln in einer Lavalampe orientieren, wird dieser im Weiteren Lavalampen-Effekt genannt.

3.2 Basis Funktion

Für den Lavalampen-Effekt wurde eine Methode namens Metaballs verwendet. Metaballs sind eine Technik in der Computergrafik, bei der mehrere kugelförmige Objekte – die Metaballs - miteinander verschmelzen, wenn sie sich nahe genug kommen. Diese Verschmelzung erzeugt organische, flüssigkeitsähnliche Formen.

Das erste Teilziel bei der Verbesserung der Darstellung des Lichts war es, das Licht als ein Metaball zu definieren (Core) und eine feste Zahl von weiteren Metaballs von diesem abfliegen zu lassen (Bubbles).

Dafür wurde eine Shader programmiert, welcher den Core und die Bubbles als Metaballs mit festem Durchmesser definiert. Die Bubbles lösen sich mit einem festgelegten Tempo und springen zum Mittelpunkt des Cores zurück, wenn sie eine maximale Distanz erreicht haben. Jeder Metaball hat außerdem eine Kernfarbe und eine Outline-Farbe, die den vorhandenen Farben des Sprites entsprechen. Ergebnis war ein Lavalampen-Effekt, welcher die Bubbles ununterbrochen nach oben fliegen lässt. Abbildung 3 zeigt das Ergebnis.

Abbildung 3: Das Licht bestehend aus Metaballs, die durch den Lavalampen-Effekt nach oben fliegen (mit dem inneren Punkt wie beschrieben in Abschnitt 3.2 am Ursprung des Effekts).

3.3 Bewegung und Bewegungstempo

Für besseres Feedback wurden Anpassungen vorgenommen, welche die aktuelle Bewegungsrichtung und das aktuelle Bewegungstempo des Lichts berücksichtigen. Dafür wurden Parameter im Shader hinzugefügt und ein Skript geschrieben, welches das aktuelle

Bewegungstempo an den Shader weiterleitet. Die Bewegungsrichtung wurde vorerst durch eine Sprite-Rotation übernommen, wie es beim alten 2D-Sprite auch der Fall war. In der Bewegung sieht das Licht aus wie in Abbildung 3. Wenn das Licht stehen bleibt, sieht es aus wie in Abbildung 4 (siehe auch Anhang 2).

Abbildung 4: Das Licht, während es sich nicht bewegt (Idle). Während der Bewegung erzeugt das Licht einen Lavalampen-Effekt wie in Abbildung 2.

Ergebnis war mehr Feedback bei der Steuerung des Lichts. Beim Starten einer Bewegung beginnen sich die Bubbles zu lösen, beim Stoppen einer Bewegung ziehen sie sich wieder zum Mittelpunkt des Cores zurück. Schnellere Bewegungen ließen die Bubbles schneller fliegen, langsamere Bewegungen dahingegen langsamer.

3.4 Umbau auf skriptgesteuerte Animation

Um langfristig mehr Kontrolle über die visuelle Darstellung zu bekommen, wurde sich dazu entschieden, die Steuerung der Kugeln in ein C#-Skript auszulagern, und den Shader mit allen Informationen zu füttern, die er braucht, um die gleiche Darstellung zu erreichen, wie zuvor. Zu dem Parameter für das Bewegungstempo wurden somit weitere Parameter für die aktuellen Positionen der Bubbles und des Metaballs hinzugefügt, und die Logik, wie sich diese Kugeln bewegen, wurde in das C#-Skript übersetzt.

Mit der Bewegungslogik der Bubbles in einem Skript hatte man nun mehr Kontrolle über jede einzelne Kugel, und kann direkt im Skript die Variablen für die Bubbles ändern, wie zum Beispiel den Durchmesser. Da diese Skripte es auch ermöglichen, Variablen im Editor durch Kontrollwerkzeug wie einen Slider zu manipulieren, konnte nun während des Testens in der Unity-Engine die Größe der Kugeln angepasst werden.

3.5 Physische Animationen

3.5.1 Überschwung

Um noch mehr visuelles Feedback an die Spielenden zu vermitteln, wurde sich für ein Überschwung beim Stoppen des Lichts entschieden. Dafür wurde eine weiterer Metaball hinzugefügt, welcher beim Stoppen für einen kurzen Moment in die letzte Bewegungsrichtung ausschlägt und sich dann wieder zum Mittelpunkt des Cores zurückzieht.

Dieser Überschwung-Effekt lässt sich mittels zwei Parametern im Editor zur Laufzeit ändern. Es kann die maximal zurückgelegte Distanz des Überschwungs definiert werden, und die Zeit, in der der Überschwung abgeschlossen werden soll. Außerdem gibt es eine Animationskurve, die die Bewegung des Überschwungs im Detail darstellt. Diese kann angepasst werden, wenn z.B. auf dem Weg zurück zum Core der Überschwungs-Metaball langsamer sein soll als beim Überschwingen. Abbildung 5 zeigt den Moment des Überschwungs. Ergebnis ist ein erhöhtes Feedback beim Stoppen des Lichts.

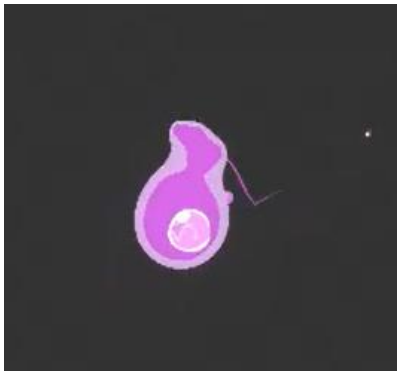

Abbildung 5: Überschwung Effekt. Der Überschwungs-Metaball schwingt im Moment des Stoppens über, sodass eine Seite größer erscheint (hier unten links)

3.5.2 Berührungen der Wand

Als große Herausforderung stellte sich das Anschmiegen des Lichts an naheliegende Wände heraus. Um das Feedback bei der Navigation des Lichts weiter zu verbessern, sollte das Licht beim Berühren der Wände oder anderer Objekte, mit denen es kollidieren kann, sich zusammendrücken und an die Wand anschmiegen. Um dies im Rahmen des vorhandenen Metaball-Systems umzusetzen, wurde sich dazu entschieden, einen Ring an Metaballs zu erstellen, welcher um den Core herum angeordnet ist (siehe Bild). Diese Squish-Metaballs bekommen einen Collider (Objekt, welches Kollisionen mit anderen Objekten in der Spielwelt überprüft), der den Metaball bei Berührung mit einer Wand in Richtung der Mitte des Cores bewegt, bis wiederum der eigene Collider des Cores übernimmt. So verformte sich der Ring der Squish-Metaballs bei Kontakt mit der Wand (siehe Abbildung 6 und Anhang 3).

Abbildung 6: Die Metaballs schmiegen sich an die Wand an, bis der Collider des Cores erreicht wird. Das Licht wird dadurch verformt.

Leider erwies sich die Umsetzung mithilfe von Metaballs als besonders knifflig, da eine hohe Menge an sehr kleinen Metaballs gebraucht wurde, um den Ring überhaupt als solchen erkennbar zu machen. Dazu kam, dass bei Kontakt mit den bestimmten Formen von Wänden, wie zum Beispiel einer Ecke, die Metaballs entweder in zwei Richtungen gedrückt wurde, und somit ihre Ringstruktur verloren, oder an einem Punkt in der Ringstruktur eine Lücke entstand, sodass ein visuelles Chaos an kleinen Kugeln zum Vorschein kam.

Nach langen Überlegungen wurde das Anschmiegen an Wände zumindest für den Rahmen dieses Projektes verworfen.

3.6 Definierte Animationspfade

Wie bereits im Abschnitt zum Umbau auf die skriptgesteuerte Animation beschrieben, bot dieser Umbau das Potenzial, einzelnen Kugeln klare Positionen zu geben. Dies sollte in Form von definierten Animationspfaden für individuelle Metaballs umgesetzt werden.

Mithilfe dieser Animationspfade könnten dann konkrete Animationen, wie das Schießen einer Metabubble in die Richtung, in die man gerade eine Interaktion tätigt, umgesetzt werden. Auch kleine Einspieler bei der Idle-Animation, die eine Monotone Animation unregelmäßig brechen soll, könnte mit definierten Pfaden für einige Kugeln umgesetzt werden. Als Inspiration hierfür diente die Idee von einem Licht, welches aus Langeweile ein paar Metaballs jongliert.

Es wurde ein Skript-Tutorial von Sebastian Langue[1], welches es ermöglicht, Pfade im Editor von Unity anzulegen und mithilfe von Joints die Kurvenverläufe des Pfades anzupassen, durchgearbeitet und für unsere eigenen Bedürfnisse angepasst.

Mithilfe dieses Skriptes war es nun möglich, einzelne Metaballs auf einen definierten Pfad zu schicken (siehe Abbildung 7). Diese Pfade konnten entweder als Rundkurs definiert werden, welcher die Kugeln immer im Kreis entlang des Pfades schickt, oder als Strecke zwischen Start und Ziel, sodass ein Metaball nach dem Abfliegen eines Pfades wieder auf den Start des Pfades zurückgesetzt wird.

[1] Siehe youtube.com, Videotitel: [Unity] 2D Curve Editor (E01: introduction and concepts). Link zu allen 6 Episoden: https://www.youtube.com/watch?v=RF04Fi9OCPc&list=PLFt_AvWsXl0d8aDaovNztYf6iTChHzrHP

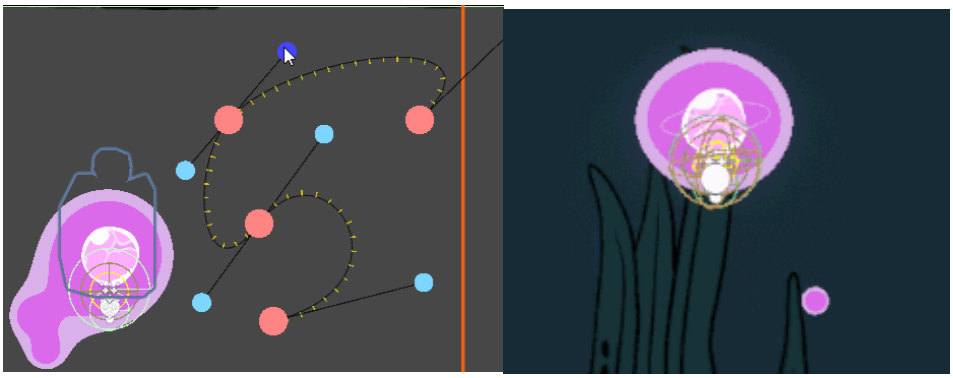

Abbildung 7: Im Editor kann mithilfe dieser Editor-Objekte ein Pfad gezeichnet werden (links). Rechts ist ein einzelner Metaball zu sehen, der über diesen Pfad gesteuert werden kann. In der Wiki-Fassung dieses Dokuments sind Animationen vorhanden.

Während die Lösung für definierte Animationen gut war, stellten sich auch hier ein paar Probleme heraus, wodurch das Weiterarbeiten an diesem Abschnitt im Rahmen dieses Projekt nicht sinnvoll war.

Die Pfade konnten nicht detailliert genug im Editor gezeichnet werden, sodass es zu Stockungen der Metaballs kam, die diese abflogen. Außerdem lag die Definition des Pfades im Editor. Damit lagen die Koordinaten jedes Punktes entlang des Pfades in Worldspace (Koordinaten beziehen sich auf das Zentrum der gesamten Spielwelt), die gewünschten Koordinaten des Pfades für die Metaballs liegen aber in nicht in der Editor-Welt, sondern basieren auf der Transparenten Grafik, auf die die Metaballs gezeichnet werden (Koordinaten beziehen sich auf das Zentrum der Grafik).

Darüber hinaus stellte sich heraus, dass sich nicht ausgiebig genug mit der Wirkungsweise von Metaballs auseinandergesetzt wurde, da das Hinzufügen von weiteren Metaballs die Größe anderer Metaballs direkt beeinflusste.

Da sich eine tiefere Auseinandersetzung mit Metaballs zu diesem Zeitpunkt im Rahmen des Projektes nicht gelohnt hat, besonders weil es noch keine richtigen Anwendungsfälle für die Animationspfade gegeben hat, wurde auch diese Änderung erstmal verworfen, mit der Option, sie zu einem späteren Zeitpunkt außerhalb dieses Projektrahmens bei Bedarf wieder zu aktivieren.

3.7 Parameter zur Anpassung

Bisher konnten die Metaballs nur in ihrer Größe angepasst werden. Mit dem gesamten Framework der Metaballs im Hintergrund war es nun an der Zeit, weitere Variablen einzufügen und im Editor mittel Slidern modifizierbar zu machen. So kann während der Laufzeit im Unity-Editor genau angepasst werden, wie die Bubbles aussehen sollen. Abbildung 8 zeigt die einstellbaren Parameter (siehe auch Anhang 4).

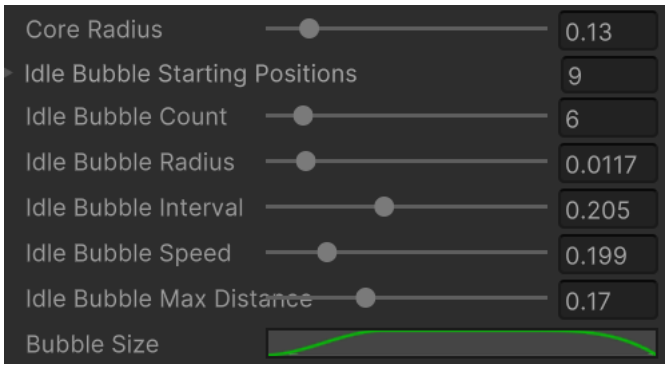

Abbildung 8: Die einstellbaren Parameter. Es gibt Core-Radius, Anzahl Bubbles, Bubble-Radius, das Interval (Wie lange wird gewartet, bis eine neue Bubble losfliegt), die Geschwindigkeit der Bubbles und ihre Maximale Reichweite, bevor sie verschwinden. Dazu gibt es außerdem eine Animationskurve für die Bubble-Größe, die auf ihrer X-Achse die Lebenszeit einer Bubble darstellt. So kann eine Bubble zu Beginn klein sein, bevor sie ihren eingestellten Radius erreicht und zum Ende wieder klein wird.

Diese Entscheidung hatte noch einen weiteren Hintergrund. Durch die Unterschätzung der Komplexität der Metaballs kam die Überlegung auf, sich in Zukunft eventuell wieder wegzubewegen von Metaballs und stattdessen auf Sprite-Animationen mit mehreren Frames zu wechseln. Mittels der verstellbaren Parameter kann eine breite Masse an möglichen visuellen Darstellungen des Lichts ausprobiert werden, um sich dann auf diese festzulegen und die als Basis für eine Spriteanimation zu nutzen.

3.8 Aufteilung in Bewegungsbubbles und Idle Bubbles

Zu diesem Zeitpunkt war das Licht in der Idle-Animation nur ein Blob, und während der Bewegung kam der Lavalampen Effekt zum Vorschein. Nach Austausch mit dem Team wurde entschieden, den Lavalampen-Effekt auch im Idle nach oben ausgerichtet darzustellen.

Diese Änderung legte nahe, auch unterschiedlich aussehende Lavalampen-Effekte während der Bewegung und während der Idle zu ermöglichen.

Dafür wurden die Parameter, die über Slider im Editor angepasst werden können, jeweils verdoppelt. Damit konnte eine komplett separate Darstellung des Effekts erzeugt werden, einerseits während der Bewegung, andererseits während sich das Licht nicht bewegt (siehe Anhang 5).

4 Resümee

4.1 Lessons Learned

Bei diesem Projekt haben sich einige Learnings herauskristallisiert. Allem voran, wie wertvoll regelmäßige Absprachen, Feedback und Testings sind. Ein iterativer Prozess über einen langen Zeitraum für einen kleinen Teilaspekt eines Spiels mag erst überwältigend klingen, jedoch im Angesicht der hohen Priorisierung, und da es sich bei dem Licht um ein

Spielelement handelt, mit dem die Spielenden eine große Zeit des Spiels in Kontakt treten, war der Prozess empfehlenswert.

Nichtsdestotrotz gab es auch große Schwierigkeiten, die sich aus einer Naivität entwickelt haben. Zu Beginn des Projektes wurde sich nicht tiefgründig genug mit Metaballs und ihren Wirkungsweisen beschäftigt. Formeln wurden bereits nach einem groben Verständnis verwendet. Dies führte langfristig dazu, dass es immer häufiger zu unerwartetem Verhalten der Metaballs kam, die dann durch Trial-and-Error gelöst wurden. Stattdessen hätte man mit einem besseren Verständnis das unerwartete Verhalten an der Ursache des Problems behandeln können.

Im Großen und Ganzen ist das Projekt mäßig zufriedenstellend, da viel Zeit in Aspekte geflossen ist, die aktuell keine Verwendung finden. Auch gab es sehr stressintensive Phasen, in denen nur sehr kleine Fortschritte unter hohem Druck erzielt wurden.

Die Shaderprogrammierung bzw. die Programmierung von visuellen Effekten war eine neue Herausforderung, die einige wertvolle Fähigkeiten gelehrt hat, aber auch nicht auf den eigenen Stärken aufgebaut hat.

4.2 Ausblick

Aktuell ist der Lavalampen-Effekt für den Moment mit fertig eingestellten Parametern intern in Benutzung. In einem ersten Schritt soll geschaut werden, inwiefern die Textur angepasst werden kann, damit es sich besser in den Rest der Grafik von Light of Atlantis einfügt. Abbildung 9 soll zeigen, was für eine Textur angestrebt wird.

Abbildung 9: Im Vergleich zu dem sehr sauberen Licht sind die Wände in Light of Atlantis rauer, mit unregelmäßigen Punkten versehen, die die großen Flächen aufbrechen. So etwas in der Art fehlt beim Licht noch.

Je nachdem, wie umsetzbar eine mögliche Lösung dafür ist, wird entschieden, ob die Metaballs weiterverwendet werden, oder ob sie stattdessen als Grundlage dafür dienen, Sprite-Animationen zu erstellen.

Die Zusammenarbeit mit dem Team wird langfristig weitergehen. Das nächste Projekt wird sich allerdings nicht mehr auf das Licht fokussieren, sondern einen Schwerpunkt auf die Erzählung und das Leveldesign legen, welches besser auf eigenen Stärken aufbaut.

4.3 Zeitaufschlüsselung

Abschnitt	**Meilenstein**	**Schritt**	**Stunden**
Einarbeitungsphase	Arbeitsumgebung	Einarbeitung	20
	Spielewelt	Einarbeitung	45
Umsetzung	Mini-Prototyp	Entwicklung	50
Ursprüngliches Konzept		Testing	8
Konzeptanpassung	Neues Projektziel	Konzeptausarbeitung	27
		Shader Einarbeitung	35
Umsetzung	Basic Functions	Entwicklung	70
Angepasstes Konzept	Bewegungstempo	Entwicklung	10
	Umbau auf Skriptsteuerung	Entwicklung	38
	Überschwung	Entwicklung	20
	Berührung der Wand	Entwicklung	42
	Definierte Animationspfade	Entwicklung	29
	Anpassbarkeit der Paramter	Entwicklung	24
	Aufteilung in Bewegung & Idle	Entwicklung	10
Abschlussphase	Anpassen der Parameter	Testing	12
	Finales Dokument im Wiki	Nachbereitung	12
			452

Tabelle 2: Tatsächliche Zeitaufschlüsselung