BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Fachbereich VI
Informatik und Medien

Bachelorarbeit

Drag'n Slay: Ein mobiles Multiplayer Game - Businessmodell, Grafik, Konzeption und Implementierung

Autor: Jan Rabe, jan.rabe@wooga.net, 766212
Jan Paschen, janp1988@gmail.com, 778472

Prüfer: Stephan Rehfeld

Gutachterin: Prof. Dr. Gudrun Görlitz

Abgabedatum: 01.07.2014

I Kurzfassung

In dieser Arbeit wird auf die Entwicklung eines Spielprototypen eingegangen, welcher auf einer Vielzahl von Mobilen Endgeräten laufen soll. Hierbei steht die Entwicklung der Netzwerkkommunikation und ein eigener Ressourcen sparender Grafikstil neben einer möglichen Verkaufsstrategie im Mittelpunkt.

Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Abkürzungsverzeichnis

MMO Massively Multiplayer Online Game
KI künstliche Intelligenz

DAW Digital Audio Workstation
VST Virtual Studio Technology
VSTi Virtual Studio Technology Instrument
SFX Sound Effects
CSR Classik Studio Reverb
EQ Equalizer

COLLADA COLLAborative Design Activity
FBX Filmbox
Saga endlos Geschichte

UI Design User Interface Design
dp density-independent pixel
ppi pixels per inch
GUI Graphical User Interface

1 Einleitung

Durch Drag'n Slay wird ein Multiplayer-Strategie-Spiel implementiert, dass in seiner Komplexität auf dem aktuellen Markt für mobile Endgeräte einzigartig ist. Der Spieler hat zahlreiche Möglichkeiten das Spiel-geschehen zu beeinflussen und damit zum Sieg zu führen. Gerade diese Komplexität sollte auf einem mobilen Endgerät wohl überlegt sein da Leistung und die Anzahl der Bedienelemente auf Grund der Größe des Endgerätes stark beschränkt sind. So wird im Laufe dieser Bachelorarbeit neben der Implementation immer wieder auf die Besonderheiten eingegangen, welche bei der Entwicklung für mobile Endgeräte wichtig sind.

Die Umsetzung erfolgt in Drag'n Slay durch Programmierung, sowie 2D und 3D Modellierung. Die folgende Arbeit wurde zu zweit verfasst, weil sich die Zusammenarbeit während der Gruppenarbeiten im Studium bewährt hat. Die unterschiedlichen Spezialisierungen beider Mitglieder in den Bereichen der Programmierung, Design, Sound- und 3D Modellierung ergänzen sich. Diese führen auch gleichzeitig zur Betrachtung aus zwei Blickwinkeln bei der Bearbeitung der Bachelorarbeit. Sie sind für Programmierer und Content Creator lesenswert. Um das Potential der Teammitglieder voll nutzen zu können, wird die ganze Arbeit unter dem Aspekt des Gamedesign stehen. Besonders motivierend für beide Teammitglieder ist das Zusammenführen ihrer Spezialisierungen.

Die während der Bachelorarbeit gewonnenen Erkenntnisse sollen zum Schluss zu einem funktionsfähigen Prototypen eines Multiplayer-Strategie-Spieles führen. Es sollte auch auf älteren mobilen Endgeräten, wie Smartphones und Tabletts, problemlos laufen und dabei trotzdem eine ansprechende Optik besitzen. Zusätzlich wollen wir uns auch mit den Vermarktungsmöglichkeiten für ein fertiges Produkt aufgezeigt.

2 Beschreibung der Idee

In der folgenden Arbeit wird die Entwicklung eines Multiplayer Spiel dargelegt. Dabei werden Gedanken zum Gesamt-Design, Hardware- und Softwareanforderungen abgewogen und ein mögliches Verkaufsmodel erstellt.

Die Kernelemente des Spieles sind Echtzeit-Multi-Spieler-Kämpfe gegen ebenbürtig erfahrener Spieler mit oder gegen Freunde. Der Spieler bewegt einzelne oder Schwärme von Flugzeugen von seinen Inseln zu den gegnerischen, um diese zu erobern.

Neben der endlos Geschichte (Saga) kann der Spieler im Metagame[1] Cloud Coins erwerben, der Soft-currancy, die im Shop gegen Unlockables eingetauscht werden können. Diese Funktionialen- und Kosmetischen Upgrades können frei kombiniert werden, um unterschiedliche Spielstile der Spieler zu verstärken.

3 Beschreibung vergleichbarer Spiele (JP)

Zum Momentanen Zeitpunkt (12.04.2014) gibt es im Google Play Store keine bei der Recherche gefundenen Spiele, welche einen vergleichbaren Spielumfang bei selber Idee bieten. Jedoch gibt es mehrere Spiele, welche auf der selben Idee basieren.

Um heraus zu finden in wie fern sich die hier noch auszuführende Idee von den vorhandenen Spielen unterscheiden, werden diese auf Inhalte untersucht, die bei der Entwicklung von Drag'n Slay besonders wichtig oder innovativ erscheinen.

Ziel des Spiels: Was ist der Spielinhalt? Welches Ziel muss erreicht werden um zu gewinnen? Was kann der Spieler tun um die Runde für sich zu entscheiden?

Spielmodi: Welche Modi werden im Spiel geboten? Ist es ein reines Singleplayer Spiel oder kann im Multiplayer mit mehreren Spielern gegeneinander angetreten werden?

Pay2Win: Ist es möglich über einen langen Zeitraum siegreich an dem Spiel teilzunehmen, ohne echtes Geld investieren zu müssen?

Grafikstiel: Welche Optik wird einem in dem Spiel geboten? Wird alles in einer simplen 2D Grafik präsentiert oder wird der Spieler in ein leuchtendes 3D Abenteuer entführt?

[1] `http://en.wikipedia.org/wiki/Metagaming` (übersetzt): (1)Metagaming ist jede Strategie, Aktion oder Methode in einem Spiel, das einen vorgegebenen Regelsatz überschreitet, externe Faktoren verwendet, um das Spiel zu beeinflussen, oder über die vermeintlichen Grenzen oder Umwelt hinausgeht, in dem das Spiel gesetzt wurde. (2) Das Spiel-Universum außerhalb des Spiels selber; die Verwendung von Informationen oder Ressourcen außerhalb des Spiels, Entscheidungen im Spiel zu beeinflussen.

Bedienung: Wie lässt sich das Spiel auf einem mobilen Endgerät steuern? Wie werden dem Spieler seine Interaktionen visualisiert.

Geschichte: Was für eine Hintergrundgeschichte bietet das Spiel? Leitet das Spiel durch eine lebendige Geschichte oder ist es ein reines Multiplayerspiel, welches nur auf Aktion ausgelegt ist.

Wird kein Spiel gefunden, dass in all diesen Punkten mit Drag'n Slay übereinstimmt, so kann davon ausgegangen werden, dass Drag'n Slay eine Chance hat sich auf dem Markt durchzusetzen, da es kein Identisches Konkurrenzprodukt gibt, welches dem Spieler dasselbe bietet.

3.1 Spawn Wars 2 (Android) (JP)

Dieses Spiel hat auf den ersten Blick ein sehr ähnliches Spielprinzip, bietet aber innerhalb des Spiels sehr viel weniger Möglichkeiten das Spielgeschehen zu beeinflussen und wird durch das damit statische und vorhersehbare Spielgeschehen schnell langweilig. Das Ziel des Spiels ist es möglichst schnell alle "Inseln" zu übernehmen. Übernommene Inseln spawnen neue Einheiten die im restlichen Spielverlauf helfen. Des weiteren ist es in höheren Leveln nicht mehr möglich ohne den Einsatz von Echtgeld zu bestehen. Außerdem bietet Spawn Wars2 keinen Multiplayer, und bei genauerem betrachten der KI fällt auf das diese nicht echt ist und innerhalb eines Level immer gleich agiert. Damit wird deutlich das Spawn Wars 2 weniger ein Strategiespiel als ein Geduldsspiel ist. Das Erreichen des Ziels ist nur eine Frage von richtigen Interaktionen zum richtigen Zeitpunkt und es lässt sich wiederholt mit der selben Abfolge gewinnen.

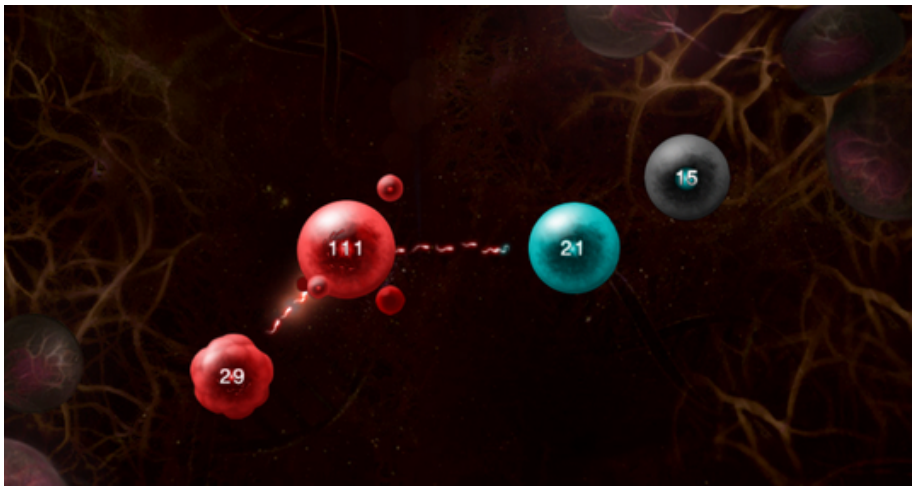


Abbildung 1: Spawn Wars 2

Zusätzlich ist Spawn Wars 2 komplett in 2D umgesetzt, Abbildung: 1 zeigt die

simple 2D Grafik. Wie in dem Bild zu Spawn Wars 2 zu sehen, steht die Geschichte in einem biologischem Kontext und zeigt den Kampf zwischen Zellen. Es finden sich aber zahlreiche Ähnlichkeiten welche die Touch-Steuerung auf mobilen Devices betrifft. So existiert auch ein Drag'n Drop System, um Einheiten von einem Punkt zum Nächsten zu senden.

3.2 Galcon Fusion (X-Plattform) (JP)

Galcon Fusion ist ein aktionreiches Echtzeitstrategiespiel bei dem es darum geht den Gegner zu besiegen, indem alle Planeten mit Raumschiffen übernommen werden. Startet man das Spiel sieht man eine Anzahl an Planeten, welche kontinuierlich Raumschiffe spawnen. Diese werden aber lediglich als Zahl auf dem jeweiligen Planeten dargestellt. Je nachdem welchem Spieler der Planet gehört, erscheint dieser in einer anderen Farbe. Um das Spiel zu gewinnen muss der Spieler die Schiffe von seinen eigenen Planeten so auf die gegnerischen Planeten verteilen, dass er diese einnimmt und dabei seine eigenen Planeten nicht in Gefahr bringt. Bei Galcon Fusion handelt es sich um ein reines Multiplayerspiel, bei dem man auf eine Internetverbindung angewiesen ist. Einen "in game shop" in dem man Spielerweiterungen oder Hilfsgegenstände käuflich erwerben kann, gibt es in Galcon Fusion nicht. Dafür hat das Spiel jedoch schon einen Grundpreis von 2,20€. Da das Spiel auf Massenschlachten abzielt ist die Grafik sehr simpel gehalten. Die Planeten erscheinen als Farbige 2D kreise. Die Raumschiffe werden als Zahlen auf dem Planeten visualisiert und werden nur während des Fluges als kleine farbige Dreiecke sichtbar (Abb.2).

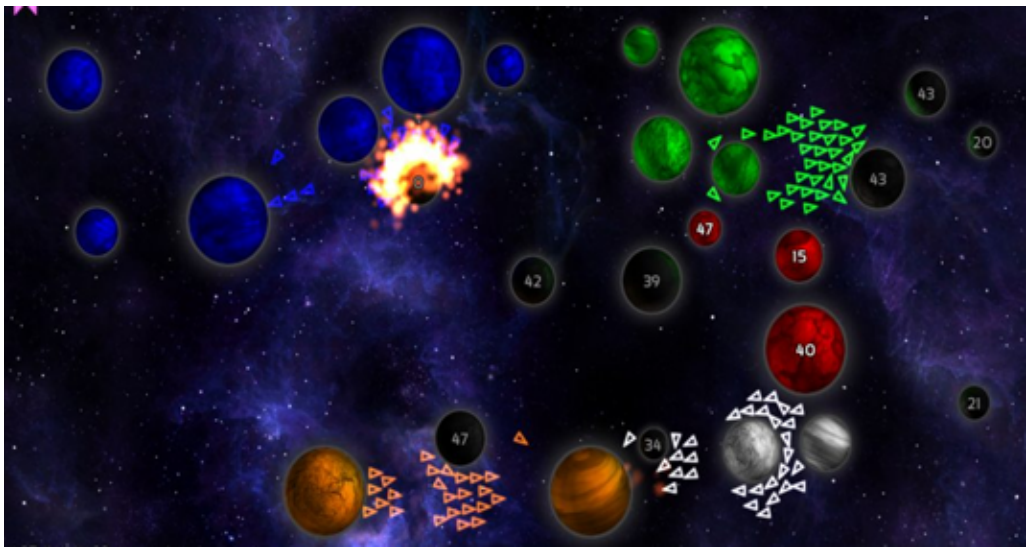


Abbildung 2: Galcon Fusion

Würden alle Raumschiffe gleichzeitig gezeigt werden, würde dieses viele Systeme

überlasten. Die Steuerung auf den mobilen Endgeräten ist sehr simpel als drag and drop gelöst. Wischt man von einem zum nächsten Planeten, starten die Raumschiffe in Richtung des Zielplaneten. Je nachdem ob dieser einem bereits gehört oder nicht landen die Raumschiffe oder beginnen einen Kampf. Insbesondere in den Menüs merkt man aber das es sich um eine Portierung eines Computerspiels handelt, Buttons werden nicht nach-skaliert und auch das Seitenverhältnis ist immer im 4:3 Format was auf den ohnehin kleinen Bildschirmen der mobilen Endgeräte nicht vorteilhaft ist. Galcon Fusion ist ein endloser aktionreicher Weltraumkampf, welcher von keiner Hintergrundgeschichte vorangetrieben wird. Selbst das Tutorial erklärt nur kurz ohne jeglichen Witz die Grundfunktionen der Steuerung.

3.3 Eufloria (X-Plattform) (JR)

Es werden steuert die sogenannten Setzlinge, gesteuert die im Grunde kleine künstliche Lebewesen sind. Es gibt noch die Auswahl ob es die stärksten, schnellsten oder die energiereichsten Setzlinge sein sollen. Das Ziel des Spieles ist es Asteroiden zu bevölkern. Diese haben verschiedene Attribute, wie Energie, Stärke und Schnelligkeit. Um einen feindliche Asteroiden zu erobern muss seine Energie auf null reduziert werden. Dies geschieht in dem sich Setzlinge opfern und in das Innere eines Asteroiden fliegen. Neutrale gehören einem sofort. Nach dem ein Asteroid erobert wurde, kann mit zehn Setzlingen ein neuer Baum gepflanzt werden. Es gibt zwei Arten von Bäumen. Einen Baum auf dem neue Setzlinge wachsen und einen auf dem Verteidigungssamen wächst.

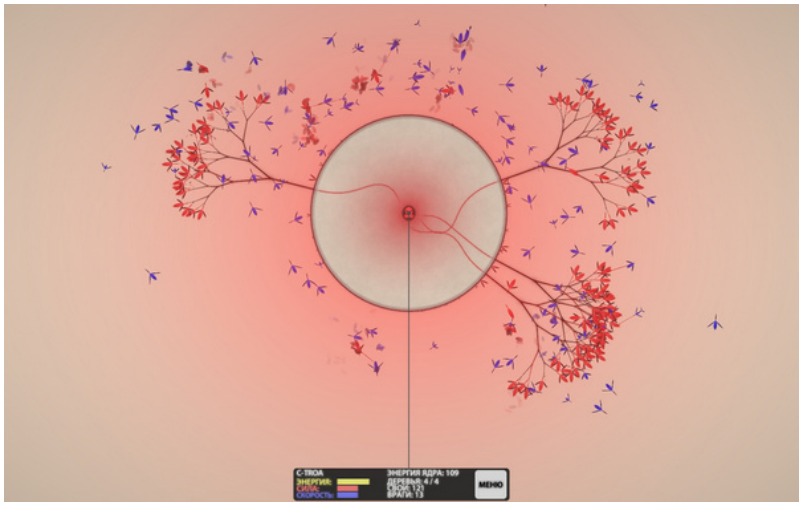


Abbildung 3: Eufloria[2]

[2] http://pervaylubov.ru/wp-content/uploads/2011/08/Eufloria-1.jpg

Letztere können nicht gesteuert werden, greifen den Feind aber im Falle einer Invasion selbständig an. Zusätzlich zu den 25 Missionen, gibt es 3 weitere Spielmodi, bei denen der Spieler zufällig generierte Einzelmissionen bestreitet.

Wie in Abbildung 3, so ist der Grafikstil in recht simpler 2D Grafik gehalten, jedoch schadet es dem Spiel nicht. Um das Spiel zu erwerben gibt es einen Festpreis dafür aber keine weiteren Kosten.

3.4 Was Drag'n Slay von anderen Spielen unterscheidet (JP)

Alle diese Spiele unterscheiden sich aber in mindestens zwei Punkten von Drag'n Slay. Erstens bietet keines der Konkurrenz Spiele eine lebendige 3D Welt. Zweitens spielen während der Arbeit bekannt gewordenen Spiele welche dieselbe Spielidee verfolgen, in leblosen 2D Welten, ohne zusätzliche Animationen. Ein weiterer Unterschied ist das Drag n Slay im Gegensatz zu manchen Konkurrenzprodukten keine Pay2Win-Philosophie verfolgt und auch keinen eigenen Kaufpreis besitzt.

Die Steuerung der Konkurrenz Spiele ist allerdings in allen Fällen sehr ähnlich. Einheiten werden immer per Drag and Drop von einem zum nächsten Ziel geschickt.

Da die Steuerung bei den Konkurrenzspielen überall ähnlich ist, lies sich die Entwicklung von Dragn' Slay davon inspirieren.

4 Konzeption (JR)

In diesem Kapitel soll geklärt werden, auf welche nDesign Entscheidungen besonders großer Wert gelegt und wo diese angwandt wurden. Außerdem werden typische Use-Cases definiert, das Gameplay analysiert und alle implementierten, sowie die geplanten Features mit deren Funktionen aufgelistet. Außerdem wird beschrieben wie die Atmosphäre des Spiels durch akustische Signale für den Spieler gestärkt wird und welche Einstellungen der Spieler selber auf seinem Endgerät tätigen kann.

4.1 Design Pillars (JR)

Anhand der folgenden Design Säulen, in Abbildung 4, wurden alle Design Entscheidungen ausgerichtet. Trotz der klar definierten Prioritäten sind Situationsabhängige Einzelentscheidungen nicht ausgeschlossen. Folgende Design Pillars wurden festgelegt:

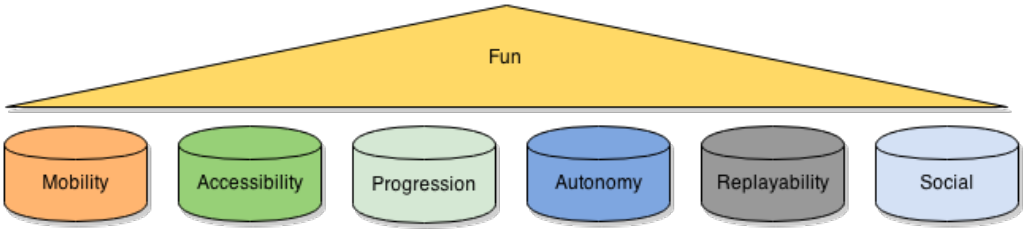


Abbildung 4: Design Pillars in Drag'n Slay

4.1.1 Mobilität (JR)

Um die größtmögliche Zielgruppe zu erreichen ist es daher sinnvoll Drag'n Slay auf dem größtmöglichen mobilen Markt veröffentlichen.

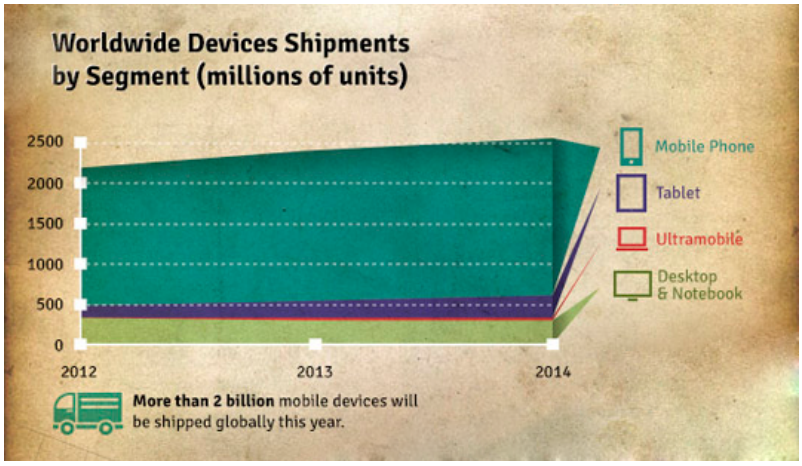


Abbildung 5: Weltweite Geräte Auslieferung nach Segment[3]

Der Statistik in Abbildung 5 nach werden mehr als zwei Milliarden mehr Tabletts und Smartphones 2014 ausgeliefert, als Laptops oder Desktop PCs. Es stellt sich die Fragen: Wie wichtig ist es für Cross Plattform zu entwickeln?

[3] http://www.digitalbuzzblog.com/infographic-2013-mobile-growth-statistics/

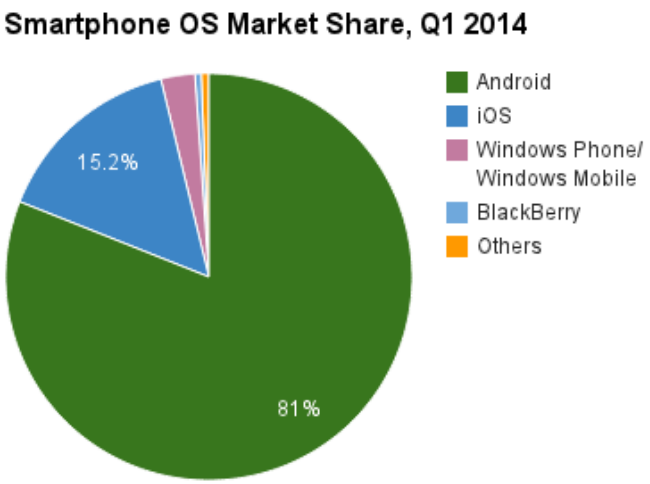


Abbildung 6: Weltweite Geräteauslieferung nach Betriebssystem[4]

In der Geräteauslieferung, wie in Abbildung 6 zu sehen ist, hat Android den größten Marktanteil mit 81%, gefolgt von iOS mit 15.2%. Somit haben Android und iOS zusammen 96.2% Gesamtmarktanteil. Zwar hat Android den weitaus höheren Marktanteil, dennoch hat Apple noch das höhere Einkommen. Laut Businessinsider[5] hat sich das Gesamtmarkteinkommen bei iOS auf 52.7% von 49.23% und bei Android auf 33.46% von 26.72% im letzten Quartal erhöht. Womit sich Android und iOS zusammen 86.16% das Einkommen des gesamten globalen mobilen Marktes teilen. Interesant ist die Frage, wie viel Zeit von Android und iOS Benutzern mit welchen Anwendungen verbracht wird.

[4] `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`
[5] `http://www.businessinsider.com/apple-revenue-android-2014-4`

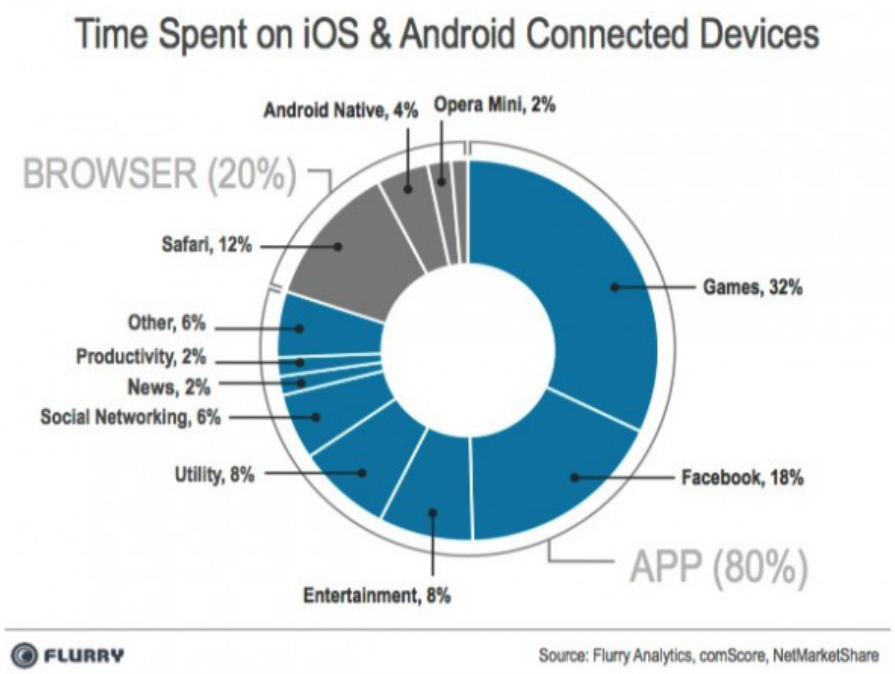


Abbildung 7: Verwendete Zeit verbundene iOS und Android Geräte[6]

In dem Kreisdiagram in Abbildung 7 geht hervor, dass in der Flurry Analytics[6] Statistik 32% der Gesamtzeit mit Spielen verbracht werden. Wird mit der technischen Begrenzung der verwendeter Tecnology die Zielgruppe erreicht?

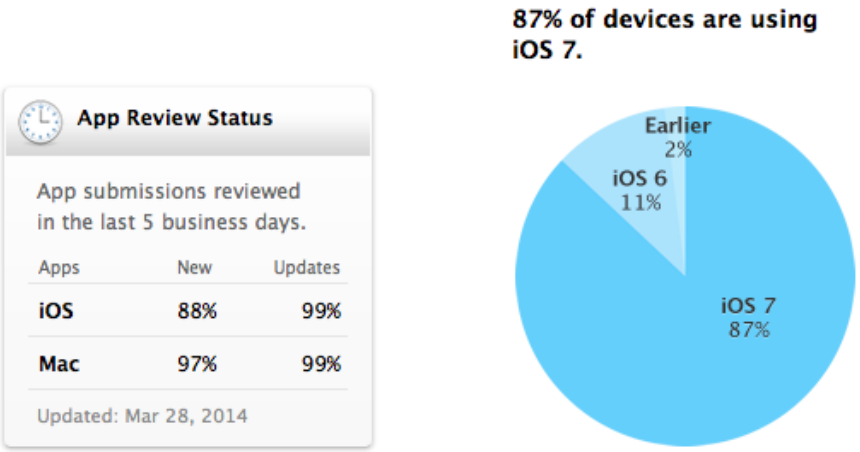
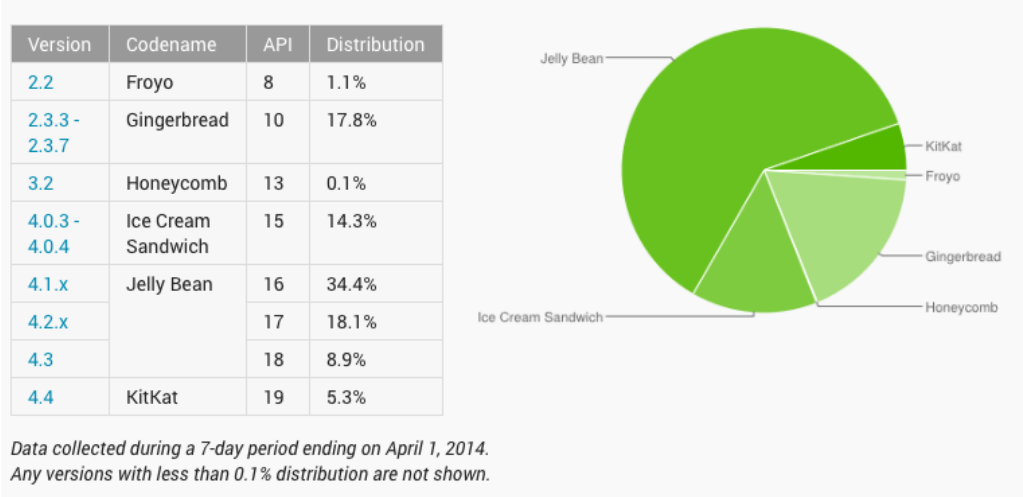


Abbildung 8: OS Version Fragmentierung iOS[7]

Die Abbildung 8 zeigt, dass die OS Version Fragmentierung bei iOS Geräten sehr gering ist. Die neueste Version iOS 7 ist bereits bei 87% aller iOS Geräte aktualisiert.

[6] http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/



Version	Codename	API	Distribution
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%

Data collected during a 7-day period ending on April 1, 2014.
Any versions with less than 0.1% distribution are not shown.

Abbildung 9: OS Version Fragmentierung Android[7]

Bei Android ist ersichtlich, dass die OS Version Fragmentierung bei wesentlich höher liegt. Wie in Abbildung 9 dargestellt, verwenden allerdings lediglich 5.3% aller Android Geräte die aktuelle KitKat Version.

Fazit

Die Etablieung im mobile Markt ist somit unabdingbar, da ein drittel aller investierten Zeit, mit mobilen Apps und Spielen verbracht werden. Durch die den hohen Marktanteilen von Android und iOS können andere Plattformen vernachlässigt werden. Um für diesen Markt zu entwickeln, wurde sich in der Arbeit für Unity3D entschieden. Da Unity3D mindestens iOS 4.0 und Android 9 unterstützt, werden dadurch 98% aller iOS und 98.9% aller Android Benutzer und sind somit auf der sicheren Seite.

4.1.2 Zugänglich (JR)

Hierbei ist es uns wichtig, so viele Personen wie möglich anzusprechen. Dazu gehört nicht nur, dass es visuell ansprechend und einladend sein sollte, sondern auch selbsterklärend und leicht zu Handhaben. Die Anforderungen Drag'n Slay zu spielen sollten hingegen so gering wie möglich gehalten werden. Außerdem ist es uns wichtig dem Spieler zu jeder Zeit audiovisuelles und habtisches Feedback über das derzeitige Geschehen zu geben. Zusätzlich wird angestrebt Wartezeiten so gering wie möglich zu halten. Interface blockierenden Elemente sollen reduziert werden, so dass der Spieler das Gefühl hat, ständig beschäftigt zu sein.

[7] `http://appleinsider.com/articles/14/04/07/apples-ios-7-reaches-87-adoption-still-growing-fas`

4.1.3 Entwicklung (JR)

Generell ist es für den Spieler sehr motivierend, wenn ihm die Möglichkeit gegeben wird, sich zu verbessern. So verdient der Spieler jedes mal Soft-currancy, wenn er ein Spiel erfolgreich beendet, welches er Shop gegen Upgrades eintauschen kann. In Drag'n Slay gibt es funktionale und kosmetische Upgrades. Um kein Pay-to-Win Spiel zu erstellen, kann der Spieler nur kosmetische Upgrades gegen Echtgeld eintauschen. Außerdem kann ein Spieler in Ranked Games gegen andere Spieler im Wettkampf antreten. Mehr dazu unter *Ranked Elo System*. Das Spieltempo nimmt über den Verlauf einer Partie ebenfalls zu. Zusätzlich ist es geplant, eine Singleplayer Sage zu implementieren.

4.1.4 Autonomität (JR)

Gutes Gameplay entsteht wie für Sid Meier *"Aus einer Reihe interessanter Entscheidungen."*[8] Der Spieler sollte eine Strategie entwickeln können, um seine Aktionen unter Berücksichtigung seines Gegners zu planen. Dennoch sollte auch jegliche Situation spontan erfassbar bleiben. In Drag'n Slay kann der Spieler sich entscheiden, welche Inseln er zu erst einnimmt, da jede Insel eigene Eigenschaften hat. So gibt es Inseln mit besonders hoher maximaler Bevölkerung, aber auch Inseln mit besonders schneller Flugzeugproduktion, sowie Inseln, die keine Einheiten produzieren, sondern einen besonders vorteilhafte strategischen Wert besitzen. Außerdem kann der Spieler Upgrades in seinem Profil vor jeder Runde neu kombinieren. Zum Beispiel können somit Flugzeuge mehr Schaden pro Sekunde austeilen oder schneller Hitpoints und Schilde regenerieren. Aber auch Spezialattacken können hier neu ausgewählt werden. Natürlich kann auch das Aussehen der Flugzeuge und Inseln beeinflusst werden.

4.1.5 Wiederspielbar (JR)

Es sollte für Beginner als auch für erfahrenere Spieler gleichermaßen anspruchsvoll bleiben. So können neue Spieler alle Funktionen in der Singlplayer Saga spielerisch erlernen. Für erfahrene Spieler haben gibt es wettbewerbsfähigen Multiplayer und skalierbare Computerspieler, sowie generierte Levels. Auch das ausprobieren der Upgrade Kombinationen ändern des Aussehens der Inseln und der Flugzeuge, sowie die Möglichkeit mit Freunden zu spielen, regt zum wieder spielen an.

4.1.6 Soziale Interaktion (JR)

Hierbei ist es besonders wichtig, dass der Spieler nur wenigen Schritten benötigt, um hat mit anderen Personen zu interagieren. So gibt es in Dragn's Slay Ranked

[8] `http://en.wikipedia.org/wiki/Gameplay`

Spiele auf Knopfdruck. Der Multiplayer sollte natürlich fair sein. Das bedeutet es sollte ein System existieren, um unterschiedliche Skill Levels zu balancieren, jeder Spieler muss potentiell die gleichen Chancen auf einen Sieg haben. Die Levels müssen balanciert sein. Eine andere Interaktionsmöglichkeit wäre das Spielen mit Freunden. Es ist geplant, Facebook Freunde zu einem Spiel einladen zu können. Es wird auch die Möglichkeit geben die Profile von anderen Spielern anzuschauen um seine eigenen Fähigkeiten mit denen seiner Freunden zu vergleichen.

4.2 Player Use Cases (JR)

In diesem Abschnitt wird einerseits die Menüsteuerung des Spielers und andererseits die Steuerung im Spiel vorgestellt.

4.2.1 Menüsteuerung (JR)

Abbildung 10: Vielfältigen Use Case in Menüsteuerung

Wenn der Spieler das Spiel startet wird er von einem kurzen und aufgeweckten Intro

zum spielen eingeladen, welches er jederzeit überspringen kann. Darauf hin wird das Hauptmenü angezeigt. Hier hat der Spieler die Möglichkeit verschiedene Spiel Modi zu starten, im Shop etwas zu kaufen, sein Profil anzusehen und zu verwendeten Upgradekombinationen neu einzustellen, Spieleinstellungen zu ändern und schließlich Drag'n Slay zu beenden. Wie ebenfalls in Abbildung 10 dargestellt, hat der Spieler die Möglichkeit verschiedenen Spiel Modi zu spielen. Er kann zum Beispiel bis zu drei weitere Freunde einladen, um zusammen Ranked Spiele zu spielen.

4.2.2 Aktionen im Spiel (JR)

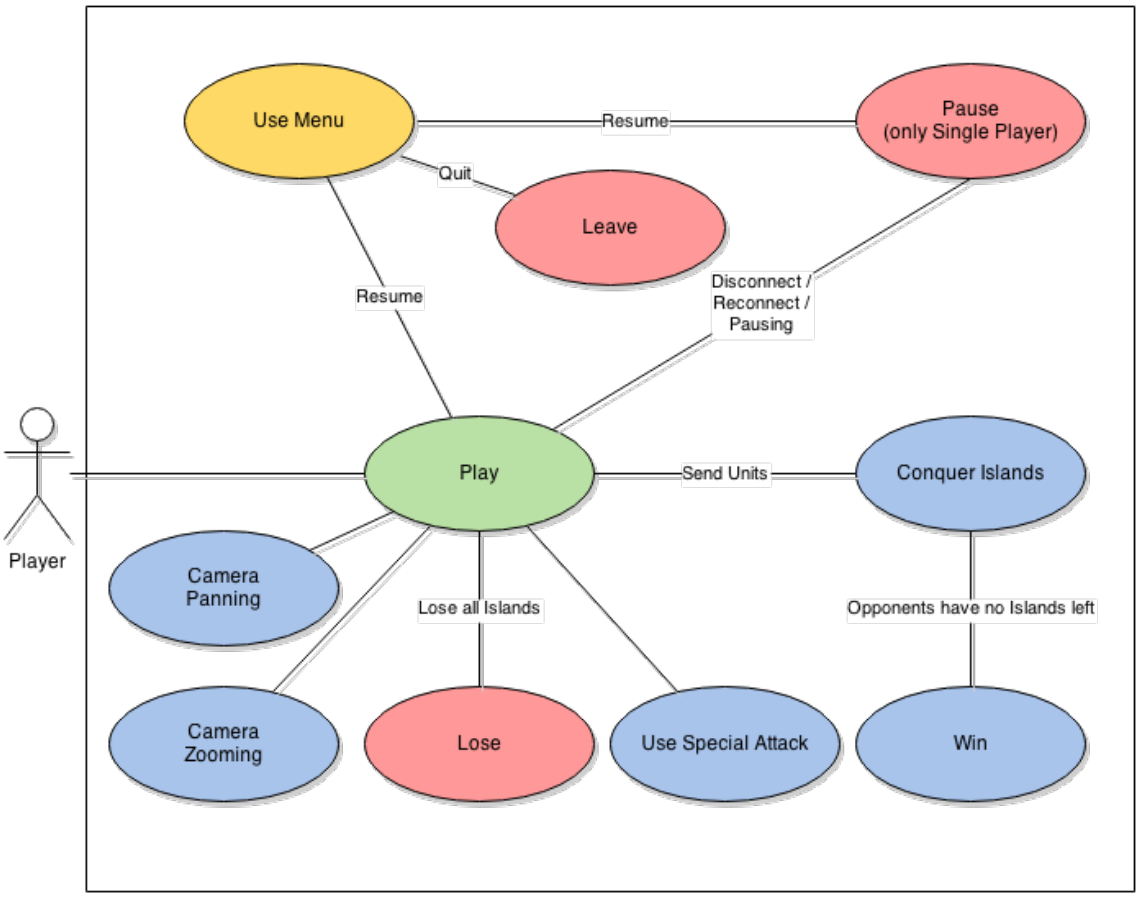


Abbildung 11: Use Case in Game

Gut dargestellt sind in Abbildung 11 die Vielfältigkeit der Interaktionsmöglichkeiten der Spieler. Der Spieler in jeder Runde die Möglichkeit Flugzeuge zu versenden um Inseln zu erobern, oder Spezialattacken zu verwenden. Gewonnen hat der Spieler der alle gegnerischen Inseln eingenommen hat. Sollte ein Spieler Verbindungsabbrüche erleiden, so pausiert das Spiel und fährt bei wiederverbinden fort. Außerdem kann der Spieler im Singleplayer das Spiel pausieren lassen, indem er ins Menu geht. Gewinnt ein Spieler so verdient er sich Elo und Cloud Coins. Sollte er jedoch verlieren, verliert

er Elo, bekommt jedoch einen verringerten Betrag an Cloud Coins. Sollte der Spieler das Spiel verlassen, so verliert er Elo und bekommt keine Cloud Coins. So werden Spieler die das Spiel beenden belohnt und Leaver bestraft. Natürlich kann der Spieler auch jederzeit die Kamera steuern.

4.3 Funktionen (JR)

Nichts charakterisiert Drag'n Slay so sehr, wie die detaillierte Beschreibung aller geplanten und implementierten Features. Alle Funktionen wurden in 13 Unterkategorien unterteilt. Um jedoch die Übersicht zu behalten wurde für dieses Kapitel eine Tabellarische Beschreibungsform gewählt. Die Tabellen bestehen aus dem Namen der Funktionen, dessen Beschreibung und im Falle der fertig gestellten Implementierung ein Kreuz in der letzten Spalte.

4.3.1 Spielmodi (JR)

In der folgenden Tabelle 1 sind alle geplanten Spiel Modi mit deren Bedeutungen aufgelistet. Wichtig für den Prototypen waren jedoch erst einmal nur ein Einzelspieler Demo Level, sowieo ein 1vs1 Multiplayer Level zu implementieren.

Name	Beschreibung	Impl.
Einzelspieler Demo Level	Prototyp Level	x
Einzelspieler Saga	Einzelspieler Kampagne mit Story	
Einzelspieler Random Levels	Zufallsgenerierte Level gegen die Ai	
1vs1	Multiplayer Ranked 1vs1	x
2vs2	Multiplayer Ranked 2vs2	
3vs3	Multiplayer Ranked 3vs3	
4vs4	Multiplayer Ranked 4vs4	

Tabelle 1: Spiel Modi

4.3.2 Spielmechaniken (JR)

In der folgenden Tabelle 2 sind detailliert Spielmechaniken mit deren Bedeutungen aufgelistet. Wie im Use-Case Kapitel beschrieben, versendet der Spieler jedoch hauptsächlich Flugzeuge durch das selektieren von Inseln, um gegnerische Inseln einzunehmen. Um Drag'n Slay interessanter zu gestalten, wurden zusätzlich dekorative Features, wie die zufälligen Bewegungen der Inseln und die Loopings der Flugzeuge

implementiert.

Name	Beschreibung	Impl.
Pathfinding	Gameobjekt bewegt sich auf ein Ziel zu.	x
Inseln: Zufällige Bewegungen	Insel führt zufällige subtile Rotations- und Translationsbewegungen auf X-,Y- u. Z-Achse	x
Inseln: Selektierbar	Raycast von Touchkoordinaten um Inseln zu selektieren.	x
Flugzeuge: Umkreisen Inseln	Flugzeuge fliegen im Orbit einer Insel	x
Flugzeuge: Variationen des Umkreisens	Flugzeug führt zufällige subtile Rotations- und Translationsbewegungen auf X-,Y- u. Z-Achse	x
Flugzeuge: Random Loopings	Flugzeuge führen zu zufälligen Zeitpunkten einen Looping aus.	x
Raketen: Interpolierte Flugbahn	Raketen fliegen beschleunigt auf einer glaubhaften Flugbahn zu ihrem Ziel.	x
Interpolation / Tweening	Verschiedene Easing Funktionen; Siehe Kapitel Interpolation.	x
Flugzeuge: Orientierungsausrichtung	Flugzeuge richten ihren Forwärtsvektor in der Richtung aus zu der sie fliegen.	x
Raketen: Ziel ausgerichtet	Z-Achse angepasst, so dass die Richtung zum Ziel abhängig vom Winkel zur Kamera stimmt.	x
Upgrades Kombinieren	Gekaufte Upgrades im Profil einstellen.	
Einkaufen	Cloud Coins, kosmetische und funktioniale Upgrades kaufen.	
Increase Game Pace	Spieltempo erhöhen über den Verlauf einer Partie.	

Tabelle 2: Spielmechaniken

4.3.3 Kamera (JR)

In der folgenden Tabelle 3 sind detailert Kamera Features aufgelistet. Unity3D bringt eine Perspektive und Orthographische Kamera Perspektive mit, jedoch muss diese auch passend konfiguriert verwendet werden. Neben dem mappen der Touchinputs des Spielers für das Panning und Zoomen der Kamera haben wir ebenso implementiert, dass die Kamera geschüttelt wird, sollte der Spieler gewinnen oder verlieren. Außerdem schwänkt die leicht Kamera, sollte der Spieler für eine kurze Zeit nichts machen.

Name	Beschreibung	Impl.
Perspektiven Kamera	Standard Kamera im Spiel (build in Unity3d)	x
Orthographische Kamera	Minimap (build in Unity3d)	x
Panning	Kamera bewegt sich auf der X- und Y-Achse.	x
Zooming	Heranzoomen durch Gestensteuerung auf der Z-Achse.	x
Idle	Leichte Kameraschwänken auf der X- und Y-Achse bewirkt Lebendigkeit.	x
Shaking	Bildschirm wird geschüttelt.	x
Flying First Person	First Person Kamera zum Debuggen.	x
Lookat Gameobject	Interpolierter, zu einem Gameobjekt, schwenkender Blick.	x

Tabelle 3: Kamera

4.3.4 UI Elemente (JR)

In der folgenden Tabelle 4 sind verschiedene UI Elemente aufgelistet. Mehr dazu unter *Asset Produktion*.

Name	Beschreibung	Impl.
Main Menu	Hauptmenü.	x
Hud	Benutzerinterface im Spiel	x
Minimap	Karte im Spiel, wurde jedoch für den Moment wieder entfernt, da nicht benötigt.	(x)
Transitions	Visuelle Übergänge zwischen den Menüs.	x
Splash Screen	Bildschirm beim starten von Drag'n Slay.	x
Special Attacks	Buttons für Spezial Angriffe.	x
Decorative Elements	Dekorative Menü Elemente.	x
Loading Screen	Animierter Ladebildschirm.	x
Win Screen	Anzeige bei Sieg.	x
Losing Screen	Anzeige bei Niederlage.	x
Buttons	Buttons. Siehe mehr unter *Menu Buttons*	x
Network	Indikator für Netzwerkverkehr. (Abbildung 52)	x
Profil	Anzeige Spielerstatistiken. (Siehe auch *Ranked Elo System*)	

Resource	Anzeige Resourcen.	
FPS	Anzeige FPS (Abbildung 53)	x
Ping	Anzeige Ping (Abbildung 53)	x
App Icon	Drag'n Slay Application Icon (Abbildung 37)	x
Upgrades Screen	Screen zum Einstellen aller erworbenen Upgrades.	
Shop Screen	Screen zum kaufen von Upgrades.	

Tabelle 4: UI Elemente

4.3.5 Grafik (JR)

In der folgenden Tabelle 5 sind verschiedene Grafik Features aufgelistet. So verwenden wir in Drag'n Slay Textur Atlanten, Partikel, 3D Modelle und eine Skybox. Spieler Aktionen, wie das Selektieren von Inseln werden ebenfalls visuell hervorgehoben.

Name	Beschreibung	Impl.
Textur Atlanten[9]	Mehre Grafiken zusammen.	x
Particle: Wolken	Simulieren von Wolken.	x
Particle: Flugzeug Explosion	Explosionsanimation von zerstörten Flugzeugen.	x
Particle: Insel einnehmen	Animation für das erfolgreiche einnehmen von Inseln.	x
Particle: Fulgzeug Rauchspuren	Flugzeuge ziehen eine Abgasrauchwolke hinter sich her.	x
Inseln: Visualisierung Selektieren	Visuel erkennbare, angewählte Insseln, sowie durch Linien dargestellte geplante Flugrichtungen.	x
Skybox	Hintergrund.	x
Animated Tiled Textures	Animierte Texturen für mehr Lebhaftigkeit.	x
Meshes	3D Meshes für Inseln und Flugzeuge.	x
Animation Meshes	Animierte Meshes, Bsp.: Windmühle.	x
Player colors	Spielerfarben zur Darsteller von Zugehörigkeiten.	x
Particle: Wasserfall	Wassferallanimation an Inseln.	x
Particle: Sparks	Im Raum herumschwierende Partikel, für mehr Lebenhaftigkeit	x

Tabelle 5: Grafik

4.3.6 Utility (JR)

Wie der der folgenden Tabelle 6 aufgelistet, benutzen wir Unity3D mitgelieferte Komponenten, wie zum Beispiel Prefabs. Wir haben ebenso für das Entwickeln eigene Hilfen implementiert, wie zum Beispiel eine Debug Log Console für den Client, so dass wir zur Laufzeit Logausgaben auf dem Tablett oder Handy direkt auf dem Display sehen können.

Name	Beschreibung	Impl.
Debug Console Client	Debug Runtime Logausgaben Client. Togglebar durch Menü Button.	x
Prefabs	Benutzen von build-in Unity3d Prefabs.	x
Manipulation: Time	Benutzen von build-in Unity Timescale.	x
Manipulation: FPS	Skalieren der Framerate.	x
Manipulation: Turntime	Skalieren Rundenzeiten. Siehe *Konsistenz: FPS vs Ping*	x
Touch-to-Mouse-Input-Converter	Gleichbehandlung von Maus- und Toucheingaben.	x

Tabelle 6: Utility

4.3.7 Sound (JR)

In der folgenden Tabelle 7 sind alle verwendeten Sounds mit deren Verwendungszweck aufgelistet. So gibt es in Drag'n Slay Audio Feedback für Spezial Attacken, das Spawnen von Flugzeugen, das einnehmen von Inseln, Kampfgeräusche, aber auch Hintergrundmusik und extra mitreißende Musik für das Intro.

Name	Beschreibung	Impl.
Theme Song	Hauptsong von Drag'n Slay	x
Spezial Attacken	Verschiedene Geräusche für Spieler Spezialattacken.	x
Wind	Kurzes loopbares Geräusch für Wind	x
Flugzeuge Spawn	Belohnendes Geräusch, wenn Flugzeuge von Inseln los fliegen.	x
Inseln Spawn	Kurzes Geräusch, wenn eine Insel auftaucht.	x
Insel einnehmen	Subtiles spannungsaufbauendes Geräusch für wenn der Spieler erfolgreich eine Insel erobert hat.	x
Abbrechen Inseln einnehmen	Negativer kurzer Sound, der das erfolgreiche verteidigen einer Insel erklingen lässt.	x

Kampfgeräusche	Hintergrundgeräusche für Kampfsituationen.	x
Gewinnen	Belohnender Sound.	x
Verlieren	Mitleiderregender, aber motivierender Sound erneutes spielen.	x
Explosionen	Badabum Geräusch.	x
Intro	Mitreißende, zu Partie startendnen einladende Musik, wenn Drag'n Slay gestartet wird.	x

Tabelle 7: Sound

4.3.8 Habtik (JR)

In der folgenden Tabelle 8 sind geplante Vibrationen aufgelistet. Diese sollen dem Spieler neben den audiovisuellen auch den habtischen Sinn ansprechen.

Name	Beschreibung	Impl.
Buttons	Knopf druck in Menus.	
Gewinnen	Bei Sieg.	
Verlieren	Bei Niederlage.	
Spezial Attacken	Beim aktivieren von Spezial Attacken.	
Ranked Tier aufsteigen	Aufsteigen einer Rank Stufe.	
Ranked Tier absteigen	Absteigen einer Ranked Stufe	
Kaufen	Zum Kauf bestätigen.	
Einstellungen	Zum Bestätigen von geänderten Einstellungen. (Siehe *Player Settings*)	

Tabelle 8: Haptik

4.3.9 Kampfsystem (JR)

In der folgenden Tabelle 8 sind die einzelnen Elemente des in Drag'n Slay implementierten Kampfsystem beschrieben. So besitzen alle Flugzeuge und Inseln eine Lebenskomponente. Sobald durch der in Angriffen entstanden Schaden mit Rüstung und Schilden nicht weiter verteidigt werden kann, werden Flugzeuge zerstört und Inseln übernommen. Flugzeuge regenerieren ihre Schilde und reparieren ihre Rüstung über Zeit hinaus. Außerdem ist es natürlich auch wichtig, dass Flugzeuge nur Feinde angreifen.

Name	Beschreibung	Impl.
Angreifen	Flugzeuge können schaden anrichten und feindliche Flugzeuge zerstören, durch das versenden von Raketen.	x
Verteidigen	Flugzeuge verwenden ihre Rüstung und Schilde, wenn sie angegriffen werden.	x
Repair HP	Gesundheitspunkte regenerieren sich.	x
Regenerate Shields	Schilde regenerieren sich.	x
Inseln einnehmen	Inseln können übernommen werden.	x
Feinderkennung	Flugzeuge greifen nur Feindliche Einheiten an.	x
Leben	Flugzeuge und Inseln besitzen Gesundheits- und Schildpunkte.	x

Tabelle 9: Kampfsystem

4.3.10 AI (JR)

In der folgenden Tabelle 10 sind geplante Schwierigkeitstufen für die Künstliche Intellegenz aufgelistet. Für den Prototypen greift die AI lediglich gegnerische Ziele zufällig an.

Name	Beschreibung	Impl.
Random AI	AI greift Ziele zufällig an.	x
Pirates	Offensive unparteiische AI, Zufallsereignisse	
Unranked	Standard AI	
Bronze	Spielstärke eines Bronze gerankten Spielers	
Silver	Spielstärke eines Silber gerankten Spielers	
Gold	Spielstärke eines Gold gerankten pielers	
Platin	Spielstärke eines Platin gerankten Spielers	
Diamond	Spielstärke eines Diamand gerankten Spielers	
Impossible	Schwerstmögliche AI.	

Tabelle 10: AI

4.3.11 Balancierung (JR)

Für spätere Versionen von Drag'n Slay sind die in der Tabelle 11 aufgelisteten Bilanzierungen geplant. Besonders Upgrades, die AI, Level und Spezial Attacken müssen angepasst werden, dass der Spieler mehr als nur eine zum Sieg führende Strategie

entwickeln kann.

Name	Beschreibung	Impl.
Heimatinsel	Spieler wird bestraft, sollte er bei Spielbeginn direkt die gegnerische Heimatinseln angreifen.	x
Upgrades	Upgrades balancieren, um mit unterschiedliche Spielstilen zu unterstützen.	
Spezial Attacken	Alle Spezialattacken haben ihre Daseinsberechtigung.	
AI	Schwierigkeitsgrade der AI skalieren.	
Level	Levelaufbau fair und gewinnbar für alle Spieler.	

Tabelle 11: Balancing

4.3.12 Netzwerk (JR)

In der folgenden Tabelle 12 sind die allgemeinen Netzwerksfunktionen aufgelistet. Mehr dazu unter *Multiplayer*.

Name	Beschreibung	Impl.
Level von Server laden	Clients laden Leveldaten vom Server.	x
Dynamische Server Kontaktdaten	Automatisches aktualisieren der Server IP Addresse und Port, so dass sich Clients automatisch zum Server verbinden, ohne manuel IP Addressen ändern zu müssen.	x
Unique Ids synchronisieren	Synchronisieren von Ids zwischen Clients und Server zur eindeutigen Identifizierung von Gameobjekten.	x
Netzwerk Events	Handlen von Netzwerk Events auf Server und Client Seiten.	x
Node.js Socket.io	Verwendung von Socket.io Websocket API mit dem Node.js zur Kommunikation verbundener Clients.	x
Monitoring Tool	Runtime Monitoring Tool für Netzwerk- kommunikation über den Server.	x
Strategie gegen Konsistenzkonflikte	Implementierung Age-of-Empires Netzwerk- Loop	x

Spielräume	Server verwaltet Spielräume, so dass nur miteinander spielende Spieler miteinander kommunizieren.	x
Auto Join or Create	Automatisches Beitreten oder Erstellen von Spielräumen	x

Tabelle 12: Netzwerk

4.3.13 Netwzerk Event API (JR)

In der folgenden Tabelle 13 sind alle implementierten Netzwerk Events aufgelistete. Natürlich gibt es Events, die nur der Server (S in Tabelle) oder der Client (C in Tabelle) bearbeiten braucht, welche jeweils mit einem Kreuz gekennzeichnet sind. Mehr dazu unter *Multiplayer*.

Event Name	**Bedeutung**	**S**	**C**	Impl.
message	Standard Nachricht	x	x	x
connection	Neue Verbindung	x	x	x
disconnect	Verbindungsabruch	x	x	x
reconnect	Wiederverbinden	x	x	x
error	Laufzeitfehler	x	x	x
uncaughtException	Unbehandelte Laufzeitfehler	x	x	x
send	Textnachrichten	x		x
ping	Ping	x	x	x
pong	Pong	x		x
os-stats	Server Statistiken	x		x
create-game	Neues Spiel erstellen	x		x
join-game	einem offenem Spiel beitreten, sonst wie *create-game*	x	x	x
client-game-ready	Client ist bereit	x		x
server-game-ready	Server ist bereit		x	x
waiting-for-player	Warte auf Spieler		x	x
leave-game	Spiel verlassen	x	x	x
request Nachricht	anfordern	x		x
start-game	Spiel starten	x	x	x
pause-game	Spiel pausieren	x	x	x
resume-game	Spiel fortsetzen	x	x	x
stop-game	Spiel beenden	x	x	x
turn-done	Spielzug fertig	x	x	x

game-data	Spielstand oder Leveldaten		x	x
spawn-unit	Einheit erstellen	x	x	x
move-unit	Einheit bewegen	x	x	x
unit-arrival	Einheit ist angekommen	x	x	x
unit-destroyed	Einheit wurde zerstört	x	x	x
unit-fires-rocket	Einheit fürt eine Rackete ab	x	x	x
island-convert	Insel wird eingenommen	x	x	x
island-converted	Insel wurde eingenommen	x	x	x
special-attack	Spezial Attacke aktivieren	x	x	x
latency	Latenzgeschwindigkeit		x	x

Tabelle 13: Server und Client Events[10]

4.4 Das Steampunk Universum (JP)

Das Steampunk Universum beschreibt ein Endzeitszenario, welches in einer parallelen Welt mit den beiden gegenläufigen Hauptelementen Maschinen und Magie hantiert. Die Maschinen erinnern hierbei sehr stark an eine Mischung aus den frühen Dampfmaschinen des 19. Jahrhunderts, wie sie während der Industriellen Revolution gebaut wurden und Kriegsfahrzeugen des zweiten Weltkrieges. Insbesondere Luftschiffe und Flugzeuge bilden hierbei eine zentrale Rolle. Auch die dunkle Stimmung der verrußten Großstädte während der Industriellen Revolution wird aufgegriffen. So tauchen im Steampunk an jeder Ecker Zahnräder, Druckmesser, Kupfer und Holzelemente sowie Dunkler Qualm auf. Im Gegensatz zu den Maschinen steht die Magie. Sie taucht oft in Verbindung mit der Natur auf und bildet grundlegende Elemente wie leuchtende Kristalle und fliegende Inseln, die eine Art Protest an der Zerstörung an der Natur durch die Maschinen sind.

4.5 Gameplay (JR)

In dem Kapitel werden alle Spielmechaniken und Spielregeln erläutern. Vor allem interessant ist es, wie der Spieler mit dem Spiel interagiert.

4.5.1 User Feedback (JR)

Wie bereits in dem Abschnitt Zugänglichkeit angesprochen, streben wir eine Möglichst hohe Zugänglichkeit für den Spieler an. Wie in Abbildung 12 dargestellt, bedeutet

[10] Server implemtation: `http://goo.gl/rgvnKZ`, Client implementation: GameMP.cs `http://goo.gl/8zAtnx` und Game1vs1.cs `http://goo.gl/C8GLdR`, sowie Admin Console: `http://goo.gl/zaWXFB`

dies dem Spieler audiovisuel und habtische Kanäle über alle Aktionen im Spiel zu informieren, so dass der Spieler immer genau weiß was passiert und er das Gefühl von Kontrolle hat. So können zum Beispiel Ereignisse durch unterschiedlich rhythmische Vibrationen und Vibrationstärken verschieden kommuniziert werden und so mehr interesante Abwechslung bieten.

Abbildung 12: Kanäle optimaler Feedback

So werden selektierte Inseln durch Linien visualisiert, wie in Abbildung 13 gut zu sehen ist. Subtile Vibrationen, sowie leise Klickgeräusche werden beim tatsächlichen aus- und einwählen ausgelöst. Daher weiß der Spieler sofort welche Inseln nun ausgewählt sind und wohin die Flugzeuge beim loslassen hin fliegen werden.

Abbildung 13: Senden von Flugzeugen

Ebenso Buttons im Menü kommunizieren mit dem Spieler. Genauere Ausführungen erfolgen in der Sektion *Buttons im Menu*. Kamerabewegungen sind klar und deutlich zu sehen. Sie benötigen keine audio oder habtik Rückmeldungen Bei Sieg oder Niederlage wird die Kamera geschüttelt. Es tönen Explosionsgeräusche und Vibrationen auf. Ebenso Flugzeugexplosionen sind hörbar. Sie sind durch visuelle Partikelanimationen deutlich zu erkennen. Zusätzlich spielt in Kampfszenen eher eine kampflastige und im Shop oder Profil eher eine enthusiastische Hintergrundmusik.

4.5.2 Menu Buttons (JP)

Wie bereits erwähnt sind die allgemein üblichen Touch Feedback States in einem Spiel nicht dringend erforderlich, aber zumindest Bestimmte sollten dennoch beachtet werden. Die wichtigsten werden in der Abbildung "Touch Feedback States" gezeigt. Da insbesondere auf einem Touchscreen kein physischen Hervorhebungen für die Buttons vorhanden sind und sie sich damit nur optisch von ihrer Umgebung unterscheiden, muss man sich als Entwickler überlegen, auf welche Weise man dem User signalisieren möchte, das er den Button getroffen hat und damit eine Aktion ausgelöst hat. So sollte auch bei jedem Button in einem Spiel zwischen einem "normalen" und einem "Gedrückten" Zustand unterscheiden werden, um dem User zu Signalisieren, das er mit dem Finger erfolgreich sein Ziel getroffen hat. Da der Button während des Klickens durch den Finger verdeckt ist, sollte hier mit einem kleinen delay gearbeitet werden, damit der Button auch nach entfernen des Fingers noch kurz im "Pressed-State" dargestellt wird. Zusätzlich können mit Hilfe des Vibrationsmoduls kurze Impulse ausgeben werden, welche dem User ein physisches Feedback geben. Durch großzügiges nutzen von Vibrationen als physisches Feedback kann jedoch die Akkulaufzeit des Endgerätes und die damit verbundene Spielfreudigkeit des Kunden sinken. Abhilfe kann an dieser Stelle ein akustisches Signal schaffen, was dem User mitteilt, das ein Button gedrückt wurde. Dieses sollte aber nicht länger als eine halbe Sekunde klingen und in sich dezent sein, da es sonst sehr schnell nerven kann.

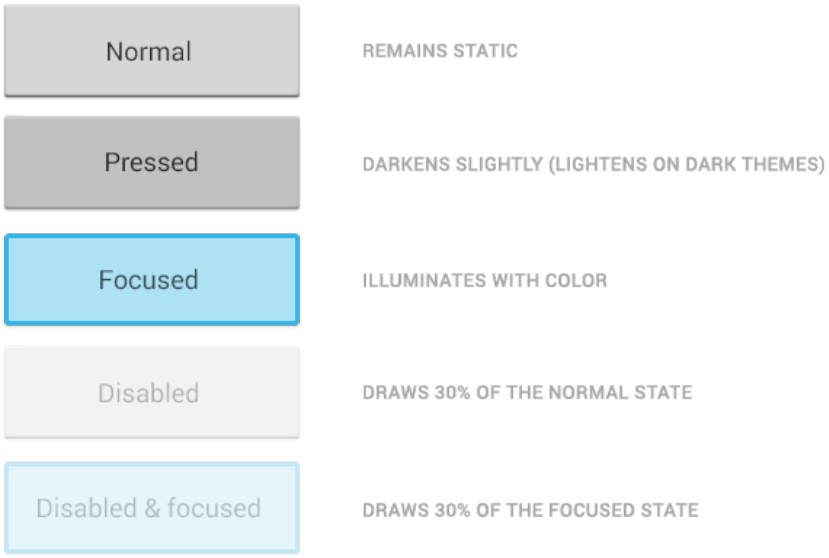


Abbildung 14: Touch Feedback States[11]

[11] Quelle: `http://developer.android.com/design/style/touch-feedback.html`

Der sonst häufig verwendete "Focused-State" wird in Drag'n Slay nicht verwendet, und wird wahrscheinlich in wenigen anderen Spielen eine Daseinsberechtigung haben, da man in einem Spiel erwartet das jeder Button zu einer sofortigen Aktion führt. Nach dieser Aussage ist es möglich auch auf die Idee zu kommen, dass Gleiches für einen Button mit "Disabled-State" zählt. Dieses ist aber nicht der Fall. Zum ersten besteht die Möglichkeit, dass Optionen in den Settings gerätebedingt nicht möglich sind. Der zweite Fall bezieht sich auf die "Skill-Buttons" des "in game UI". Da ein Skill nach dem Benutzen erst eine Cooldown erhält bis er wieder genutzt werden kann muss dieses dem User signalisiert werden. Durch das Ausgrauen des Buttons wird dem User deutlich gemacht, das dieser momentan nicht aktiv ist. Zusätzlich zeigen wir an dieser Stelle dem User einen Timer über dem Button an, welcher anzeigt wie lange es noch dauert bis der Skill wieder verfügbar ist. Diese drei Buttons haben zusätzlich noch die Eigenschaft, dass sie einen Kurzen Vibrationsimpuls aussenden, wenn sie wieder in den aktiven Status wechseln. Hierbei vibriert der erste Skillbutton einmal, der Zweite zweimal und der Dritte dreimal um dem Benutzer auch ohne hinsehen zu signalisieren, das die Fähigkeit wieder bereit ist. So wird der Spieler nicht durch erzwungenes überwachen des UI's aus dem Spielgeschehen gerissen und kann sich auf das eigentliche Spiel konzentrieren.

4.5.3 Versenden von Flugzeugen (JR)

Auf mobilen Geräten, wie Handys und Tabletts hat der Spieler keine Maus und keine Tastatur, sondern einen Touchscreen. Zur Folge bieten sich besonders Finger Gesten für die Steuerung an, z.B.: Drag and Drop. Einer der größten Nachteile von Touchscreens allerdings ist der Prezisionsverlust durch den vergrößerten "Mauszeiger". Deshalb sind Micromanagements[12] in Echtzeit besonders schwierig zu steuern. Zum Beispiel ist es bei größeren Schlachten leichter Schwärme zu steuern, als jede Einheit separat. Daher sendet der Spieler in Drag'n Slay immer mehrere Flugzeuge auf einmal von einer Insel zur nächsten, um diese einzunehmen, anstelle von einzelnen Flugzeugen, in der Abbildung 13 gesehen werden kann. Das die Flugzeuge um Inseln herumschwirren, erleichtert dem Spieler die Positionierung zusätzlich.

[12]Verwalten von jeden kleinen Schritt in einem Prozess.

Abbildung 15: Ablaufplan Spieler Steuerung innerhalb einer Partie

Der in Abbildung 15 dargestellte Ablaufplan, beschreibt den Selektionsvorgang. Der Spieler verbindet Inseln mit Linien, um so seine Flugzeuge von seinen Inseln zu der zuletzt ausgewählten Insel zu schicken. Beim ankommen der Flugzeuge, stellt sich nun die Frage, ob der Spieler diese Insel besitzt. Gehört sie ihm, so wird die Spawnrate angepasst. Sollte das Bevölkerungslimit auf dieser Insel durch die gerade angekommenen Flugzeuge erreicht sein, so werden keine weiteren Flugzeuge mehr gespawnt.

In der folgenden Formel wird berechnet, wie viele Einheiten pro Sekunde spawnen.

$$\begin{aligned}\text{Einheiten pro Sekunde} = Clamp[\ &\text{Spawnrate} * \\ &(1 - \frac{\text{Anzahl Eigene Schiffe} + \text{Anzahl Gegnerische Schiffe}}{\text{Bevölkerungslimit}}), \\ &0, \text{Bevölkerungslimit}]\end{aligned} \tag{1}$$

Gehört die Insel jedoch einem gegnerischen oder neutralem Spieler, so wird entweder darum gekämpft oder es wird damit begonnen die Insel einzunehmen, abhängig davon, ob gegnerische Flugzeuge anwesend sind.

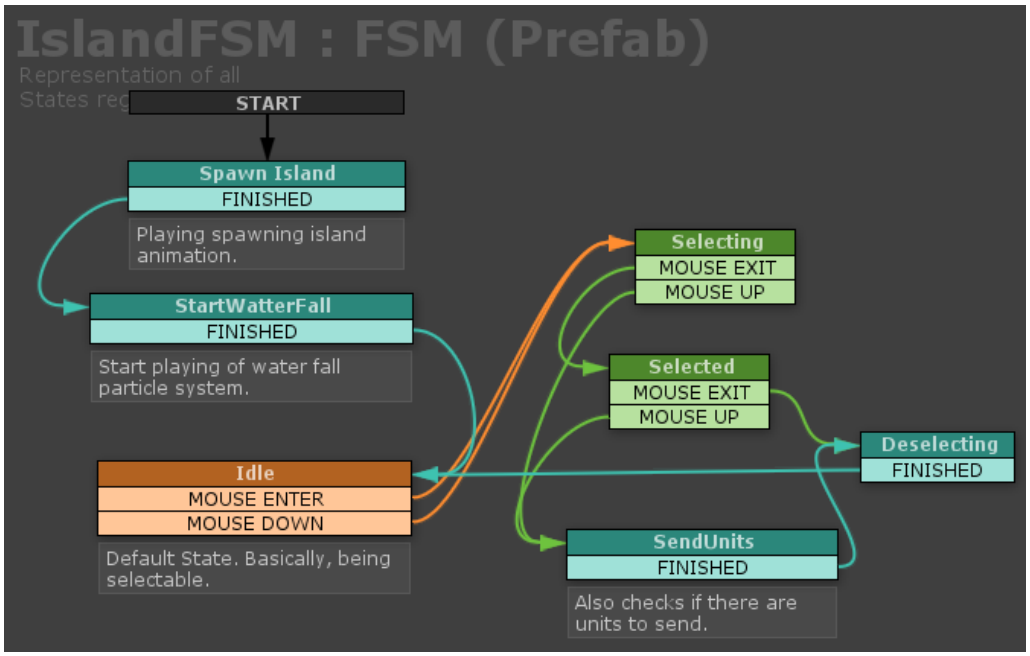


Abbildung 16: Statemachine Selektieren

Die tatsächliche Implementierung des Selektiervorgangs unterstützt auch das Abspielen unterschiedlicher Animationen durch die in der Abbildung 16 dargestellten Zustände und ihre Übergänge. So wird die Inselfarbe erhellt, sollte der Spieler diese gerade selektiert haben und somit den Zustand in den *Selected* gewechselt haben.

Jede Insel hat ebenso eine Spawning Statemachine, Abbildung 17. Diese ermöglicht das Animieren bei der Übernahme, sollte eine Insel in den Zustand *ChangeOwner* wechseln, oder wenn ein Schiff im Zustand *Spawning Ship* spawned.

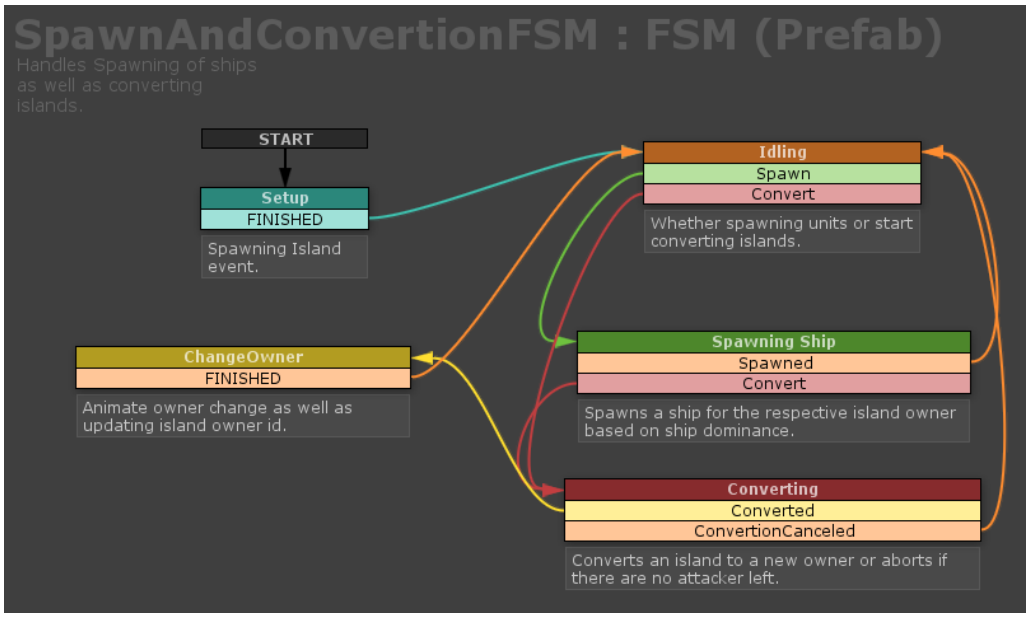


Abbildung 17: Statemachine Spawning von Flugzeugen und Erobern einer Insel

4.5.4 Progression (JR)

In Drag'n Slay kann der Spieler sich Achievements und Badges verdienen, aber auch seine Ranked-Einstufung beeinflussen, sowie Upgrades erwerben. Daher werden diesem Kapitel diese Entwicklungsmöglichkeiten für den Spieler näher beschrieben.

4.5.4.1 Archivements (JP)

Archivements dienen in erster Linie zur Steigerung der Spielermotivation. Durch sie erhält der Spieler ständig das Gefühl kleine Fortschritte im Spiel zu machen. Seitens der Entwickler bieten die Archivements eine schnelle und damit kostengünstige Möglichkeit das Spiel durch neu implemierte Archivements zu erweitern und dem Spieler auf diese Weise neue Inhalte zur Verfügung zu stellen. Die Archivements lassen sich in zwei verschiedene Arten unterschieden. Die einen lassen sich durch den normalen Spielverlauf erhalten, im betrachteten Fall zum Beispiel erfolgreiches beenden des Anfänger Tutorials. Für andere müssen spezielle Ziele erfüllt werden, wie zum Beispiel eine Multiplayer Runde in unter einer Minute zu gewinnen oder auch bereits alle anderen Archivements zu besitzen.

Obwohl die Archivements eigentlich nur virtuelle wertlose Auszeichnungen sind und keinen Mehrwert bieten, streben viele Spieler nach ihnen. Das dieses Prinzip funktioniert zeigt der Erfolg der großen Spieleschmieden. Dennoch stellt sich uns die Frage warum dieses Model so erfolgreich ist.

Es kann davon ausgegangen werden, dass es an dieser Stelle wieder an der menschlichen Natur liegen, sich mit allem und jedem Messen zu wollen. Dieses wird insbesondere unter Freunden der Fall sein. Wer hat die Singleplayer Kampagne zuerst abgeschlossen oder mehr Gegner besiegt? Hingegen die Tatsache ob ein anonymer Spieler dieselben oder mehr Archivements freigeschaltet hat, wird dem Spieler egal sein.[13]

Wird dieses Prinzip weiter gedacht, kann dieses Streben insbesondere von große Spieleschmiede genutzt werden.wird zum Beispiel für alle Spiele aus dem eigenen Haus pro User nur ein Benutzerkonto angelegt, können übergreifende Archivements genutzt werden die zu einer Punktzahl zusammen gerechnet werden. Hat ein Spieler bereits alle Archivements in einem Spiel erhalten und möchte mehr Punkte sammeln, ist er gezwungen ein weiteres Spiel aus dem selben Haus zu Spielen und in dieses zu Investieren. An dieser Stelle sind die Archivements auch eine gute Werbestrategie, wenn man dem User die Möglichkeit gibt mit einem Klick das erhaltene Archivement zum Beispiel auf Facebook zu Posten. So sehen die Freunde des Users nicht nur das Archivement, sondern erfahren unter Umständen auch noch von einem Spiel das sie nicht kannten. Oft sind die Interessen innerhalb eines Freundeskreises identisch.

Alternativ können die Archivements auch als Leitfaden durch das Spiel genutzt werden, welcher den Spieler dazu bringt, dass Spiel vollständig auszureizen und den Spieler damit indirekt zu der von den Entwicklern gedachten Vorgehensweise im Spiel zwingt. Da der Spieler von Drag'n Slay das Spiel aber auf seine eigene Weise entdecken soll, nutzt Drag'n Slay diese Möglichkeit abgesehen von einer kleinen Ausnahme nicht. Die Ausnahme besteht hierbei im Absolvieren des Training Levels, welches als Voraussetzung für den Multiplayer gälten soll. Mit Erhalt dieses Archivements erhält der Spieler zeitgleich sein erstes Badges, das ab diesem Augenblick sein Profil ziert.

4.5.4.2 Badges (JP)

Badges sind eine Erweiterung der Archivements und dienen damit ebenfalls der Spieler Motivation. Der Begriff leitet sich vom Englischen badge ab, dass zu Deutsch ein Abzeichen oder eine Medaille beschreibt. Diese Badges erhält der Spieler, wenn er besonders schwere Archivements abschließt. An Stelle echter Medaillen erhält der Spieler kleine Bildchen, die er wie in einem Stickerbuch sammeln kann. Optisch ähneln die Badges in Drag'n Slay militärischen Orden, die an die Steampunk The-

[13] Quelle: Evolution des Bewußtseins: Grundlagenforschung der korrelativen Verhaltens-Psychologie, 978-3826038730

matik angepasst sind, siehe hierzu (Abbildung 18).

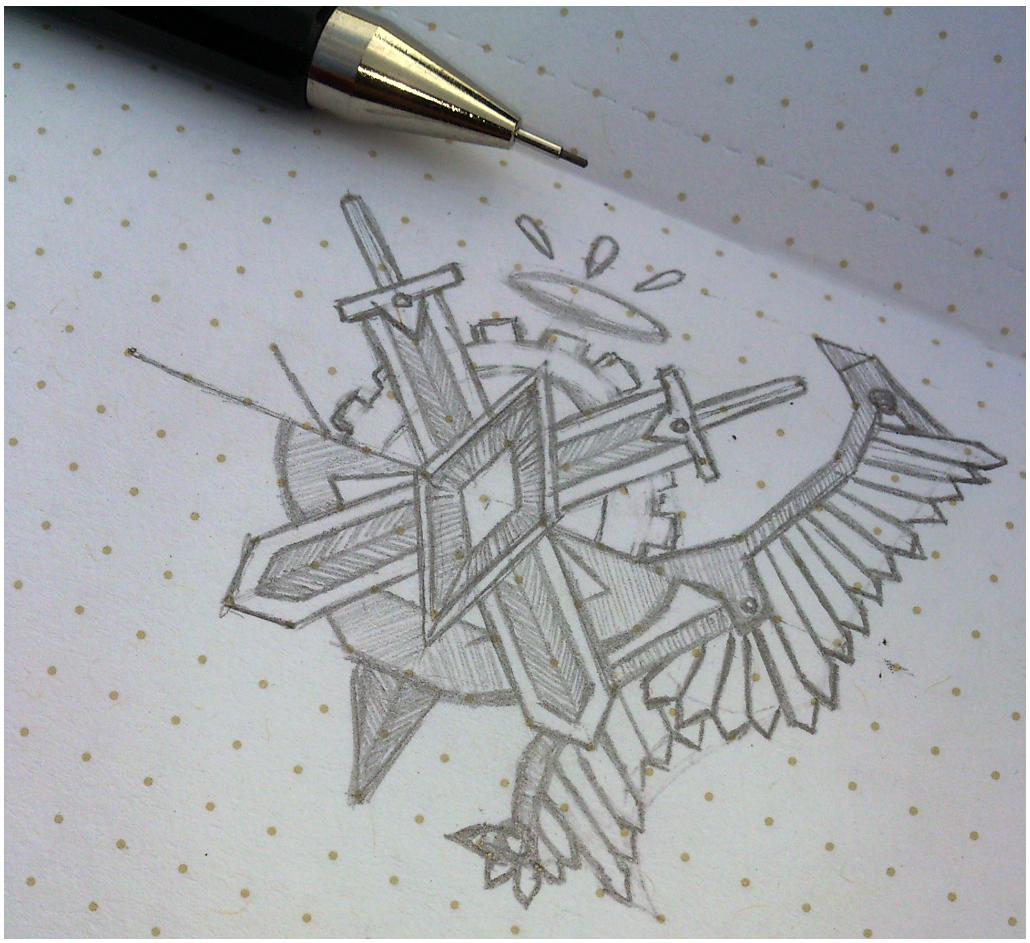


Abbildung 18: Badge

Auch die Badges geben dem Spieler keinen Mehrwert, haben für ihn aber trotzdem eine noch höhere Wertigkeit als die Badges, da sie schwerer verdient werden müssen.

Im Unterschied zu den normalen Badges gibt es noch fünf, die der Spieler für den eintritt in eine höhere Liga erhält. Diese sind in Kohle, Kupfer, Bronze, Silber und Gold eingeteilt. Die fünf besonderen Badges kann der Spieler im Gegensatz zu den Normalen wieder verlieren, wenn er zu lange an keinem Ranglistenspiel Teilgenommen hat oder wegen Schlechter Leistungen eine Liga abgestiegen ist. So soll der Spieler motiviert werden kontinuierlich an Ranglistenspielen teilzunehmen.

4.5.4.3 Upgrades (JR)

Im Shop hat der Spieler die möglichkeit Upgrades mit Cloud Coins zu erwerben. Hierbei sind, neben den noch geheimen Spezialattacken, hauptsächlich die folgenden passiven Verstärkungen für den Kämpfe zwischen Flugzeugen zu kaufen:

HP: Gesundheitspunkte.

HP Regen: Gesundheitsregeneration.

Armor: Rüstung

Shield: Schilde

Shield Regen: Schildregeneration

Attack Speed: Angriffsgeschwindigkeit

Attack Damage: Angriffschaden

Armor Penetration: Rüstungsdurchdringen

Shield Penetration: Schilddurchdringung

Invasions Speed: Invasionsgeschwindigkeit

4.5.4.4 Ranked Elo System (JR)

Um besondere Wiederspielbarkeit zu erreichen hat ein Spieler die Möglichkeit gegen andere Spieler zu spielen und Elo[14] Punkte bei Sieg zu verdienen oder aber bei einer Niederlage Elo Punkte zu verlieren. Die Grundidee des Systems ist es, unterschiedliche Spieler Skill Levels zu balancieren, so dass so wenig wie möglich einseitige Spiele gematcht werden.

Startet ein Spieler ein Ranked Game, so versucht der Server nun einen Spieler zu finden, der ähnlich viel Elo erspielt hat. Aus Gründen der oben erläuterten Spielerfragmentierung ist es nicht immer möglich. Aus diesem Grund sind die Computerspieler in Dragn' Slay skalierbar.

[14] `http://en.wikipedia.org/wiki/Elo_rating_system`

Abbildung 19: League of Legends Tier Icons[15]

Jeder Spieler startet bei 1200 Elo und bei bestimmten Grenzen werden Titel und Medaille aktualisiert. Die Medaillen könnten so aussehen, wie die von League of Legends, welche in Abbildung 19 zu sehen sind..

In der unten aufgeführten Tabelle 14 wird aufgelistet, wie viel Elo jeweils welche Medaille darstellt.

Unranked	<=	5 Ranked Spiele
Bronze	<=	1250 Elo
Silber	>	1250 Elo
Gold	>	1500 Elo
Platin	>	1800 Elo
Diamand	>	2200 Elo

Tabelle 14: Ranked Status und benötigte Elo

Die Anzahl an Elo die gewonnen oder verloren werden, hängen in erster Linie von dem Unterschied der gematchten Spieler ab, jedoch wird mindestens immer um ein Elo gespielt. Es gibt noch ein paar andere Faktoren, wie zum Beispiel die Spieler Performance. Hierbei bekommt ein extrem dominierender Spieler mehr Punkte um

[15] `http://na.leagueoflegends.com/en/competitive/ranked-leagues`
[15] `http://blog.kibotu.net/design/texture-packer`

schneller aufzusteigen. Daraus wird folgerichtig gegen stärkere Gegner gematcht, aber auch anders herum.

Die jetzige Formel berücksichtigt eine Siegserie und den Skillunterschied zwischen den gematchten Spielern. Jedoch gibt es ein Limit wie viele Punkte maximal pro Match erworben werden können.

```
Erworbene Elo =     Max[ Abs( EloSpieler1 - EloSpieler2 )
                  +  Abs( Anzahl Spiele Erolgsserie EloSpieler1
                  -  Anzahl Spiele Erolgsserie EloSpieler2 )
                  *  Erfolgsserien Multiplikator, Maximal Erwerbare Elo ]
```

Der Erfolgsserien Multiplikator und die maximal erwerbaren Elo müssen noch ausführlich getestet werden.

Fazit

Hierbei werden paradoxerweise sowohl Sieger als auch Verlierer belohnt. Sieger bekommen Elo und schließlich eine bessere Medaille in ihrem Profil angezeigt, aber diese müssen nun in kommenden Runden gegen stärkere Spieler antreten. Was oft übersehen wird, ist dass die Verlierer nun gegen schlechtere Spieler gematcht werden. Somit werden sie indirekt mit erhöhten Siegchancen ebenso belohnt werden. Die Möglichkeit Punkte, einen Titel und eine Medaille zu verdienen, sowie sich mit anderen Spielern messen zu können, steigert den Langzeitspielspass für alle.

Weiterhin geplant ist es die Spielperformence zu berücksichtigen. Eine Idee wäre es, erwartete und tatsächliche Spieldauer und die Einheiteneffizienz mit einzubeziehen.

4.5.5 Game Modes (JR)

Wie in Tabelle 1 aufgelistet, soll es mehrere Spiel Modi geben. So hat der Spieler die Möglichkeit im Singleplayer seine Skills zu trainieren und diese mit Freunden im Muliplayer gegen eins bis vier weiteren Spieler einzusetzen. Die Grundmechaniken aller Spiel Modi sind jedoch gleich und die Zustände des Spiels können mit der Statemachine in Abbildung 20 implementiert werden.

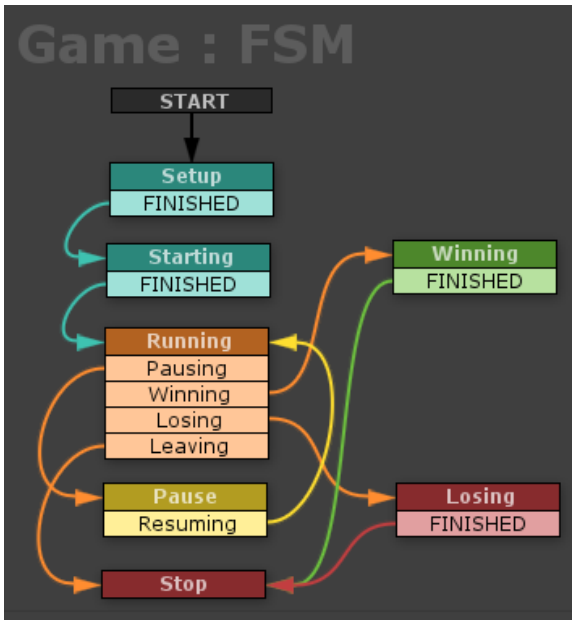


Abbildung 20: Statemachine Game

Nach einer Initialen *Setup* Phase, in der das Spiel die Spielwelt lädt, wird das Spiel gestartet. Während des *Startens* fährt die Kamera in Richtung der Startinsel des Spielers. Daraufhin wechselt das Spiel in den *Running* Zustand. In diesem kann der Spieler Pausieren, die Gewinn- oder Verlierbedingung erfüllen oder das Spiel verlassen. Bei allen Zustandsübergängen kann die Kamera animiert, Sounds und Vibrationen abgespielt werden.

4.5.6 AI (JR)

Für den Single Player Modus ist geplant die AI Schwierigkeitstufen menschlich einschätzbarer zu gestalten, wie zum Beispiel dies in Ultimate General (Abb. 21 umgesetzt wurde.

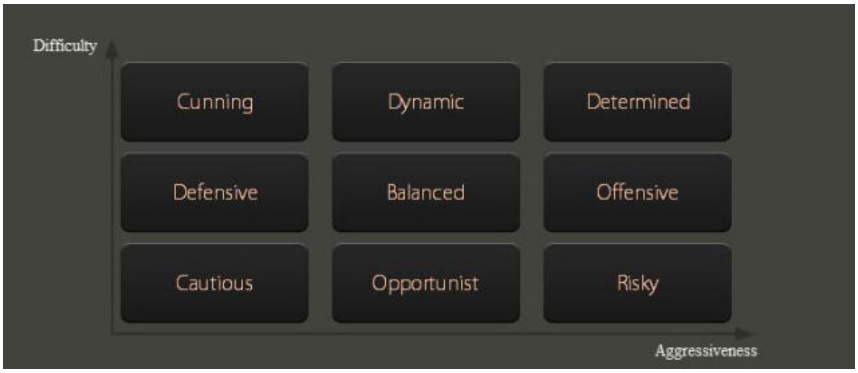


Abbildung 21: Schwierigkeitsgrade AI in Ultimate General[16]

[16] http://www.ultimategeneral.com/blog/from-darthmod-to-game-design

Die künstliche Intellegenz kann relativ komplex werden. Daher wurde für den Prototypen lediglich ein Computerspieler entwickelt, der seine Ziele zufällig auswählt und diese angreift.

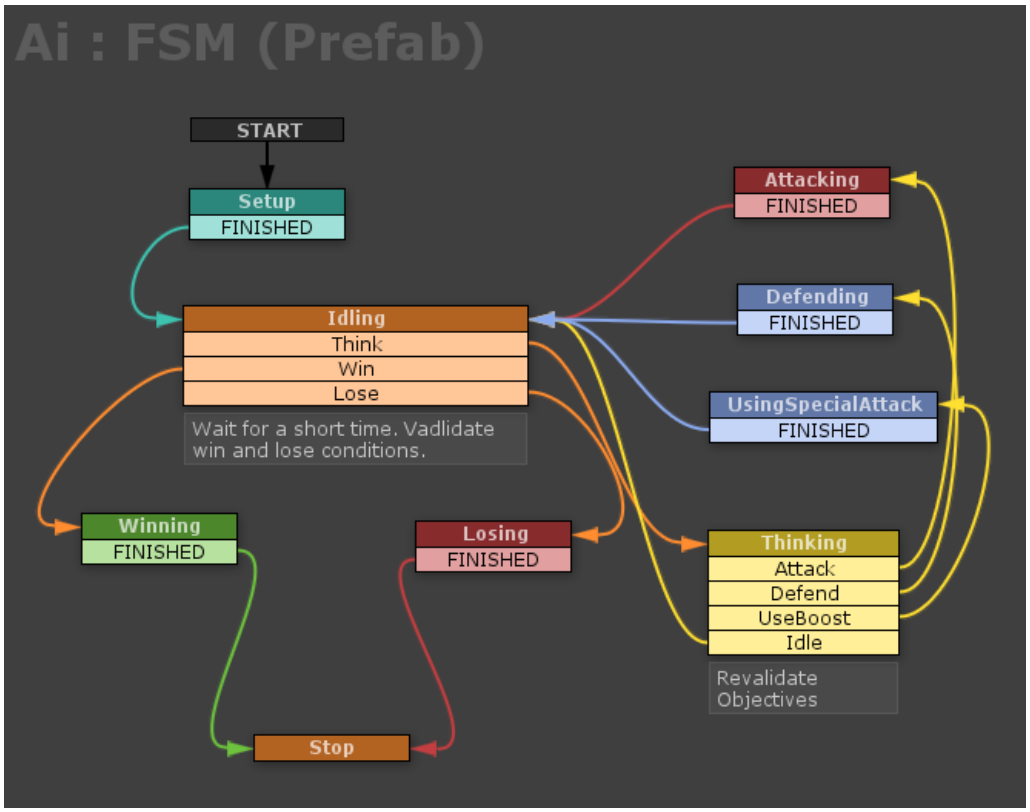


Abbildung 22: Statemachine Random AI

In der Statemachine in Abb. 22 ist der Grundaufbau für diese Ai gut zu erkennen. Nach der *Setup* Phase wird in den *Idle* Zustand gewechselt. Indem zu Eintritt jedes mal ein kurzes Interval gewartet wird, bevor daraufhin in den *Thinking* Zustand gewechselt wird. In diesem Zustand werden mögliche strategische Angriffs- oder Verteidigungsziele abgewägt. Die priorisierte Tätigkeit wird daraufhin ausfgeführt und anschließend zurück in den *Idle* Zustand zurück gewechselt.

Skalierbare Ansätze wären das Interval zu veringern bevor erneut in den *Thinking* Zustand gewechselt wird, die Anzahl an Aktionen erhöhen, die gleichzeitig ausgeführt werden können, Angriffszielprioritäten an Hand von effektiver Kampfstärke per Insel abzuwägen und die möglichkeit für Rückzüge.

4.5.7 Kampfsystem (JR)

In Drag'n Slay bekämpfen sich Flugzeuge am Himmel mit Raketen. Diese Flugzeuge besitzen eine Grundgesundheit, die bei Erschöpfung zum zerstören des Flugzeuges

führt. Natürlich ist es spannender, wenn ein paar Attribute mehr hinzufügt werden, die der Spieler zwischen den einzelnen Matches mit Hilfe von Upgrades variieren kann. So besitzt jedes Flugzeug ebenso eine Grundpanzerung, die den eingehenden Schaden der Raketen verringert und in späteren Upgrades zusätzlich Energieschilde.

Um das längere Überleben von Flugzeugen und somit überlegener Strategien guter Spieler zu belohnen, regenerieren sich die Gesundheit und Schilde über einen Zeitraum wieder. Der Schaden ist von dem Angriffschaden und der Attackrate der Angreifer abhängig, sowie von der Rüstungs- und Schilddurchdringung.

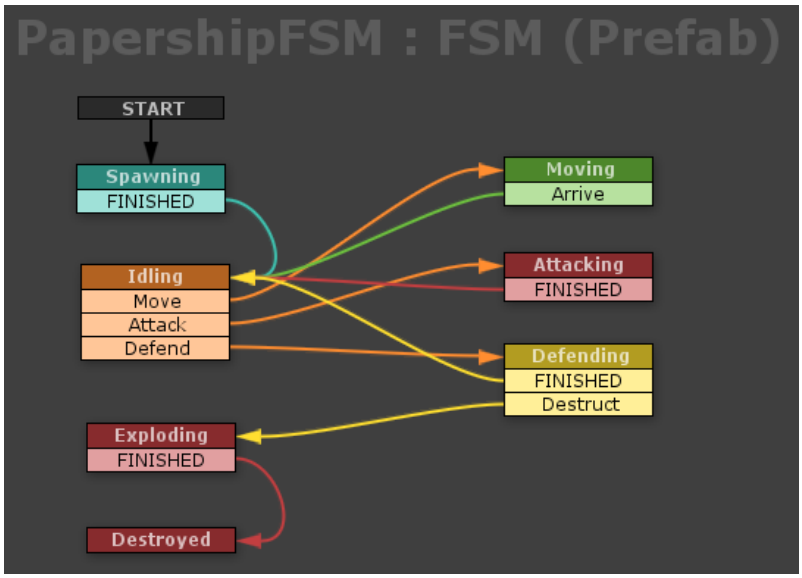


Abbildung 23: Statemachine Flugzeug

Flugzeuge können mit der in Abbildung 23 implementiert werden. Nach dem sie im Initialen *Spawning* Zustand anfangen um ihre Insel herum zu schwirren, wechseln sie in den *Idling* Zustand über. In diesem Zustand überprüfen sie, ob der Spieler diese auf die Reise zu einer anderen Insel schickt, oder ob sich gegnerische Schiffe in der nähe befinden, bzw. ob sie Angegriffen werden und animieren dies in den jeweiligen Zuständen. Der wesentliche Vorteil der Statemachine ist jedoch, dass nur eine Animation und Tätigkeit gleichzeitig ausgeführt wird. So kann ein Flugzeug das sich von einer Insel zur nächsten bewegt nicht gleichzeitig andere Flugzeuge angreifen.

Zusätzlich sind Spezial Attacken geplant, welche jedoch zu dem jetzigen Zeitpunkt vernachlässigt werden.

4.5.8 Leveldesign (JP)

Die Karten in Drag'n Slay unterscheiden sich in zwei Typen. Zum einen gibt es die symmetrisch aufgebauten Multiplayer-Karten und zum anderen die asymmetrischen Einzelspieler-Karten, bei denen der Spieler sich meist in eine bestimmte Richtung vorkämpfen muss.

Die Einzelspieler Karten sollen so konzipiert sein, das diese zum vorantreiben der Geschichte dienen. Hierbei ist egal ob der Spieler oder die künstliche Intelligenz (KI) durch den Kartenaufbau einen Vorteil erhält, da es dem Spieler wenn er gegen eine KI antritt egal ist. Daher sollte an dieser Stelle die Möglichkeit genutzt werden, die Karten möglichst unterhaltend und abwechslungsreich zu gestalten. Wenn sich der Spieler mit anderen menschlichen Spielern in einer Multiplayerschlacht misst, ist es anders. Hier würde sich der Spieler mit hoher Wahrscheinlichkeit über den kleinsten Nachteil beschweren, den er seitens der Kartengestaltung erhält. Daher wurde für Drag'n Slay entschieden die Multiplayer-Karte für jeden Spieler Symmetrisch zu gestalten, sodass kein Spieler durch die Karte einen Vor- oder Nachteil erhält.

Die zweite Herausforderung im Leveldesign ist es richtige Entfernungen zwischen den einzelnen Inseln zu wählen. Hierbei haben wir uns zunächst nur auf eine Regel, welche Mindest- und Maximalabstand regelt festgelegt. Die Flugzeit zwischen zwei Inseln muss mindestens 2 Sekunden dauern und darf eine Zeit von 4 Sekunden nicht übersteigen. Der Mindestabstand ist notwendig, damit man als Spieler vor dem eintreffen der Flieger reagieren kann.

4.6 Sound (JP)

Für den Prototyp des Spiel Drag'n Slay wurden alle Soundeffekte so wie jegliche Musik selber erstellt und komponiert. Diese sind in rein digital erstellte Audiodaten und analoge mikrofonierte Schallquellen zu unterscheiden. Alle Aufnahmen und Bearbeitungen sind mit der Digital Audio Workstation (DAW) Reaper getätigt wurden. Zusätzlich sind zahlreiche Virtual Studio Technology (VST) und Virtual Studio Technology Instrument (VSTi) Plugins zum Einsatz gekommen. Der Technische Aufbau, siehe Abb.24, zum modellieren der digitalen Sounds besteht neben dem Computer, die über ein externes USB-Audio-Interface verfügt auch aus einem Microkorg Synthesizer und einem Paar frequenzneutraler Midrange Monitore welche durch einen Pioneer SA-9500 II angetrieben werden.

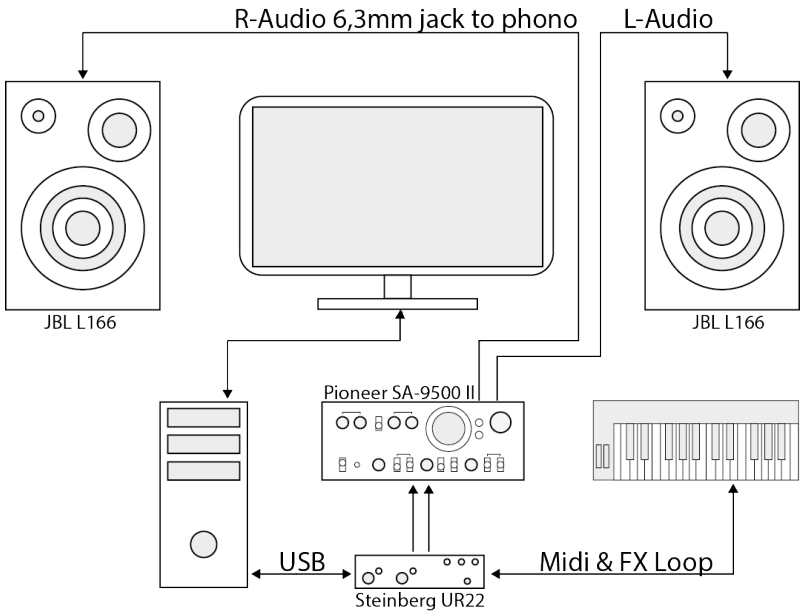


Abbildung 24: Technischer Aufbau der Audio Engineering Umgebung

4.6.1 Hintergrundmusik (JP)

Die Akkordfolgen und Rhythmen der Hintergrund Musik für Drag'n Slay wurden mit Klavier und Gitarre komponiert. Aus Kostengründen wurde die endgültige Komposition allerdings nicht mit einem echten Orchester aufgenommen sondern mit VSTi. Um mit VSTi zu arbeiten gibt es zwei Möglichkeiten. Die erste ist das komplette Sampeln der Audiospur, dass heißt es muss jede Note einzeln in ein Zeitraster eingetragen werden. Der Nachteil dieser Methode ist, dass das Instrument sehr statisch klingen kann, da zunächst sämtliche Anschlagsdynamik fehlt. Dieses lässt sich entweder manuel erledigen oder mit so genannten Humanizern. Ein Humanizer verschiebt jede Midi-Note um einen anzugebenden Toleranzwert in Zeit und Lautstärke. Gut nutzen lässt sich diese Methode zum Beispiel für perkussive Instrumente. Die zweite Methode ist das Einspielen über ein Midi-Keyboard. Hierbei wird über den Midi-Port des Audio Interfaces eine Verbindung zum Midi-Keyboard hergestellt. Anstelle eines echten Instruments, kann man auf diese weise eine Kette an Midi-Actions speichern, welche durch das Drücken einer Taste auf dem Midi-Keyboard gesendet werden. Auf diese Weise erhält man ein dynamischeres und gefühlvolleres Klangbild, so das diese Methode insbesondere für Melodiestimmen geeignet ist. Natürlich setzt diese voraus das man die Tonfolge auf dem Midi-Keyboard spielen kann. Die Hintergrundmusik zu Drag'n Slay hat noch eine kleine Besonderheit. Sie ist so komponiert, dass sie aus zwei Teilen besteht, welche sich ohne Einschränkung zusammen setzen lassen. Das Ziel hierbei war es zwischen einer ruhigen, friedlichen

Musik in den Überlegungsphasen und einer aktionsreicheren und rythmischeren Musik während der Kampfphasen faden zu können.

4.6.2 FX (JP)

Es wurden zwei arten der Sound Effects (SFX) Erstellung genutzt. Ein Teil der SFX wurde durch unterschiedliche Manipulationen von White- oder Pink-Noise erstellt, andere aus analog aufgenommenen Geräuschen, welche durch unterschiedliche Verfahren an den gewünschten Ton angeglichen wurden.

Erstellen eines SFX aus einem linearen weißen Rauschen: Bei einem weißen Rauschen handelt es sich um ein Akkutisches Signal, dass durch ein gleichbleibendes Leistungsspektrum, bei einem definierbaren Frequenzbereich beschrieben werden kann. Das heißt es gibt ein lineares Geräusch, in dem keine Frequenzen hervorgehoben sind. Dieses kann jetzt von Grund auf neu geformt werden. Als Beispiel für eine solche Transformation, wird die Erstellung einer Explosion aus einem weißen Rauschen betrachtet. Hierzu sind mehrere Schritte notwendig. Wird der Lautstärker verlauf einer Explosion betrachtet stellt man fest das der Pegel schnell bis zur Hochstelle ansteigt und danach langsamer fällt. Es ist allerdings zu beachten, dass die gesamte Explosion aus mehreren Geräuschen besteht, somit auch aus mehreren manipulierten Rauschen. Da es sich bei Drag'n Slay um ein Spiel für ein mobiles Endgerät handelt und mit diesem eh nur eine schwache Soundwiedergabe möglich ist, reicht es die beiden Hauptbestandteile des Klanges der Explosion zu beachten. Der optimale Frequenzbereich eines Mobilen Endgerätes liegt bei ca 11KHz - 16KHz. Dieses errechnet sich aus der Formel[17]: $Frequenz = (c)/\lambda$ Hierbei beschreibt c die Schallgeschwindigkeit und lambda die Wellenlänge des Lautsprechers. Gehen wir von einem Lautsprecher mit einem Durchmesser von 30mm aus, kommen wir zu der Berechnung: $Frequenz = 330/30$ also eine eine Frequenz von 11KHz. Damit eine Membran eine Frequenz wiedergeben kann, muss die Membran größer als die wiederzugebende Welle sein.

Die Druckwelle: Da sich der Wiedergabebereich eines mobilen Endgerätes nach unten auf einen Frequenz von 11KHz beschränkt und das menschliche Gehör nicht in der Lage ist Frequenzen über 20KHz wahrzunehmen, wird ein weißes Rauschen im Bereich von 11KHz bis 20KHz erzeugt. Um das Rauschen zu einer Druckwelle zu formen, muss es mit einer Lautstärkekurve und einem Mehrband Peak Filter verformt werden. Die Lautstärkekurve steigt schnell an und fällt minimal langsamer. So erhalten wir ein Geräusch, welches sich druckvoll in Szene setzt und noch ein

[17] Quelle: Tontechnik: Schwingungen und Wellen, ISBN 978-3446423954

wenig nachklingt. Der Mehr-Band Peak Filter funktioniert ähnlich wie ein Equalizer, jedoch hebt oder senkt er die Frequenzen in einem bestimmten Bereich nicht linear sondern glockenförmig und interpoliert. So ist der Übergang zwischen den manipulierten Frequenzen immer weich. Die Abbildung: 25 zeigt die Wirkung der Filterbereiche auf die Amplitude.

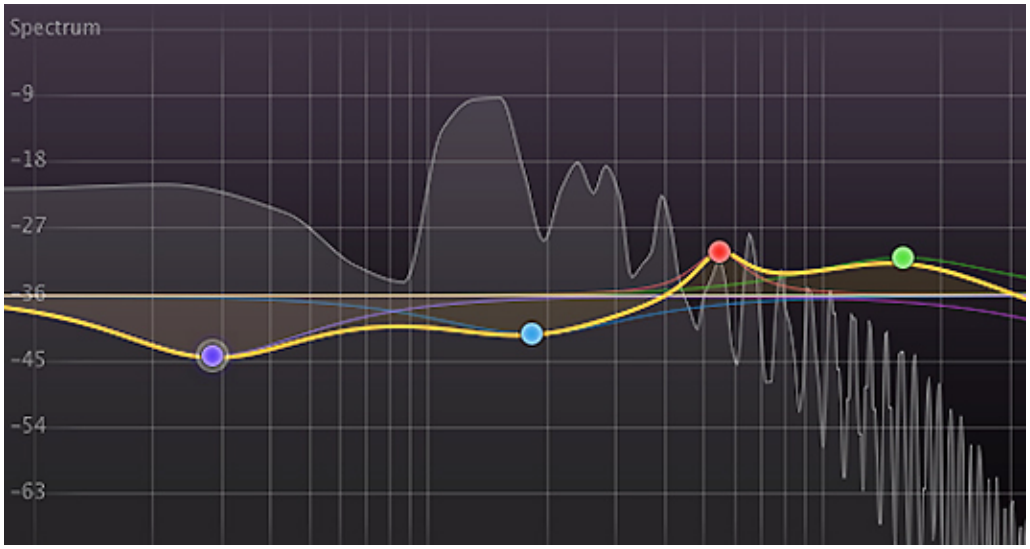


Abbildung 25: Graph eines Multi-Band peek Filters

Mit dem Peak Filter werden zuerst der tiefe Frequenzbereich aus dem Rauschen gefiltert und dann langsam wieder hinzugegeben. So entsteht durch das sich füllende Klangspektrum der Eindruck, dass die Welle näher kommt.

Das Feuer: Der zweite wichtige Klangbestandteil einer Explosion ist das prasselnde Feuer. Um dieses zu emulieren dient wieder ein Rauschen im Bereich von 11KHz bis 20KHz als Grundlage. Damit der Ton mehrere und unterschiedliche Obertöne erhält, sollte er durch einen Sägezahn-Generator laufen, welcher mit stetig wechselnden Werten arbeitet. Kombiniert man den so entstandenen Klang wieder mit einem Mehrband Peak Filter, welcher alle Frequenzen in Abhängigkeit zu einer Sinuskurve mit steigender Herzzahl und fallender Amplitude Manipuliert, erhält man ein loderndes Feuer.

Werden diese beiden Geräusche zusammen gemischt entsteht bereits der Eindruck einer Explosion. Durch das Komprimieren bestimmter Frequenzbereiche, das Hinzufügen eines Halls und Reverbs lässt sich der Klang noch verbessern und eine realistischere und druckvollere Wirkung erzielen.

Aufnahme einer analogen Schallquelle: Für andere Sounds wurde wiederum eine klassische Mikrofonierung von Gegenständen verwenden. Als Mikrofon kam ein Blue

Bluebird zum Einsatz gekommen. Da es sich bei diesem Mikrofon um ein Großmembran-Kondensator-Mikrofon handelt, ist zu beachten, dass dieses im Gegensatz zu einem Dynamischen Mikrofon eine Phantomspeisung benötigt. Der Abbildung: 26 kann man die Verkabelung und Einstellungen am Interface während der Aufnahmen entnehmen. Da der Aufnahmepegel eines Kondensator Mikrofons deutlich höher und linearer ist als der eines Dynamischen Mikrofons ist dieses aber trotz der benötigten Phantomspeisung zu bevorzugen.

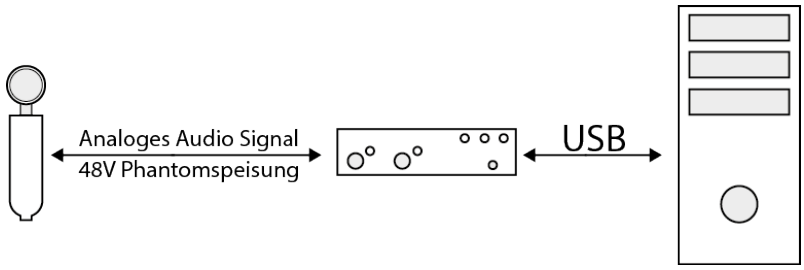


Abbildung 26: Technischer Aufbau der Recording Umgebung

Genutzt wurde dieses Verfahren zum Beispiel für die Erzeugung der Button-Sounds in den Menüs. Das aufgenommene Geräusch, ist der Klang von zwei Weingläsern, welche so mit Wasser gefüllt sind, dass sich zwei unterschiedlich zueinander passende Töne ergeben, wenn diese gegeneinander gestoßen werden.

Anpassen des aufgenommenen Tons: Der Ton klingt direkt nach der Aufnahme noch etwas dünn und unnatürlich. Dieses lässt sich aber mit drei VST beheben. Der Ton klingt etwas unnatürlich, da man die Schallquelle sehr nah am Mikrofon erzeugt. Hierdurch wird sehr wenig Raumklang mit aufgenommen. Dieses führt zu einem sehr trockenen und cleanen Signal. Fügt man dem Aufgenommenen Signal jetzt mit Hilfe eines VST einen kleinen Hall (In unserem Fall der Classik Studio Reverb (CSR) der Firma IK-Multimedia) hinzu, klingt es bereits so wie wir den Ton auch im Raum wahrnehmen. Mit Hilfe eines Mehrbandkompressors, wird der Klang noch etwas druckvoller, da durch diesen die leiseren Frequenzen auf einen höheren Pegel gedrückt werden. Mit einem Equalizer (EQ) lässt sich jetzt noch durch anpassen der Frequenzkurve der Ton modellieren.

Beim Export von Audiodatein sollte man beachten, dass auf einem Android Device .ogg Container eine bessere Unterstützung erhalten und unter IOS .MP3 Container.

4.7 Player Settings (JP)

Zwar ist geplant das die Grafikeinstellungen bei Drag'n Slay automatisch anhand der Device-Informationen gesetzt werden, jedoch soll der Spieler nicht bevormun-

den werden und ihm die Möglichkeit geben viele Einstellungen selber einzustellen. Hierdurch erhält Möglicherweise der Entwickler interessante Informationen über unterschiedliche Fehler, die bei unterschiedlichen Einstellungen entstehen. Diese helfen dabei das System ständig weiter zu entwickeln.

In den Optionen gibt es drei Reiter. Im Ersten befinden sich die allgemeinen Einstellungen in denen der Spieler die Sprache des Spiels einstellen kann. Der hier genutzte Default-Wert soll die Systemsprache sein. Außerdem findet der User hier eine Einstellmöglichkeit für den Schwierigkeitsgrad des Single Player Modes in drei Stuffen.

Im zweiten Reiter befinden sich die Audio-Einstellungen. Neben einer allgemeinen Mute Funktion, welche sämtliche Sounds deaktiviert gibt es auch zwei Regler mit der sich die Lautstärke der Hintergrundmusik sowie die der Hintergrundgeräusche (FX) stufenlos Regeln läßt.

Der dritte Reiter soll dem Spieler die Möglichkeit geben die Grafik mit Hilfe dreier vordefinierter Profile einzustellen und unabhängig davon die Texturqualität und maximale Framerate zu setzen.

Auf der niedrigsten der drei Einstellungsmöglichkeiten sind außer der Edge Detection, die für den Toon-Effekt sorgt, alle anderen Grafikeffekte deaktiviert um auch auf schwachen Endgeräten eine ausreichende Framerate zu erzielen. Bei der Mittleren Einstellung sollen zusätzlich Sun Shafts und der Bloom Effekt aktiviert werden. Der in Unity 3D als Sun Shafts beschriebene Effekt wird normaler weise als God Rays beschrieben, Lichtstrahlen die durch die Wolken brechen. Bei Drag'n Slay werden diese nicht von der Sonne ausgesendet sondern von jedem hellen Bereich in der Skybox. Der Bloom Effekt kommt ursprünglich aus der Digitalen Fotografie und beschreibt ein Phänomen was durch die Überlastung der Pixel auf dem Sensor entsteht. Bei der Entwicklung der Grafikshader wurde dieser Effekt aufgenommen, um die Bilder realistischer wirken zu lassen. Hierbei werden alle Bereiche im Bild welche einen bestimmten Helligkeitswert überschreiten weichgezeichnet. Die dritte und Höchste Grafikeinstellung wird nur mit den Spitzenmodellen des ersten Quartals 2014 flüssig laufen. Hier wird zusätzlich ein Motion Blur Effekt aktiviert, welcher zwar eine schöne Optik und Dynamik ins Spiel bringt aber das Endgerät auch sehr belastet. Gleichzeitig wird ein Retro-Effekt aktiviert. Dieser setzt sich aus einer Vignette, welche den Bildrand abdunkelt und einem Noise Filter, welcher ein weiches Rauschen über das Bild legt, zusammen. Neben den drei Einstellungen gibt es noch einen Regler mit dem sich die maximale FPS einstellen lässt. Der Bereich des dafür vorgesehenen Sliders reicht von 30-60FPS. Für die Texturen soll es ebenfalls drei

Detailstufen geben, bei denen die Auflösung der Texturen gewählt wird.

Neben all diesen Einstellungen gibt es noch die Möglichkeit den Energiesparmodus zu aktivieren. Dieser deaktiviert alle besonders Ressourcen lasstigen Effekte, so das hierfür weniger Energie benötigt wird. Zu diesen zählt wegen der vielen Datenzugriffe insbesondere der Motion Blur Filter. Zusätzlich wird die Framerate auf 24FPS nach unten gesetzt, hierdurch wird gleichzeitig die Anzahl der Drawing Calls reduziert, was auch zu einer Rechen- und damit Energieersparnis führt.

5 Projektliche Organisation(JR)

Vor allem da Drag'n Slay zu zweit angegangen wurde, werden in diesem Kapitel neben der Projektstruktur auch die verwendeten Kommunikationskanäle und Vorgehensweise näher gebracht.

5.1 Projekt Struktur (JR)

Wie in Abbildung 27 dargestellt, beinhaltet der Git Root Projekt Ordner einem Node.js Server-, Resources-, Unity3D Client- und einem Android Socket.io Test Client Ordner.

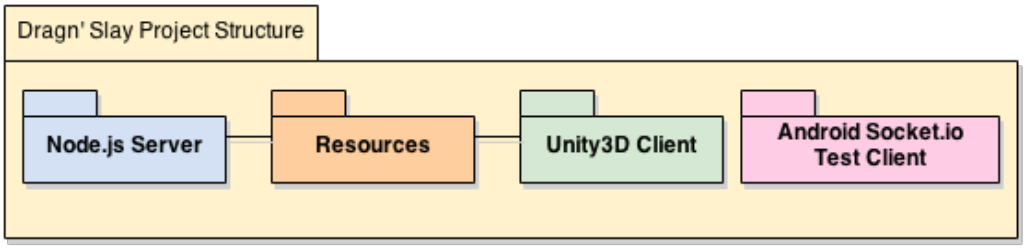


Abbildung 27: Projekt Struktur: Drag'n Slay[18]

5.1.1 Node.js Server (JR)

Der Node.js Server Ordner repräsentiert alle Server relevanten Dateien.

[18] `https://github.com/kibotu/net.kibotu.sandbox.unity.dragnslay`

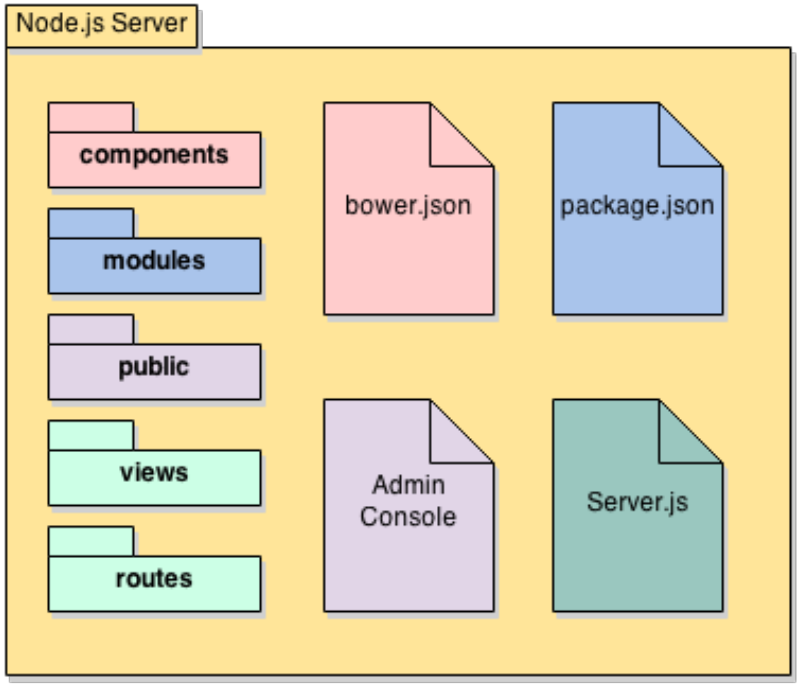


Abbildung 28: Projekt Struktur: Node.js Server[19]

Wie in Abbildung 28 dargestellt, beinhaltet dieser Server Ordner einen Komponenten, Module, Public, Views, Routes Ordner. Das Node.js Hauptprogramm befindet sich in der Server.js Datei. Für Node.js wurden zwei Package Manager verwendet. Zum einen den von Node.js mitgelieferte NPM[20], der mit dem Commandline Befehl *npm install* die package.json Datei verwendet um alle verwendeten Node.js Module herunter zu laden und diese in den Modules Ordner zu installieren. Zum anderen Bower[21], welcher Node.js Komponenten mit Hilfe des Commandline Befehl *bower install* und der bower.json Datei in den Komponenten Ordner installiert. Das hat den Vorteil, systematisch alle Fremdabhängigkeit der Module- und Komponente Versionen zu verwalten, aber auch dass diese beiden Ordner Inhalte nicht in das Git Repository mit aufgenommen werden brauchte. In dem Views Ordner befinden sich alle Jade Template Datein, die bei Internet Requests von Node.js in HTML Dateien übersetzt werden. In dem Routes Ordner befinden sich alle REST[22] Routen, wie zum Beispiel unsere Admin Console, die auf `http://localhost:1337/admin` geroutet ist. Schließlich befinden sich in dem Public Ordner alle von dem Internet abrufbare Bild-, CSS-, JS Dateien.

[19] `https://github.com/kibotu/net.kibotu.sandbox.unity.dragnslay/tree/master/server/ChatServer`
[20] `https://www.npmjs.org/`
[21] `http://bower.io/`
[22] `http://en.wikipedia.org/wiki/Representational_state_transfer`

Wichtige Module sind Express[23], Jade[24], Hat[25], Socket.io[26], Jsftp[27], Request[28], Seq[29], Os-Utils[30], Nodetime[31], Os-Monitor[32], Node-Highcharts[33] und Dgram[34].

Ebenfalls wichtige Komponenten sind Twitter Bootstrap[35], Normalize-CSS[36], jQuery[37] und Underscore[38].

5.1.2 Resources (JR)

Der Resources Ordner beinhaltet alle Assets. Dieser Ordner hat sein eigenes Git Repository und ist als Submodule eingebunden. Das hat den Vorteil, dass das Resource Repository keine Branches erstellen brauchen. Denn fast alle Assets haben nur ein Binärformat und ändern bei Änderungen meist ihre komplette binäre Anordnung. Das erschwert Git diese Dateien effektiv zu verwalten. Vor allem ist es nicht Sinnvoll mit dem Div-Tool Konflikte zu mergen.

[23] https://github.com/visionmedia/express
[24] https://github.com/visionmedia/jade
[25] https://github.com/substack/node-hat
[26] https://github.com/Automattic/socket.io
[27] https://github.com/sergi/jsftp
[28] https://www.npmjs.org/package/request
[29] https://github.com/substack/node-seq
[30] https://github.com/oscmejia/os-utils
[31] http://nodetime.com/
[32] https://github.com/lfortin/node-os-monitor
[33] https://github.com/davidpadbury/node-highcharts
[34] https://github.com/isaacs/udp
[35] https://github.com/twbs/bootstrap
[36] https://github.com/necolas/normalize.css
[37] https://github.com/jquery/jquery
[38] https://github.com/jashkenas/underscore

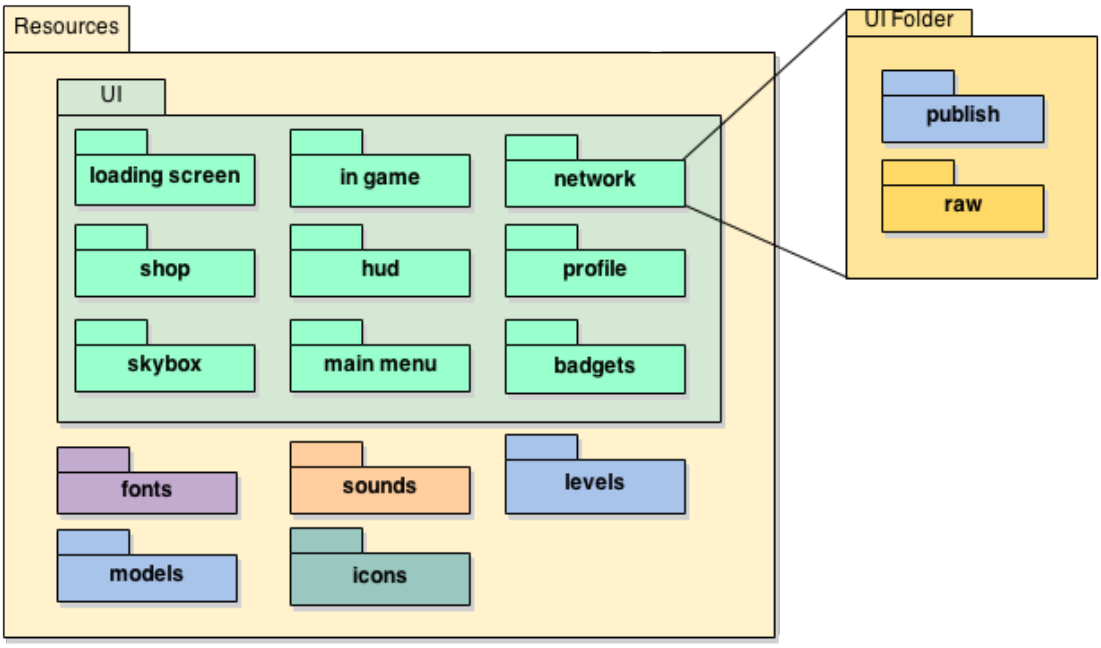


Abbildung 29: Projekt Struktur: Resources[39]

Wie in Abbildung 29 dargestellt, beinhaltet das Hauptverzeichnis Ordner für Schriftarten, Sounds, Levels, 3D Models, Icons und alle UI Elemente. Die UI Elemente bestehen immer aus einem Publish und einem Raw Ordner, wo bei in dem Raw Ordner alle einzelnen Bilddateien liegen und in dem Publish Ordner die mit Texture Packer[40] jeweils zu einem Textur Atlas zusammengefassten Datei.

5.1.3 Unity3D (JR)

Der Unity3D Projekt Ordner beinhaltet alle relevanten Dateien für den Client.

[39] `https://github.com/JanPaschen/net.kibotu.sandbox.unity.resources`
[40] `http://blog.kibotu.net/design/texture-packer`

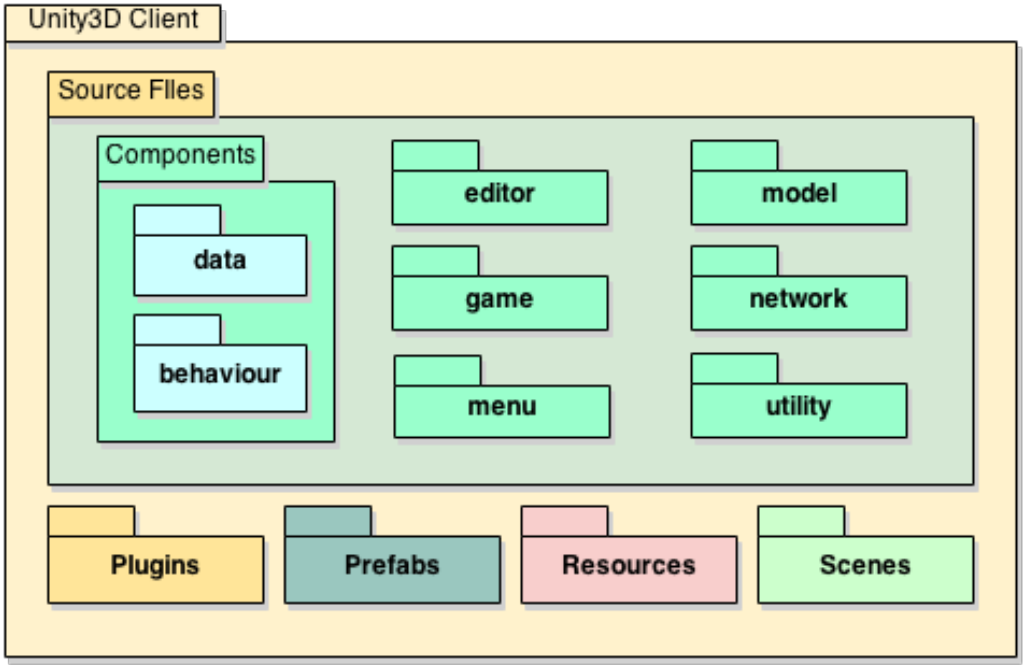


Abbildung 30: Projekt Struktur: Unity Client[41]

Wie in Abbildung 30 dargestellt, beinhalten in dem Prefabs Ordner alle vorgefertigten Muster Gameobjekte, in dem Resource Ordner alle Assets, in dem Scenes Ordner alle Szenen und schließlich in dem Source Files Ordner alle C# Quellcode Dateien. Verwendete Plugins sind, Socket.io[42], Playmaker[43], UIToolkit[44], iTween[45], und Detonator[46].

5.2 Kommunikation (JR)

Neben den wöchentlich stattfindenden Treffen, wurde sich täglich über Facebook, Skype und Emails miteinander verständigt. Für das Austauschen von Daten wurde hauptsächlich Github und Google Drive verwendet. Bei dem zusammenführen von 3D Models wurde relativ schnell klar, dass sich auf ein Format geeinigt werden muss. Vor allem die Darstellung im 3D Raum ist zwischen Blender und Unity3D unterschiedlich, da Blender OpenGL und der Unity Editor DirectX zum Rendern verwendet. Somit wurde beschlossen Blender Models mit angepassten Translations- und Rotationsvektoren, sowie mit einem Blender:Unity Skalierungsmaßstab von 10:1

[41] https://github.com/kibotu/net.kibotu.sandbox.unity.dragnslay/tree/master/unity/Assets
[42] https://github.com/NetEase/UnitySocketIO
[43] http://www.hutonggames.com/
[44] https://github.com/oddgames/UIToolkit
[45] http://itween.pixelplacement.com/index.php
[46] http://variancetheory.com/detonator/

zu exportieren. So wurden 3D Meshes schließlich mit Translationsvektoren (X,Y,-Z) und Rotations Quaternions mit (X,Y,-Z,-W) nach Unity exportiert.

Um eine Übersicht über die benötigten Sounds zu erhalten wurde im Google Drive eine Tabelle erstellt, in der alle benötigten Sounds eingetragen wurden. Wenn ein Sound fertig erstellt worden ist, wurde die Tabelle in diesem Register nach dem Ampelsystem grün gefärbt. Zusätzlich wurde in dieser Tabelle auch vermerkt, an welcher Stelle dieser Sound später eingebaut werden soll und auch Überlegungen darüber angestellt. Aber auch UI Bilder wurden in dem Resource Ordner so strukturiert, dass beide jederzeit unabhängig voneinander mit den Rohdaten und den Textur Atlanten arbeiten konnten.

In dem Projekt kamen auch neue Arbeitsweise hinzu, wie zum Beispiel das Erstellen von interpolierten Kamerafahrten und Bewegungsanimationen für das Intro. Hierbei ergab sich die Arbeitsteilung zum einen die Möglichkeit zu erschaffen diese sinnvoll zu animieren und zum anderen das tatsächliche dekorative Animieren. So hat einer iTween in das Projekt integriert und dem anderen gezeigt, wie dies zu verwenden ist. Nun konnte der andere dies verwenden, um das Intro zu erstellen.

5.3 Vorgehensweise (JR)

Hauptsächlich haben wir Trello[47] als Werkzeug für unsere Scrum[48] Arbeitsweise verwendet. Wie in Abbildung 31 dargestellt ist, wurden so Ziele definiert und Aufgaben in verschiedene Listen eingeordnet. So gab es Prioritätslisten, in denen Aufgaben nach niedriger, mittel oder hoher Priorität eingeteilt. Gerade bearbeitete Aufgaben wurden in die Doing-Liste eingeordnet und bei erfolgreichen beenden der Aufgabe in die Done-Liste verschoben. Jede Aufgabe war mindestens einer Person und einem Label zugewiesen. Das hatte den Vorteil das auf den ersten Blick erkennbar war, wer was und in welchem Arbeitsbereich zu tun hatte. Aufgabenbereiche waren Android, Unity, Server, Grafik, Sound und Thesis. Unwichtige Aufgaben wurden in die On-Hold Liste verschoben. Aufgaben waren immer für maximal eine Woche Arbeit ausgelegt. Somit waren klare Aufgaben definiert und eine flexible Arbeitsweise ohne Leerphasen möglich.

[47] `https://trello.com/`
[48] `http://en.wikipedia.org/wiki/Scrum_%28software_development%29`

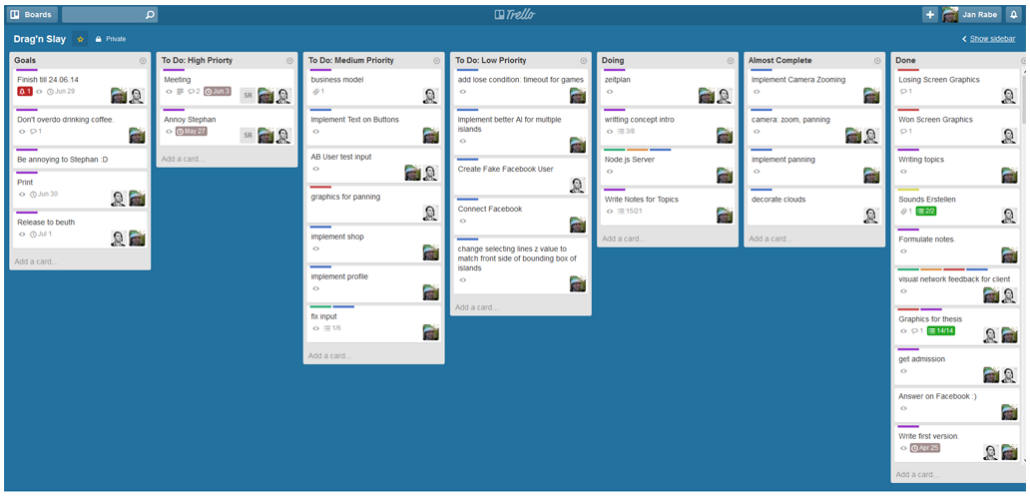


Abbildung 31: Scrum via Trello[49]

In den regelmäßigen Treffen haben wir Pair Programming[50] und Brainstormings[51] betrieben, sowie Mindmaps[52] erstellt. Besonders hilfreich waren die wöchentlichen Reviews zu bisherig geschafften Inhalten. So wurden Tasks neu priorisiert, sollte dies sinnvoll erscheinen. Das testen von Drag'n Slay auf einem Tablett und zweier Android Smartphones war ebenso bereichernd.

6 Asset Produktion (JP)

Im Folgenden Kapitel geht es um die Erstellung von 3D Modellen mit einer geringen Anzahl von Polygonen, die Texturierung dieser mit detaillierten Texturen und Fakeshadows und der Export der Modelle. Außerdem gibt es eine Erläuterung zur Erstellung von UI Objekten mit Illustrator und Photoshop.

6.1 3D Low Polygone Modelle und Fakeshadows (JP)

Bei der Entwicklung von mobilen Anwendungen geht es immer um Rechenressourcen sparendes entwickeln. Auch wenn die mobilen Endgeräte sehr schnell immer bessere Prozessoren und Grafiklösungen erhalten, sollten diese bei der Entwicklung von mobilen Spielen nicht komplett ausgelastet werden. Dagegen sprechen zum einen, dass unter Volllast die Akkulaufzeit massiv sinkt und zum anderen nicht nur die Nutzer der neusten Gerätegeneration bedient werden sollen. Insbesondere Letzteres ist im Fall von Drag'n Slay sehr wichtig, da ein Massively Multiplayer Online

[49] https://trello.com/b/xg1yqwhp/drag-n-slay
[50] http://en.wikipedia.org/wiki/Pair_programming
[51] http://en.wikipedia.org/wiki/Brainstorming
[52] http://en.wikipedia.org/wiki/Mind_map

Game (MMO) mehr von einer großen Anzahl an potentiellen Spielern profitiert als durch eine perfekten Optik.

Große Einsparungen lassen sich im Bereich der 3D Modelle erzielen. Insbesondere wenn von einem Objekt eine große Anzahl an Duplikaten dargestellt werden soll, darf dieses nicht zu detailliert modelliert sein. Da im Spiel eine möglichst hohe Anzahl an Fliegern gleichzeitig dargestellt werden soll, ist es hier besonders wichtig wenige Details zu modellieren. Als Ziel wurde ein Maximum 100 Polygone gesetzt. Der leichteste Weg Polygone zu sparen ist immer mit einer primitiven Form zu beginnen und diese nur dann zu unterteilen wenn es wirklich nötig ist.

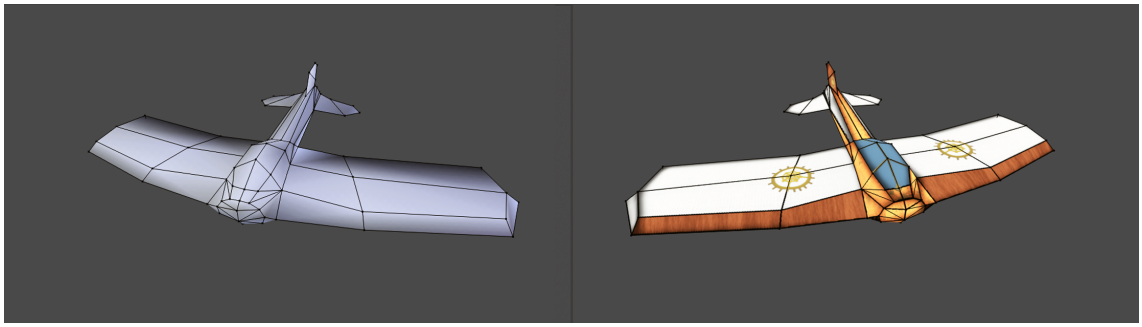


Abbildung 32: Darstellung eines Low-Poly Modells mit und ohne Textur

Dabei ist stets zu überlegen welche Details sich im Nachhinein durch Texturierung und Bumpmapping hinzufügen lassen und welche dringend nötig sind. Ein Modell mit Texturen wirkt automatisch plastischer und detaillierter. (Abb.32)

Da auf mobilen Geräten die Anzahl an Texturen welche gleichzeitig geladen werden können stark begrenzt sind, muss auch an dieser stelle überlegen werden ob auf Bumpmapping verzichtet und stattdessen mit "Fake Schatten" gearbeitet werden sollte. Da diese direkt auf der Textur gezeichnet sind bewegen sich diese bei einer Änderung des Einfallwinkels des Lichtes nicht. Auf den im Vergleich kleinen Displays von mobilen Endgeräten fällt dieses aber nicht auf. Um "Fake Schatten" zu erzeugen gibt es zwei Möglichkeiten.

Manuel: Eine Möglichkeit ist es die Schatten während der Texturierung in der Bildbearbeitungssoftware als Schlagschatten mit einzufügen. Um eine möglichst realistische Wirkung zu erzielen, sollten alle Schatten in dieselbe Richtung fallen. Werden nicht zu detaillierte Schatten benötigt ist dieses der schnellste Weg, um mit ein paar Schatten eine bessere Plastizität zu erzeugen. Diese Plastizität könnte auch durch eine Bumpmap in der zweiten Texturebene erzielt werden, es würde dann aber, durch die während der Laufzeit entstehenden Berechnungen, zu einem erhöhten Rechen- und Speicheraufwand kommen.

Baken: Der deutlich elegantere und genauere Weg ist das Baken. Hierbei wird das Objekt in der 3D Umgebung ausgeleuchtet. Entgegen der Realität sollte hierbei nicht nur eine große Lichtquelle genutzt werden, sondern mehrere kleine. Da jedes Modell anders ist gibt es keine festen Regel, wichtig ist zu beachten, dass eine symmetrische Beleuchtung des Modells entsteht. Als guter Startaufbau hat sich eine Kombination von Hemisphere, Sun-Light und einem "Shadow Only Spot" herausgestellt.

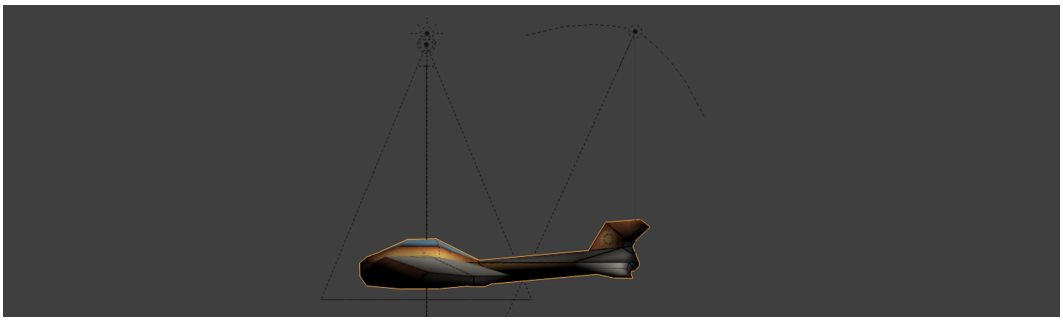


Abbildung 33: Beleuchtungsschema zum Backen von Texturen

Hierbei sorgt das Hemisphere-Light für eine gleichmäßige Grundbeleuchtung des Modells und dient gleichzeitig als Backlight. Die Sun wird als Key-Light verwendet und wird zentriert über dem Model platziert. Um einen deutlicher hervorgehobeneren Schattenwurf zu erhalten wird der "Shadow Only Spot" mit identischer Ausrichtung genau an den Koordinaten der Sun platziert. Die härte der Schatten lässt sich bei einem Spot-Light über den Fokus bestimmen.

Wird ein höherer als der so zu erreichende Detailgrad benötigt, kann auch das Low-Poly-Modell zuvor bearbeitet werden und um jegliches gewünschte Detail im Mesh zu erweitern. So können zum Beispiel im Fall des Fliegers mit einem Sculpting-Tool Nieten auf der Textur plastisch aus dem Mesh herausgearbeitet werden. So können die Nieten die zuvor nur auf der Textur waren Schatten werfen (Abb.34). Wenn jetzt noch einmal der Schattenwurf auf die Textur gebaked wird, entsteht eine Textur mit vielen detaillierten Schatten.

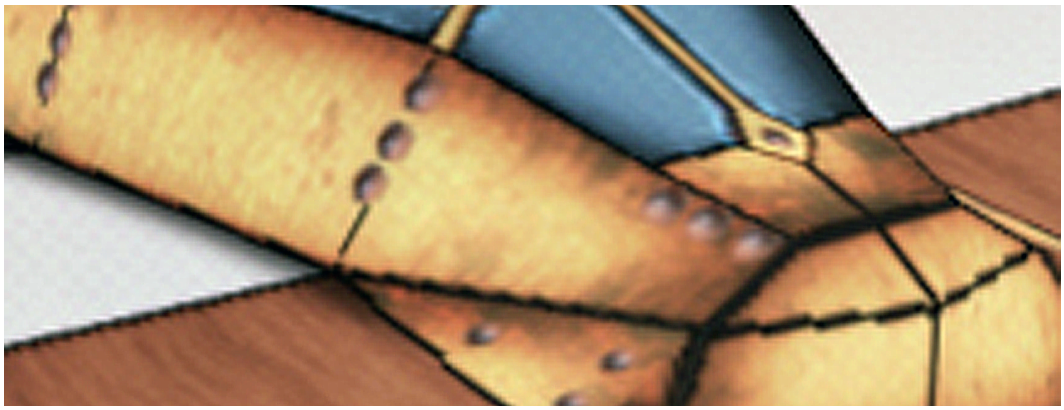


Abbildung 34: Detail Aufnahme Schatten

Das bearbeitete Modell ist nach diesem Vorgang nicht mehr Low-Poly. Da die UV-Map jedoch unverändert ist, passt die detaillierte Textur auch auf die unbearbeitet Version des Mesh.

Auf diesem Weg ist es möglich Texturen zu erstellen, welche auch bei einer hohen Zoomstuffe innerhalb des Spiels oder während Cutscenes nicht langweilig und Detail-los wirken, ohne das dabei ein höherer Rechenaufwand auf dem Endgerät entsteht (Abb: 35).

Abbildung 35: Gepolischtes Standard Flugzeug

6.1.1 Export von Blender nach Unity (JP)

Am Anfang des Projekts wurde entschieden alle Modelle aus Blender als COLLAborative Design Activity (COLLADA) zu exportieren, also im ursprünglich von Sony Computer Entertainment entwickelten .dae Format. Das COLLADA-Fileformat hat eine stabile Kompatibilität zu Unity3D und es werden beim Import alle Funktionen

unterstützt. So werden bei einem Import eines 3D Modells nicht nur das Mesh und die Textur geladen sondern auch in Blender gespeicherte Animationen so wie die Positionierung und Benennung von Bones[53]. Die COLLADA-Files haben jedoch einen Nachteil, da sie im XML-Schema designed sind, kann sich die Ladezeit insbesondere auf mobilen Endgeräten durch das Parsen der XML-Files erheblich erhöhen. Auch wenn hierdurch in unserem Projekt noch keine Probleme aufgetreten sind, wurde vorsorglich auf das Filmbox (FBX) Format gewechselt. Da dieses als Nativ lesbarer Binärcode gespeichert wird, verkürzen sich durch den Wechsel die Ladezeiten. Eine Optimierung des Workflow erbrachte sich dann noch die Erkenntnis, dass sich Blender-Projekt-Datein direkt in Unity3D importieren lassen. Hierbei wird dann automatisch eine Datei im Filmbox-Format erstellt.

Neben der Wahl des richtigen Austauschformates gab es weitere Punkte die zu beachten waren. Damit im Laufe des Projekts alles mit denselben Faktor multipliziert werden kann, ist es auch notwendig das alle Modelle im selben Größenverhältnis stehen. Hierbei wurde für das Projekt eine Baugröße 10:1 festgelegt. Dieses ergab sich aus einer angenehmen Arbeitsgröße in Blender, da bei zu kleinen Modellen beim zoomen Probleme mit der Kameraführung entstehen. Zusätzlich ist zu beachten das Blender beim Export wirklich nur das Objekt exportiert. Daher werden alle Transformationen und Translationen, die im Öbjectmode"getätigt werden ignoriert. Dieser dient nur für Transformationen innerhalb des Blender eigenen Worldspace. Insbesondere bei Rotationen kann dieses zur Verwirrung führen. Daher ist zu beachten das beim Export aus Blender alle Transformationen im "Objectmode" auf 0 stehen und auch die Skalierung auf eins normiert ist. Insbesondere bei der Ausrichtung der Modelle gibt es aber noch eine weitere Fehlerquelle, die beim Exportieren von Blender zu Unity3D zu beachten ist. Da Blender einen OpenGL Renderer nutzt und Unity3D unter Windows nativ mit DirectX arbeitet, kommt es zu einer Umkehrung in der Ausrichtung der Z-Achse im Worldspace. Das lässt sich darauf zurück führen, dass die Koordinaten einmal in einem Linkshändigen und einmal in einem Rechtshändigen Koordinatensystem berechnet werden. Um dieses Problem zu lösen, wurden in Drag'n Slay Konvention festgelegt, dass alle Modelle beim Import in Unity3D rotiert werden, da so die Arbeit an den Modellen in Blender nicht beeinträchtigt werden.

[53] http://unity3d.com/unity/workflow/asset-workflow

6.2 UI Design (JP)

Unter dem User Interface Design (UI Design) wird die Gestaltung der Oberfläche zur Mensch-Maschine-Kommunikation verstanden. Das Ziel, dass dieser Disziplin des Designs zugrunde liegt, ist das Gestalten eines Graphical User Interface (GUI), welches möglichst vielen Benutzern eine intuitive Bedienung ermöglicht.

Besonderheiten bei mobilen Endgeräten insbesondere mit Multi-Touch-Oberflächen: Auf Anordnung achten, da Teile durch Handlungen verdeckt werden. Daher sollten nicht interaktive Elemente immer oben angeordnet werden.

Kleine Bildschirmgröße darf aus Gründen der Übersicht nicht überladen werden.

Da Mobile Endgeräte auch im Freien benutzt werden, sollten hohe Kontrastwerte genutzt werden, um der unvermeidbaren Sonneneinstrahlung und damit schlechter Erkennbarkeit auf dem Display entgegen zu wirken.

Endgeräte haben bis zu 4 mal so große pixels per inch (ppi) (Aktuelle Smartphones liegen bei 360ppi ein Computer Monitor mit Full-HD Auflösung kommt hingegen nur auf ca 140ppi) Damit sind die Elemente auf dem Display des Mobilendevices im Verhältnis nur circa 1/4 so groß wie sie auf dem PC Bildschirm erscheinen.

Lösung: "responsive Desig" Multiplikation der Elemente Größe anhand eines Faktors in Abhängigkeit zur Geräteauflösung.

Die richtige Typografie an der richtigen Stelle! Auch die Schriftauflösung mit anpassen und dabei auch beachten wie weit man das Gerät von den Augen weg hält.

6.3 UI Elemente (JP)

Unsere UI-Elemente unterscheiden sich in zwei Bereichen. Zum einen gibt es in Drag'n Slay die UI-Elemente, die nur zum Zieren und Navigieren in Menüs und Shop existieren und dem Game-HUD, der eine Interaktive Bedien- und Informationsfläche während des Spiels ist.

6.3.1 Buttons (JP)

Das Buttondesign ist an die Grundthematik des Steampunk-Universums angepasst. So finden sich an allen Buttons viele kleine interaktive Details. Es drehen sich Zahnräder und aus kleinen Öffnungen tritt dunkler rußiger Dampf aus. Da das Steampunk Universum doch eher eine düstere Optik besitzt, ist es eine Herausforderung die Buttons so zu designen, dass auf ihnen gut lesbare Typografien untergebracht werden können, da bei überwiegend dunklen Materialien schnell der

Kontrast verloren geht. Vor dem eigentlichen Design der Buttons wurden zuerst die allgemeinen Richtlinien für UI Elementen auf Mobilen Endgeräten betrachtet.

Nach Google Guidlines hat das Ideale Touchitem eine Auflösung von mindestens 48 density-independent pixel (dp) Seitenlänge. Hierbei wird die Maßeinheit dp genutzt, welche unabhängig von der ppi und der damit verbundenen Pixeldichte ist. Ein dp entspricht circa einem Pixel bei einer ppi von 160. Damit würde man je nach ppi des Displays auf eine Seitenlänge von 7-11mm kommen. Diese Größe ist auch zwingend notwendig, da jedes Element, dass auf dem Bildschirm kleiner dargestellt wird von dem in der Zielgruppe definiertem User, nicht mehr mit hundertprozentiger Sicherheit mit dem Finger getroffen werden kann.

Desweiteren wird empfohlen zwischen den einzelnen Buttons mindestens ein Spacing von 8dp einzuhalten, da kleinere Werte auch hier die Treffsicherheit des Users negativ beeinflussen würden. Auch wenn es sich bei Drag'n Slay um ein Spiel und nicht um eine klassische App handelt, sind diese Erkenntnisse für das Layout durchaus Richtungsweisend, da die physikalischen Beschränkungen auch hier zählen. Auch das Beachten der inzwischen Typischen "Touch Feedback States" ist für das UI eines Spiels nicht zwingend erforderlich, hilft aber dem User bei der Orientierung.

Um erste Ideen für das Design zu sammeln, wurde mit der Hand skizziert. Das bietet den Vorteil, dass sehr schnell eine Designidee festgehalten werden kann und diese mit dem Team geteilt werden kann ohne diese vorher am Computer visualisieren zu müssen. Nach mehreren gezeichneten entwürfen, wurden diese Genutzt um mit Hilfe eines A/B Tests das beliebteste Design zu ermitteln.

6.3.2 Vektorgrafiken mit Illustrator (JP)

Nachdem sich für ein Design entschieden wurde, was nicht heißt, dass dieses bis zum Ende bestehen bleibt, kann man anfangen das Bild in Adobe Illustrator oder einem anderen Vektor basiertem Grafikprogramm zu vektorisieren. Durch die Vektorisierung entsteht der Vorteil, dass man nicht an eine Pixel gebundene Auflösung der Grafik gebunden ist und daher immer ohne Qualitätsverlust skalieren, transformieren und Rotieren kann. Am Anfang gibt es zwei Möglichkeiten. Entweder scannt man die Skizze ein und zieht auf diesem die Linien mit Vektorpfaden nach oder zeichnet den Entwurf Freihand aus Vektorpfaden nach. Da ich mich für letztere Möglichkeit entschieden habe, begann ich damit mit einem Rechteck die Außenmaße aufzuspannen, um im richtigen Seitenverhältnis zu bleiben. Danach habe ich mit den großen Hauptelementen der Buttons angefangen. Das geht am leichtesten wenn man die Grundform erst grob aus primitiven Formen nachbaut und diese dann mit dem

Pathfinder-Tool gemerged wird. Nachdem so die Grundform für den Button entstanden ist, wurden immer kleinere Details herausgearbeitet. Zu diesem Zeitpunkt sind sämtliche Flächen des Buttons nur mit ihren späteren Grundfarben oder einem Farbverlauf gefüllt. Texturen und Lichteffekte werden im nächsten Schritt mit Hilfe von Adobe Photoshop hinzugefügt.

6.3.3 Postproduction mit Photoshop (JP)

Mit Hilfe des Exchange-Tools Adobe Bridge, wird der Button als Smartobject zum weiterarbeiten in Photoshop transferiert. Durch das Smartobject entsteht der Vorteil, dass sich die Dateiinformationen des Bildes immer noch in der Vektor basierten Illustrator Datei befinden und dadurch Speicherplatz eingespart wird. Ein weiterer Vorteil ist, dass sich bei nachträglichen Änderungen in Illustrator auch der Button in Photoshop anpasst. Bei dieser Arbeitsweise ist es nur wichtig die einzelnen Ebenen als Masken zu nutzen und keine neuen Masken zu erstellen, da sich nur diese automatisch mit anpassen. wird dieses genutzt, können die Ebenenmasken als "dynamische Stempel" für unsere Texturen verwenden. Damit sich der Button gut in das Steampunk Universum einpasst, wird er an dieser Stelle also mit Holz und Kupfer Texturen versehen. Damit das Ganze noch etwas interessanter wirkt, wird über den gesamten Button eine Textur mit ein paar Kratzern gelegt, welche die Oberfläche nicht so glatt und künstlich wirken lassen. Aus Photoshop kann der Button dann als Pixelgrafik gespeichert werden. Da eine Freistellung der Grafiken, also ein transparenter Hintergrund benötigt, haben wir uns für das .png Format entschieden. Hier lässt sich ein Bild mit freien Alphawerten speichern was bei einem .jpg nicht möglich ist. Dieses verfügt über keine Transparenz-Ebene. Ein .tif würde über eine Transparenz-Ebene verfügen, jedoch würde hier das gering komprimierte Dateiformat die Speicherkapazität sprengen. Aus diesen Gründen ist das png-Format hier vorzuziehen.

6.3.4 Design Regeln (JP)

Auch bei einem Design für ein Mobiles Endgerät sollte die UI Regel der "Magic Seven", welche von der Millersche Zahl abgeleitet ist, beachtet werden. Diese besagt das auf einem Bildschirm zur gleichen Zeit nicht mehr als 7 +/-2 Steuerungselemente sichtbar sein sollten, da der Benutzer sonst schnell überfordert sein kann und die Übersicht verliert, da er die einzelnen Chunks dann nicht mehr im Kurzzeitgedächtnis speichern kann. Bei mobilen Endgeräten sollte man auf Grund des Platzmangels eher von minus zwei Elementen ausgehen, da sich kleine Elemente schlechter einprägen.[54]Da man auf diese Weise in der Anzahl der Benutzbaren Ele-

mente stark eingeschenkt ist, kann es hilfreich sein Funktionen auf Touchgesten auszulagern, anstatt diese auf zusätzliche Buttons zu legen. Ein gutes Beispiel hierfür ist die genutzte "pinch to zoom" Fingergeste. So konnten gleich zwei Buttons im "in Game UI" eingespart werden. Da diese Gesten nicht im Kurzzeitgedächtnis gespeichert werden, sondern zuvor gelernt werden, fallen diese nicht unter die Regel der Magic Seven.

6.3.5 App Icon (JP)

Das App-Icon ist das erste Interaktionsobjekt zwischen dem Spieler und der Anwendung und das bereits vor der Installation. Da das Icon bereits im Playstore angezeigt wird, kann es entscheidend für die Installation der Anwendung sein. Deswegen ist Gestaltung des App-Icon enorm wichtig. Es muss zum einen so auffällig sein, dass es im Store aus den Massen an anderen Apps heraussticht und zur gleichen Zeit eine visuelle Verbindung zur Anwendung herstellt. An dieser Stelle bietet es sich an eine kleine Recherche zu beginnen. Wird im Store nach Apps, welche der eigenen ähneln gesucht, wird man meist sehr schnell fündig und stellt fest, das viele Icons sich sehr ähnlich sehen. So kann es schon reichen, einfach eine andere Farbpalette zu wählen oder das Hintergrundbild anders zu gestalten um die Aufmerksamkeit auf die eigene Anwendung zu ziehen. Da ein Großteil der Stores in Graustufen designet ist, um nicht von den Produkten abzulenken, sollten Graustufen im App-Icon vermeiden werden, da diese sich sonst in den Shop integrieren und sich nicht von ihm hervorheben.

Um eine Verbindung zwischen dem Icon und der Anwendung herzustellen, bietet es sich an die gleiche Farbpalette wie für die Anwendung zu nutzen oder auch kleine grafische Elemente aus dem UI zu übernehmen. Da die Auflösung für das Icon auf den meisten mobilen Endgeräten aber sehr klein ist, sollte darauf geachtet werden mit klaren Linien und nicht zu vielen Details zu arbeiten, da diese sonst bei einer geringen Auflösung verschwimmen. Für eine höher wirkende Schärfe ist es auch hilfreich mit stark kontrastierenden Farben zu arbeiten.

Möchte man eine Anwendung im Android Play-Store und im Apple App-Store vertreiben und dabei zwecks Wiedererkennungswert ein Identisches Icon nutzen, sollte man von Anfang an mit "abgerundeten Ecken" Designen, da auf manchen IOS Systemen die Ecken der App-Icons automatisch abgerundet (beschnitten) werden.

Bei Betrachtet der System-Maße von Android und IOS stellt man leider fest das die Auflösungen für dasselbe Einsatzgebiet unterschiedlich sind. So betragen die

[54]Quelle: Virtual Reality - Human Computer Interaction, 978-953-51-0721-7

Auflösungen des Icons im Launcher bei Android basierten Geräten mit 160dpi aufgelöstem Bildschirm 48dp und beim iPhone ca 57dp. Bei der Auflösung für die Shop-Preview unterscheiden sich beide Systeme noch mehr. Während der App-Store mindestens eine Auflösung von 1024x1024 Pixeln verlangt, reichen im Play-Store bereits Grafiken mit einer auflösung von 512x512 Pixeln.

Zuerst wurde versucht dieses Problem mit einer selbstgeschriebenen Photoshop Aktion zu automatisieren, da verhindert werden sollte die Grafiken mehrfach von Hand in den entsprechenden Größen exportieren zu müssen. Nachdem das Icon zum ersten Mal in das Unity Projekt eingefügt wurde, zeigte sich aber, dass Unity eine eigene Funktion besitzt, welche das Icon beim Export automatisch auf die vom OS vorgeschriebenen Maße bringt.

Nach der Installation trifft der Kunde zum zweiten mal auf das App-Icon. Ein gutes Zeichen ist, wenn es ihm im Launcher sofort wieder ins Auge fällt und er es nicht suchen muss. Ein App-Icon ist also um einiges wichtiger und sollte durchdachter sein, als im ersten Moment vorstellbar.

Aus diesem Grund sollte der Designer an dieser Stelle auch nicht nur auf sein eigenes Gespür vertrauen, sondern mit Hilfe eines A/B Tests den für eine Vielzahl von potenziellen Spielern am interessantesten wirkenden Entwurf verwirklichen. Um nicht unzählige Icons Visualisieren zu müssen, können auch Skizzen wie in Abbildung 36 zu Testzwecken angefertigt werden. Diese müssend nicht perfekt sein, sollten aber die dahinter liegende Idee deutlich zeigen. Zur leichteren Vorstellung ist es für die befragten Personen leichter, wenn das Icon in ein Mockup eingefügt wird, da es so auch im Vergleich zu anderen Icons in der gewohnten Umgebung steht.

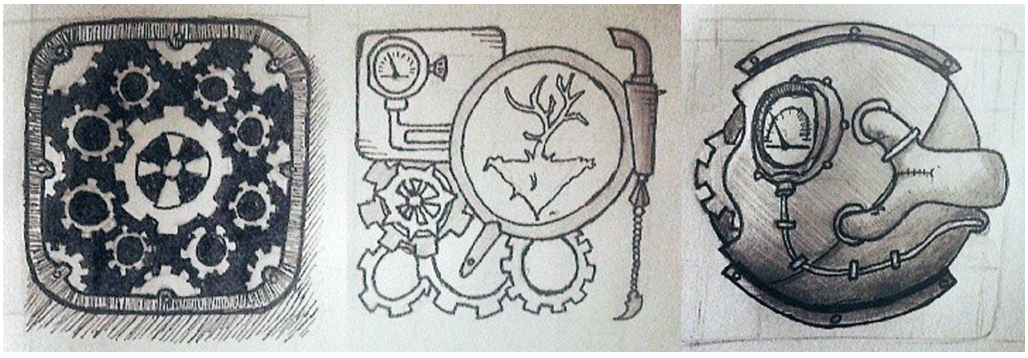


Abbildung 36: Drei Icon Entwürfe

Bei unserem A/B Test hat sich das in Abbildung 36 mittlere Icon als beliebtestes durchgesetzt. Die beiden anderen wurden als zu düster und weniger einladend beschrieben. Auch wenn die düstere Ausstrahlung gut ins Steampunk Universum

passen würde, ist uns eine Massentauglichkeit von Drag'n Slay wichtiger als ein Hundertprozentige Widerspiegelung des Steampunk Universums. Das Endresultat zeigt Abbildung 37.

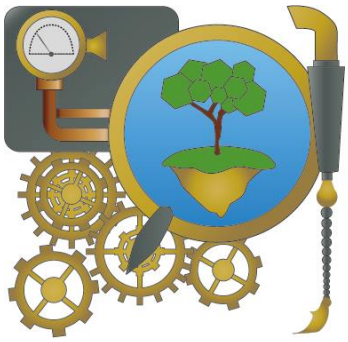


Abbildung 37: Finales App Icon

6.3.6 Loading Screen (JP)

Hier entsteht nach dem Icon für die App eine zweite Chance für den ersten Eindruck. Deswegen ist das Design des Loading Screen sehr wichtig und sollte den Spieler bereits in die Thematik des Spiels leiten. Das Design sollte so angelegt werden, dass es dem User Signalisieren kann, dass etwas im Hintergrund etwas passiert und das System nicht stehen geblieben ist. Es sollte also wenigstens ein sich bewegendes Element haben. Als sinnvoll kann sich auch eine Progress-Bar erweisen, die den Fortschritt im Detail anzeigt.

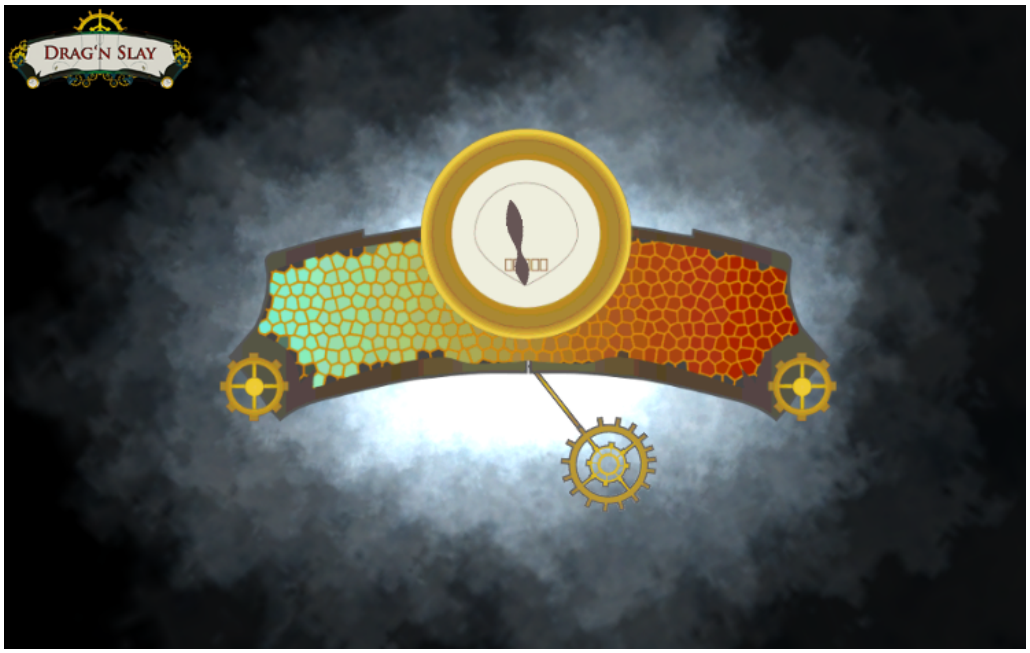


Abbildung 38: Loading Screen

Der Drag'n Slay Loading Screen in Abbildung 38 besitzt neben dem Hauptelement drei animierte Objekte. Ein oszillierendes Pendel, eine Partikel Animation im Hintergrund und einen Druckzeiger, welcher den Ladestatus anzeigt.

6.3.7 Game-HUD (JP)

Bei der Erstellung eines Game-HUDs für ein mobiles Endgerät müssen ein paar wichtige Dinge beachtet werden. Am wichtigsten ist es, dass Bedienelemente immer unten und Informationselemente immer oben angeordnet werden. Würde man dieses nicht beachten, würden Informationselemente beim Nutzen der Bedienelemente durch die Finger des Spielers verdeckt werden. Wie in Abbildung 39 zu sehen, sind diese Regel streng befolgt worden.

Abbildung 39: Game HUD

Momentan nur auf Rechtshänder ausgelegt, befinden sich die drei Buttons für die Skills in der unteren rechten Ecke und die Informationselemente, wie Spieldauer und die Minimap oben. Nur für den Menü-Button wurde diese Regel absichtlich gebrochen und er wurde in die obere linke Ecke gelegt, da die Wahrscheinlichkeit, dass er an dieser Stelle versehentlich betätigt wird am geringsten ist.

Bei der Gestaltung des HUDs wurden absichtlich einzelne nicht gekoppelte Elemente gewählt, da sich diese leichter platzieren lassen, ohne die Skalierung bei unterschiedlichen Seitenverhältnissen zu verlieren. Über das dabei genutzte UI-Toolkit lassen sich die Elemente so an prozentual beschriebenen Positionen, welche sich auf fixe Positionen des Displays beziehen platzieren. Bei der Positionsbeschreibung mit dem

UI-Toolkit übergibt man der Funktion zunächst eine Position auf dem Display, im Falle der Skillbar zum Beispiel "bottom-right". Das Abziehen der Länge und der Höhe des Elementes erfolgt automatisch, ist aber notwendig, da der Nullpunkt der Grafiken bei Unity immer oben links ist. Würden die Dimensionen des UI-Elements nicht abgezogen werden, würde es außerhalb des Bildschirms landen. Möchte man ein Objekt nicht direkt an der Kannte haben, lässt sich dieses auch Prozentual in positive und negative X und Y Richtung verschieben.

Um das Spiel lebendiger wirken zu lassen, sollen auch einige Elemente im HUD animiert sein. So wird sich zum Beispiel in Zukunft das Zahnrad oben mittig dauerhaft drehen und das hydraulische Element an der Skillbar soll sich beim benutzen von Skills bewegen. Die dafür benötigten Techniken haben wir bereits beim animieren des Loadingscreens erfolgreich genutzt.

7 Technische Implementation (JR)

In diesem Kapitel wird beschrieben, wie die Animationen in Drag'n Slay gelöst wurden und wie der Multiplayer implementiert wurde. Außerdem werden Testgeräte aufgezählt und welche Optimierung geplant sind.

7.1 Animations (JP)

In Drag'n Slay werden unterschiedliche Arten von Animationen verwendet. Hierbei reicht die Spanne von einfachen Easing Animationen im 2D Bereich über Blender timeline Animationen und Path Animations bis zu script gesteuerten Animation. Fast alle Animationen werden durch zusätzlich animierte Partikel unterstützt, da so die Bewegungen auf dem weitläufigem Himmel besser zur Geltung kommen. Bereits in der Introszene werden diese vier Animations-Arten genutzt, um eine lebendige Welt zu gestalten. Flieger werden mittels Path-Animations durch die Welt navigiert. Dabei kommen sie an fliegenden Inseln vorbei, welche sich in einem beschränktem Rahmen zufällig neigen. Auf manchen der Inseln fließen Partikelwasserfälle in den endlosen Himmeln und auf anderen stehen animierte Windmühlen.

Um die Rotation der Flügel zu implementieren, wurde die Blender eigene Animations-Engine genutzt. Bei dieser handelt es sich um ein "time based Animation" Das heiß,t dass für jedes Element eine Translation, Transformation oder Rotation zu einem bestimmten Zeitpunkt hinzugefügt werden kann. Bei der in Abbildung 40 zu sehende Windmühle ist die Beschreibung der Animation sehr simpel.

Abbildung 40: Windmühle mit Blender Animation

Für die Rotation werden die Flügel der Mühle auf der Z-Achse im Koordinatensystem der Mühle selbst, über einen Zeitraum von drei Sekunden um 360 Grad rotiert. Hierbei muss eine lineare Interpolation der Bewegung gewählt werden, da es sonst im loop der Animation keinen flüssigen Übergang auf den ersten Frame geben würde.

Zur Animation der Flieger im Intro wurde ein Pathfinding-Script verwendet, welches ein Array mit Punkten im 3D Raum übergeben bekommt. Des weiteren wird noch eine Zeit übergeben, in der gesagt wird nach welcher Zeit das Objekt das Ende des Pfades erreichen soll und nach welchem Schema die Interpolation zwischen den Punkten erfolgt. Neben dem Flug der Flugzeuge wird auch die Kamerafahrt mit diesem Script umgesetzt. Damit die Kamera den in Szene gesetzten Flieger im Sichtfeld behält, ist diese zusätzlich mit einer tracing Funktion ausgestattet, welche dafür sorgt das der Flieger immer im Mittelpunkt bleibt. Damit die Ausrichtung der Kamera nicht unnatürlich wirkt, sind alle Schwenkbewegungen kubisch interpoliert. Die Abbildung 41 zeigt die Szene aus der Draufsicht des Unity 3D Editors mit eingezeichneten Pfaden der Flieger. So entsteht eine bessere Vorstellung für den Ablauf der Szene. Der rote Pfeil zeigt den Pfad der Kamera, die gelben die Wege der langsam fliegenden Schiffe und die beiden grünen Pfeile zeigen die Strecke für die beiden Flieger die im Mittelpunkt der Szene stehen.

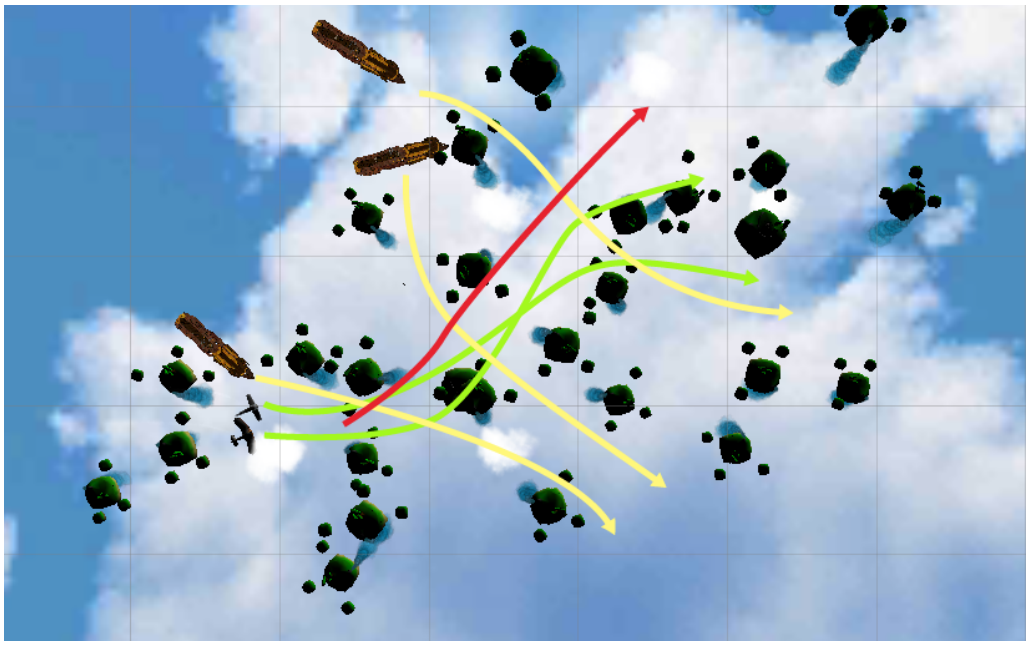


Abbildung 41: Top ansicht der Intro Szene im Editor

Um die Szene noch lebendiger wirken zu lassen, fliegen noch Partikel-Wolken durch den Himmel und die Sonne schmeißt helle Lichtstrahlen durch den Himmel, welche durch die Inseln und Flieger gebrochen werden. Das fertige Ergebnis sieht man in Abbildung 42. Obwohl es sich nur um einen Screenshot einer auf Bewegung ausgelegten Szene handelt, wirkt diese sehr dynamisch.

Abbildung 42: Screenshot der fertigen Introszene

Nach einem Klick oder dem Ablauf von 20 Sekunden öffnet sich das Hauptmenü über dem Intro, während dieses im Hintergrund weiterhin zu sehen ist. Das Menü faded zunächst über den Alphawert ein und skaliert sich danach langsam etwas größer. Durch diese Skalierung in Verbindung mit der Introszene im Hintergrund, entsteht

ein optischer Eindruck, die dem des Vertigo-Effekts aus der Kameratechnik ähnelt.

7.2 Multiplayer (JR)

Um dem Spieler ein konsistentes Multiplayer-Erlebnis zu bieten, gibt es folgende Voraussetzungen. Dazu ist die Überlegung notwendig zu überlegen, wie viel Spieler gleichzeitig spielen können, welche Daten dynamisch auf dem Server verwaltet werden, wie groß sind diese Datenpakete und wie wird niedrigen Ping und niedrigen FPS umgehen.

7.2.1 Spieler Fragmentierung (JR)

Spieler Fragmentierung entsteht immer genau dann, wenn diese in unterschiedlichen Spielmodi spielen, in unterschiedlichen Zeitzonen leben oder wegen zu hohen Skillunterschieden nicht miteinander spielen können.

Warteschlangen in Multiplayer Spielen können für Spieler sehr frustrierend sein, vor allem da es bei niedrigen Spielerzahlen relativ oft vorkommt, dass sich ein Spieler einloggt, zwei Minuten wartet, ausloggt, sich in dem Moment ein anderer Spieler einloggt, auch zwei Minuten wartet nur um sich dann ebenfalls genervt auszuloggen.

Damit dies nicht passiert, sollte vorher geklärt werden, wie viele Spieler durchschnittlich mindestens am Tag benötigen werden, so dass immer mindestens ein Spiel gematched werden kann.

Dazu wurden zehn Leute gefragt, wie lange sie in einer Warteschlange eines Mobilen Spieles warten, bevor sie es beenden.

Abbildung 43: Spielertest[55]

Wie die Abbildung 43 erkennen lässt, liegt die Toleranzgrenze für Wartezeiten schreckend gering. Zwar ist die Testgruppe mit zehn Personen nicht besonders groß, jedoch geht ein klarer Trend hervor. Die Wartezeit sollte so gering wie möglich sein. Hier sogar unbedingt unter 30 Sekunden, da sonst User verloren gehen. Interessant hierbei ist auch, dass die Toleranzgrenze sich erhöht, sollte das Spiel bereits schon einmal gespielt worden sein.

Es wird davon ausgegangen, dass eine Partie in Drag'n Slay durchschnittlich etwa fünf Minuten dauert. Damit ergibt sich folgende Formel:

$$SpielerProTag = \frac{Tag}{WartezeitProSpiel} - AnzahlSpieleProSpieler * SpielerProTag \quad (2)$$

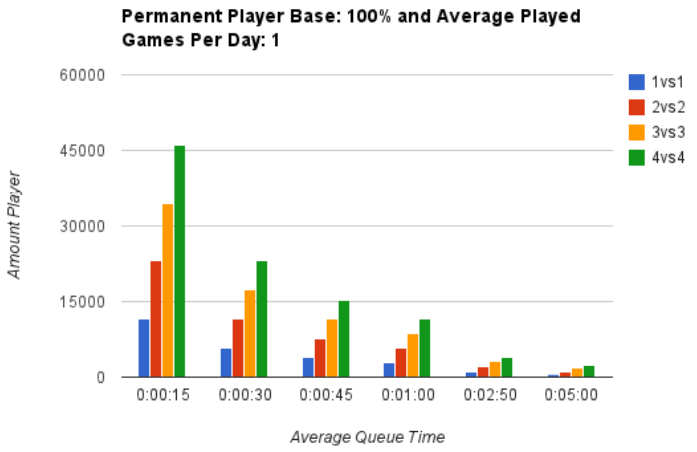


Abbildung 44: Mindestanzahl Spieler mit Wartezeit[55]

Die in Abbildung 44 zeigt, dass in einer perfekten Welt bei fünf Minuten Spieldauer 5760 Spieler am Tag benötigt, damit mindestens immer ein Spieler innerhalb von 30 Sekunden Wartezeit gematched werden kann. Natürlich verteilen sich online Zeiten in einer realen Welt nicht perfekt. Sondern es gibt Spitzenzeiten mit niedriger und hoher Aktivität.

Für Drag'n Slay Zwecke sind nur die niedrigen Spitzenwerte interessant, da bei zu wenig Spielern keine Spiele gematched werden könnten. Bei zu hoher Aktivität müsste im idealen Fall lediglich die Bandbreite, Server CPU- und Ramgrößen skaliert

[55] `http://goo.gl/N6yXQz`

werden. Natürlich ist dies naiv, da dort ebenfalls komplexe Probleme entstehen, die für den Moment vernachlässigt werden.

Wie groß müsste dann die Spielerbasis sein, um niedrige Spitzenwerte zu kompensieren? ’

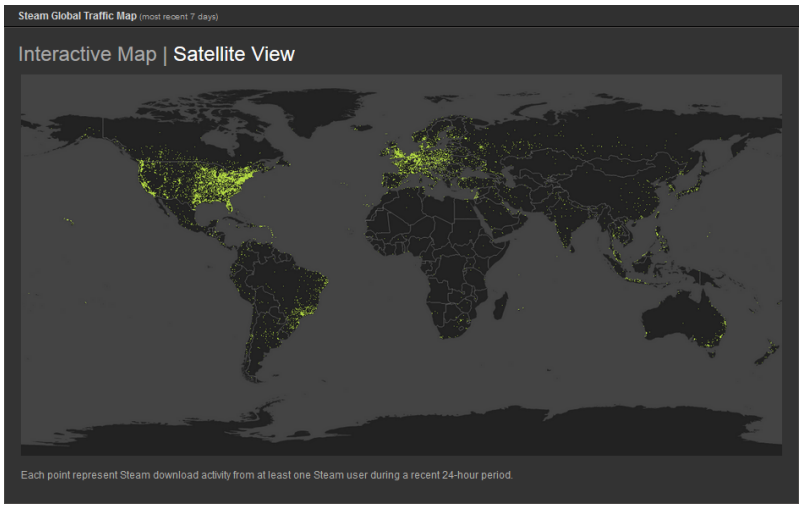


Abbildung 45: Globale Spieler Fragmentierung Steam (June 2014, UTC -7)[56]

Hierfür wurden Steam’s öffentliche Login Statistiken zur Hilfe genommen, um uns einen Überblick zu verschaffen. Natürlich deckt Steam nicht nur den mobilen Markt ab, jedoch, wie man in Abbildung 45 gut sehen kann, nutzen die USA und Europa Steam am meisten, welches die Initialen Zielgruppenländer perfekt abdecken.

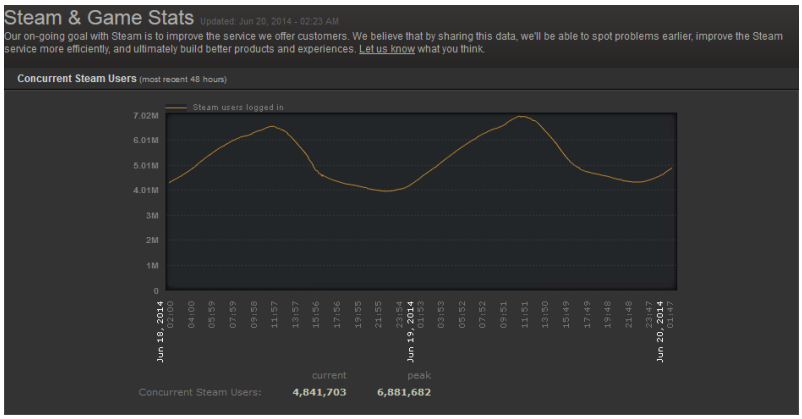


Abbildung 46: Zeitliche Fragmentierung Steam (Juny 2014, UTC -7)[57]

Wie in der Abbildung 46 zu sehen ist, loggen sich etwa 7.1 Millionen Spieler am Tag zu Steam ein. Die niedrigste Aktivität liegt zwischen 16 bis 4 Uhr UTC -7 mit

[56] Quelle: `http://store.steampowered.com/stats/content/`
[57] Quelle: `http://store.steampowered.com/stats/`

3.9 - 5.1 Millionen Spielern, was umgerechnet zwischen 18 bis 6 Uhr nach deutscher Sommerzeit mit UTC +2 liegt. Somit sind mindestens immer etwa 55 Prozent der kompletten Spielerbasis eingeloggt. Die hohe Mindestauslastung ist auf die hohe Zeitzonenabdeckung zurück zu führen.

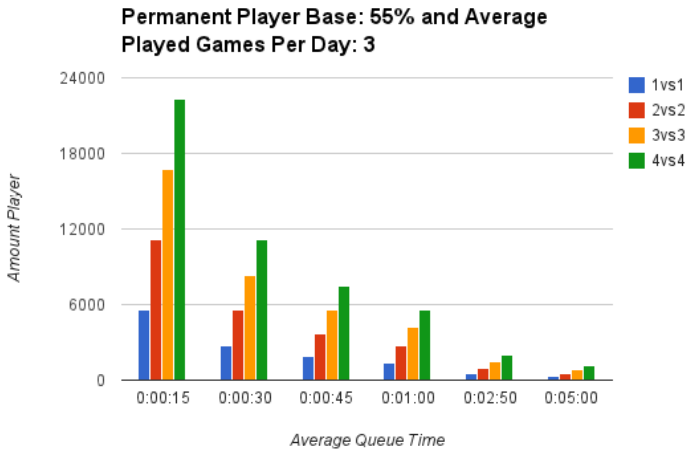


Abbildung 47: Mindestanzahl Spieler vs Wartezeit mit 55 Prozentiger Fragmentierung[55]

Von ungetester Selbsteinschätzung werden im Durchschnitt etwa drei Spiele am Tag pro Spieler gespielt. Daraus ergibt sich die Grafik in Abbildung 47. Es werden somit am Tag mindestens 8352 Spieler benötigt, damit ein Spieler immer mindestens einem neuem Spiel gematched werden kann, um ebenfalls zu kompensieren, dass sich nur 55 Prozent der Spielerbasis einloggen.

Verstärkt wird dies noch zusätzlich durch mehrere Spielmodi und die Skillfragmentierung des Ranked Systems. Somit würde Drag'n Slay bei allen vier Spielmodi und allen fünf Ranked Stufen bei einer Wartezeit von 30 Sekunden etwa 139200 Spieler benötigen.

Fazit

Die hohe Anforderung an die Anzahl von Spielern, bewog dazu in Drag'n Slay Spieler gegen den Computer zu matchen, sollte die Wartezeit über 30 Sekunden betragen. Außerdem würde der Einfachheithalber entscheiden den Prototypen erst einmal nur den 1vs1 Spielmodus zu unterstützen.

7.2.2 Server (JR)

Änderungen sollen flexible vernehmbar sein, wie zum Beispiel Preisänderungen im Shop oder Balancing der Levels. Dazu ist zu erwähnen, dass es relativ zeitaufwendig

ist eine App, die im Google Play Store oder im Apple App Store relased wurde, zu aktualisieren. Grund ist einerseits die Synchronisationsverzögerung in der Cloud und andererseits der Apple's App Store Review Guideline[58]. Außerdem sollte es eine zentrale Anlaufstelle für alle Clients aller unterstützen Plattformen gehen. Zusätzlich sollten die Clients nicht direkt, sondern nur indirekt über den Server miteinander kommunizieren, um serverseitig Konsistenzprobleme regeln zu können.

Dies alles und akademische Neugier haben dazu beitragen, nicht die mitgelieferte proprietäre Mutplyplayer Komponente von Unity3D, sondern als Backend Server Node.js[59] mit Socket.io[60] zu verwenden. Socket.io ist eine Cross-Platform Websockets Bibliothek, welche eine gleiche API für alle Clients mitbringt. Ein weiterer Grund liegt an Node.js verwendeten Sprache Javascript, die es extrem erleichtert ein zusätzliches Monitoring Tool als Hilfsmittel zu erstellen.

7.2.3 Node.js Server (JR)

Generell verwaltet der Server Spieler und Spiele, genau so wie ein Chat-Server.

```
[
  'f3faac27c45ca9a6d0f312f7bb3da2h7': {
    user: [
      'bdc4d657302e4462c33cc4dfa82c6b62}',
      'd2faac27c45ca9a6d0f312f7bb3da203'
    ],
    bots: [
      '447b54136728ad1f880f2ddd7681aa2a'
    ],
    game: {
      game_type: 'game1vs1',
      data: {
        'island-type': [Object],
        'ship-type': [Object],
        world: [Object],
        players: [Object],
        'host-uid': 'bdc4d657302e4462c33cc4dfa82c6b62'
      },
      running: true
    }
  }
]
```

[58] https://developer.apple.com/appstore/resources/approval/guidelines.html
[59] http://nodejs.org/
[60] http://socket.io/

Abbildung 61: Spielräume [61]

Er verwaltet Spieler in Chat Räumen, um so deren Kommunikation miteinander zu steuern. Ein Beispiel eines 1vs1 Spieles ist in Abbildung 61 dargestellt. Der Raum *f3faac27c45ca-9a6d0f312f7bb3da2h7* beinhaltet in Zeile drei bis sechs die Unique Ids aller teilnehmenden Spieler und Bots. Die Zeile 19, ob das Spiel aktiv ist. Außerdem verwaltet der Server den aktuellen Game State eines Spieles in Zeile zehn bis 18, wodurch bei Konsistenzfragen logischerweise immer der Server Recht hat.

Interesant hierbei ist auch zu erwähnen, dass in Drag'n Slay lokale IP Adressen dynamisch verwenden. Drag' Slay wird an mindestens vier unterschiedlichen Orten entwickelt. Unter normalen Umständen müsste mindestens einmal am Tag die IP Adresse auf allen Testgeräten geändert werden, da diese sich einmal am Tag seitens des Internet Provider her ändert. Daher wird mit Node.js auf einem Webserver eine Datei mit den aktuellen Server Daten hinterlegt, die benötigt werden, um sich mit dem Server zu verbinden. Nun brauchen nur noch alle Clients diese Datei nach der aktuellen Adresse fragen um sich mit Server zu verbinden. Die genaue Installationsleitung ist in dem Blog blog.kibotu.net[62] zu finden.

7.2.4 Unity3D Client (JR)

Zu Beginn eines Multiplayer Spieles werden, Event Listener zugewiesen, die in Abbildung 63 aufgelistet sind, indem zu dem SocketHandler Delegates hinzugefügt werden.

```
1 SocketHandler.OnJSONEvent += OnConnectEvent;
2 SocketHandler.OnJSONEvent += OnJSONEvent;
3 SocketHandler.OnJSONEvent += OnStringEvent;
4 SocketHandler.OnJSONEvent += OnConnectionFailedEvent;
5 SocketHandler.OnJSONEvent += OnReconnectEvent;
6 SocketHandler.OnJSONEvent += OnErrorEvent;
7 SocketHandler.OnJSONEvent += OnDisconnectEvent;
```

Abbildung 63: Zuweisen von Event Handlern [63]

Dank den dynamisch aktualisierten und online hinterlegten Serververbindungsdaten, brauchen diese nur abgefragt werden und mit dem SocketHandler.Connect() mit dem Server verbunden werden. Alle Netzwerk Events des Clients werden in den

[61] http://goo.gl/nZkd38
[62] http://blog.kibotu.net/jquery/dynamic-updating-local-ip-node-js
[63] http://goo.gl/ztO21x

jeweils dafür vorgesehen Komponenten der Gameobjekte mit Hilfe von SocketHandler.SharedConnection.Emit() und SocketHandler.SharedConnection.EmitNow() versendet.

7.2.5 Netzwerk Event API (JR)

Die wichtigsten Server und Client Netzwerkevents sind in der Tabelle 13 aufgelistet.

Die Events werden mit Socket.io's Emit[64] Funktion versendet. Auf Server Seiten wird die Broadcast[65] Funktion zum verteilen der Events in einem Raum verwendet, welche an alle im Raum Nachrichten sendet, außer dem aktuellen Client, so dass ein Client nicht seine eigenen gerade versendeten Nachrichten zurück bekommt.

Auf Client Seiten wird ebenfalls die Emit[66] Funktion verwendet, jedoch es gibt bei dem Client die Möglichkeit Nachrichten sofort zu senden oder aber gebündelt im nächsten Gameloop. Denn das zusammenfassen von mehreren Events zu einem größerem Paket vor dem versenden, verringert die Anzahl an Paketen, die gesendet werden müssen und somit entlastet es nicht nur die Leitung, sondern ebenfalls die Serverlast.

7.2.6 Konsistenz: FPS vs Ping (JR)

Laut den Design Säulen soll der Spieler einen möglichst nahe Echtzeit reagierenden Multiplayer erleben. Doch wirkliche Echtzeit ist nur eine Illusion. Einerseits existieren Zeitverzögerungen bei der Kommunikation zwischen zwei Netzwerkclients und andererseits gibt es Hardwaregrenzen, besonders bei mobilen Endgeräten. Außerdem ändert sich die Netzwerk- und Hardwareauslastung über einen Zeitraum. Die üblichen FPS auf mobilen Geräten betragen 60, in Drag'n Slay haben wir es jedoch auf 30 limitiert um auf schwächeren Geräten erhöhte Leistung zu erzielen. Die in Drag'n Slay getesteten übliche Pingzeiten liegen zwischen 30 und 150 Millisekunden, abhängig von der verwendeten Datenübertragungsart. Im Wi-Fi sogar 0 bis 3 Millisekunden.

Hierzu ein kleines Beispiel, wieso das Auseinandersetzen mit dem Konzistenproblem wichtig ist. Angenommen Spieler Markus sendet seine Flugzeuge von Insel A zu der gegnerischen Insel B von Spieler Anne. Anne sendet jedoch in dem Moment in weiser Voraussicht Verstärkung von Insel C zu ihrer Insel B. Nun, wegen der zeitlichen Verzögerung des Übermittelns beider Befehle, schafft es Markus auf seinem Android

[64] `https://github.com/Automattic/socket.io#serveremit`
[65] `https://github.com/Automattic/socket.io#socketinroomstringsocket`
[66] `http://goo.gl/66Y57D`

Tablett diese Insel einzunehmen und Verstärkung zu produzieren und Anne schafft es auf ihrem iPhone ihre Insel zu verteidigen und ebenfalls Nachschub zu produzieren. Nun existieren zwei widersprüchliche Welten. Wer hat nun Recht?

Eine Strategie wäre es einen von beiden Hoheitsrechte zu geben und bei Spielbegin den Host zu definieren, der immer Recht hat. Das löst zwar das Konsistenzproblem, erschafft jedoch zugleich neue, wie zum Beispiel erhöhte Netzwerk- und Hardwareanforderung an den Host, was durch erhöhte Spieleranzahlen sehr verstärkt wird. Besser wäre es doch, wenn die Skalierung dem Server obliegt. Außerdem hätten wir als Entwickler bessere Möglichkeiten Netzwerkauslastungen zu überwachen.

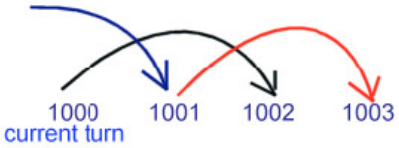


Abbildung 48: Planung von Spielzügen[67]

Eine bessere Strategie ist die Spielzugkontrolle, in dem 2001 veröffentlichten Artikel *1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond* von Mark Terrano und Paul Bettner . Die Grundidee ist es, eine Rundenbasierende Kommunikation stattfinden zu lassen. Wobei jedes neue Spielevent zwei Spielzüge voraus geplant wird, wie in Abbildung 48 zu sehen ist. So dass sichergestellt werden kann, dass alle Clients ein Event erst dann ausführen, sobald alle dieses Event erhalten und bestätigt haben. Diese Technik bedeutet zwar initial mehr Aufwand, bringt aber einige kostenlose Nebeneffekte, regeln von FPS und Spielgeschwindigkeit zusätzlich mit. Wie dies genutzt wird, ist in der Tabelle 15 gut zu erkennen.

	Pakete verwerfen	Zeit Spielzug erhöhen	Zeit Spielzug senken	FPS limierteren
Niedriger Ping Niedrige FPS		X bei hohen Lags		X
Hoher Ping Niedriger FPS	X	X		X
Hoher Ping Hohe FPS	X bei Timeouts	X		
Niedriger Ping Hohe FPS			X	

[67] `http://www.gamasutra.com/view/feature/131503/1500_archers_on_a_288_network_.php`

Tabelle 15: Netzwerverhalten variabler Ping und FPS

In dem folgenden Flussdiagramm in Abbildung 49 wird genau dargelegt, wie der Ablauf dieser Technik in Drag'n Slay verläuft. Die ganze Zeit über werden asynchron Spiel Events empfangen und gesendet. In der Main Loop wird jede Schleifeniteration überprüft, ob die Zeit eines Spielzuges vorüber ist. Bei Drag'n Slay werden Initial zehn Spielzüge pro Sekunde ausgeführt.

Sollte es nicht der Fall sein, so wird die FPS und Spielzeit analysiert. Ist die Zeit jedoch vorbei, so wird eine *'Done'* Nachricht versendet, welche ein Timing und geplante Paket Ids beinhaltet. Daraufhin wird der Planungszähler erhöht, so dass neue Spielevents ebenfalls zwei Spielzüge später ausgeführt werden.

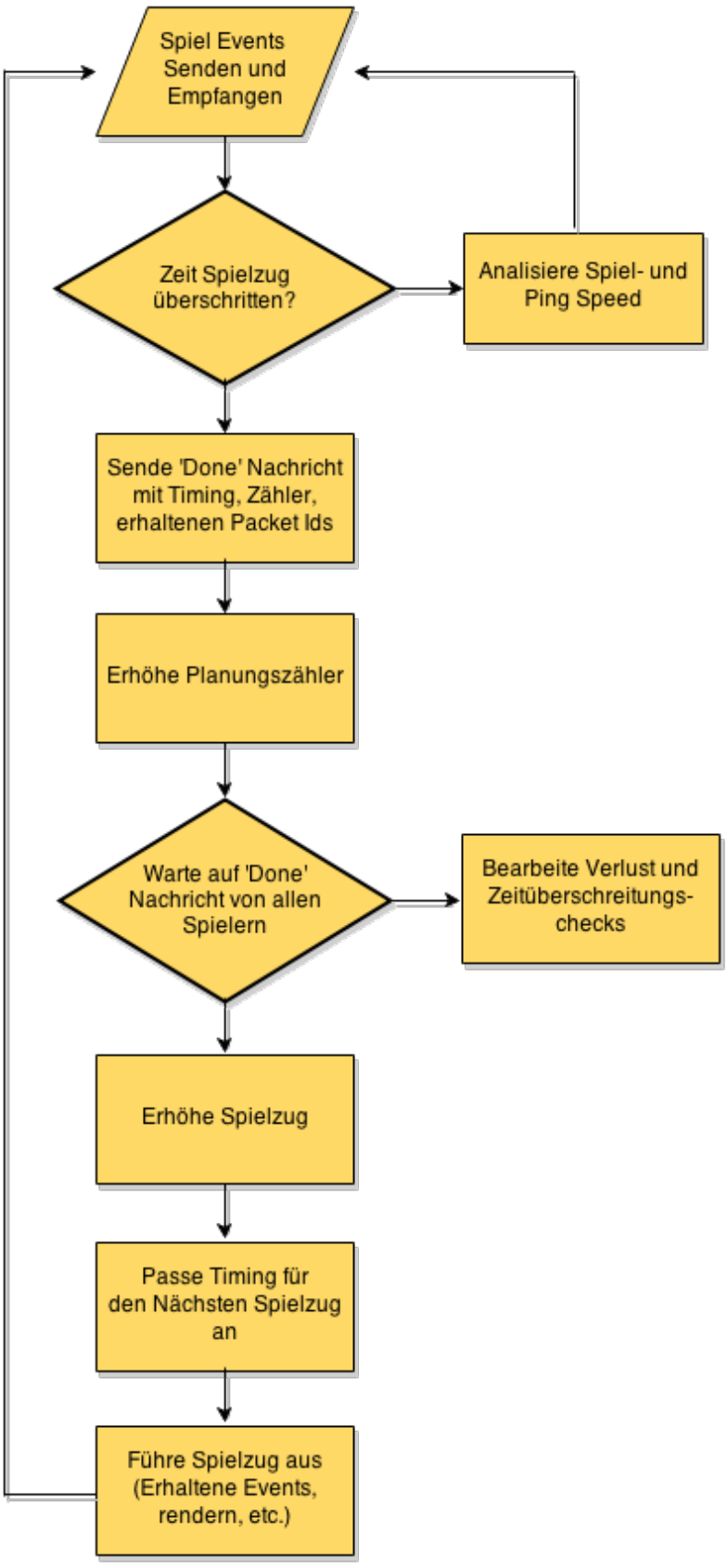


Abbildung 49: Bearbeitung eines Spielzug[68]

Nun wird auf alle Spieler gewartet, bis deren *'Done'* Nachricht ankommt. Während des Wartens werden Verluste und Zeitüberschreitungen überprüft. Hier kann überprüft werden, ob ein Spieler das Spiel verlassen hat.

Verluste sind geplante aber nicht ausgeführte Events, da mindestens ein Mitspieler ein Event nicht oder zu spät erhalten hat. In dem Fall wird entschieden, ob es Sinn ergibt ein Event erneut zu versenden, oder es zu verwerfen. Natürlich ergibt es keinen Sinn, Events von vor 1000 Spielzügen auszuführen.

Sind alle *'Done'* Nachrichten angekommen, so wird der Spielzug Zähler erhöt. Außerdem wird hier an Hand der aktuellen durschnittlichen FPS und des durchschnittlichen Pings der letzten paar Spielzüge berechnet, wie schnell die nächsten Spielrunden sein können. So könnte es passieren, dass bei besonders guten Bedingungen eine Spielrunde nun 60 mal in der Sekunde ausgeführt wird und die FPS Limitierung weg fällt.

Zum Schluss werden nun alle geplanten Spiel Events ausgeführt.

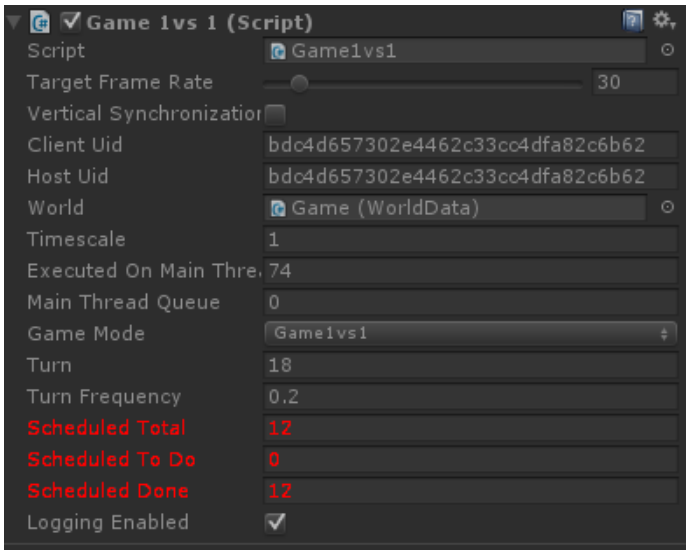


Abbildung 50: Unity Inspector Game

Im Unity3D Inspektor, Abbildung 50, können Konsistenzkonflikte Live überwacht werden. So wird der *Scheduled To Do* Zähler erhöht, sollten neue Netzwerk Events eingehen oder auf dem Client passieren und werden in dem Moment verringert, wenn das Event abgearbeitet wurde. Sollte dieser Zähler für mehr als zwei Spielzüge unbehandelte Events anzeigen, so ist etwas schief gegangen.

[68] `http://www.gamasutra.com/view/feature/131503/1500_archers_on_a_288_network_.php`

Fazit

Diese Architektur ermöglicht es nicht nur TCP[69] und UDP[69] gleichermaßen zu unterstützen, sondern ebenfalls gezielt während des Spielverlaufs auf unterschiedliche FPS und Ping Raten einzugehen. Außerdem ermöglicht es Spiel Events zu versenden, ohne diese einzeln anzuerkennen. Da diese einmalig nach dem erhalten mit der *'Done'* Nachrichten zusammen gefasst bestätigt werden können, nämlich genau dann wenn es wichtig ist. Im Moment werden TCP Sockets verwendet und dies wird daher bereits auf der Transportschicht garantiert. Es ist es geplant später auf UDP umzusteigen, um noch mehr Netzwerkgeschwindigkeit herauszuholen.

7.2.7 Spiel Event Netzwerkpakete (JR)

Zwar wurde die Art und Weise der Synchronisierung geklärt, jedoch würde dennoch in Inkonsistenzen gelaufen werden können, da zum einen keinerlei Transformationen außer der Levels zu Beginn übermittelt werden und zum anderen sehr viele Zufallsmechanismen verwenden werden, um das Spiel lebendiger wirken zu lassen, zum Beispiel die Idle Bewegung von Inseln, oder der Looping der Flugzeuge. Das hat zur Folge, dass die Flugdauer von Endgerät zu Endgerät variiert und einen Schneeballeffekt auslößen würde, der dazu führt, das einige Flugzeuge früher oder später schießen, et cetera.

Doch welche Spiel Events müsste denn überhaupt alles synchronisiert werden, damit in keine Inkonsistenzprobleme gelaufen wird?

Ganz klar wäre erst einmal der Spielstart, der beginnt, nachdem ein Spieler das vom Server angeforderte Level geladen hat. Wichtig wären außerdem folgende Events:

- immer wenn ein Flugzeug erstellt wird,
- das Los fliegen,
- das Ankommen,
- das Zerstören eines Flugzeuges,
- aber auch wenn Raketen los fliegen, zu welchem der Zeitpunkt, wenn der Schaden abgezogen wird, um Geisterraketen zu vermeiden,
- sowie der Zeitpunkt, sowie eine Insel konvertiert
- sobald die Konvertierung abgeschlossen ist.

[69] `http://gafferongames.com/networking-for-game-programmers/udp-vs-tcp/`

- Natürlich gehören auch Spezialattacken dazu.

Wird die Größe von Paketen Sorge bereiten?

```
1  {
2    "name": "message",
3    "args": [
4      {
5        "message": "spawn-unit",
6        "spawns": [
7          {
8            "island_uid": 9,
9            "uid": 500009
10         }
11       ],
12       "packageId": 25,
13       "scheduleId": 10,
14       "ack": true
15     }
16   ]
17 }
```

Abbildung 70: Flugzeug Spawn Event [70]

Dazu ein Standard Paket eines Spawn Events (Abbildung 70). Wie alle selbst definierten Pakete, wird es im JSON Format überragen. Es besteht aus einem Namen, Zeile 2, und Argumenten, Zeile 3 bis 16. Der Name kommt leider von der C# Socket.io Implementierung und kann nicht ohne größeren Aufwand geändert werden. Der Typ der Nachricht steht in Zeile 5, gefolgt von einem Array aller zu erstellenden Flugzeuge und deren Inseln. Wichtig hierbei ist, das die Unique Id der Insel, sowie die Unique Id des neuen Flugzeuges auf allen Clients gleich ist, so dass eine einmalige Identifizierung für spätere Spiel Events der Gameobjekte stattfinden kann. Zusätzlich gibt es noch eine Paket Id, welche für zukünftige UDP Fehlerüberprüfungen bereits eingeplant ist. In Zeile 13 befindet sich der geplante Spielzug, in dem das Flugzeug erstellt wird. Zusätzlich enthält jedes Netzwerkpaket noch ein Bestätigungs-Flag, dass bekannt gibt, ob auf diese Nachricht eine Antwort erwartet wird.

Die Paketgröße des Spawn Events beträgt 292 Bytes, wobei 270 Bytes von den 137 Zeichen und 22 Bytes[71] overhead von Socket.io's Emit Funktion kommen. Jedoch

[70] `http://goo.gl/d1fZBA`
[71] http://stackoverflow.com/a/18907653

werden dank gzip[72] am Ende sogar nur noch 143 Bytes übetragen. C# verwendet 2 Bytes für UTF-8 Strings. Auch unser Standard Level mit 1.680 Bytes wird so nur noch mit 537 Bytes auf die Clients verteilt.

Fazit

Die versendeten Pakete sind relativ winzig. Auch die Übetragungshäufigkeit ist relativ gering, da fast nur Spiel Events übermittelt werden, im Vergleich zu First Person Shootern, die quasi fast jeden Frame Transformationen abgleichen müssen. Das bedeutet, das Client-seitig mit Drag'n Slay niemals an irgendwelche Grenzen gestoßen wird, wie zum Beispiel die maximale TCP Paketgröße von 64 KB. Die niedrigen Datenpakete schonen natürlich auch die Langzeitnutzung der Spieler, Außerdem müsste Drag'n Slay dadurch auch unter relativ schlechten mobilen Netzwerverbindungen spielbar sein.

7.2.8 Monitoring (JR)

Es ist schwierig die Netzwerkkommunikation zwischem dem Server und alle Spieler eines Spieles auf ihren Endgeräte gleichzeitig zu debuggen.

Daher wurde ein Monitoring Tool erstellt (Abbildung 51). Mithilfe von Node.js's Express[73] Plugin kann diese von überall aus abgerufen werden.

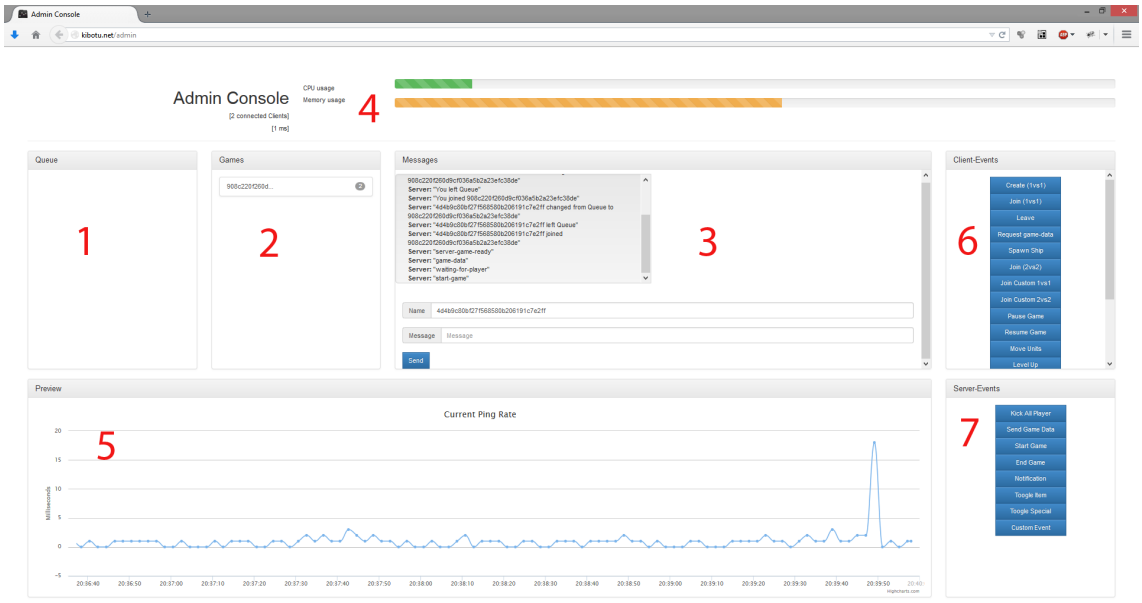


Abbildung 51: Admin Console[74]

[72] http://www.gzip.org/
[73] https://github.com/visionmedia/express
[74] http://www.kibotu.net/admin

Wenn ein Spieler sich mit dem Server verbindet, so wird er in die Standard Warteschleife (1) gelegt. Sollte dieser Spieler ein neues Spiel erstellen, so wird er in die Spiele Liste (2) verlegt. Alle Netzwerkkommunikation, die in einem Spiel stattfindet, kann nun in der Logausgabe (3) verfolgt werden. Für Testzwecke können auch selbst Nachrichten an die Spieler gesendet werden. Zusätzlich können ein paar vorgefertigte Nachrichten Makros für Client Befehle (6) und Server Befehle (7) per Knopfdruck versendet werden. Die Admin Console ermöglicht es ebenso die Server CPU und RAM Auslastung (4) zu verfolgen. Ebenfalls können die Pingzeiten der Spieler (5) als Life-Tracking Chart rückverfolgt werden.

Erstellt wurde die Website mit Jade[75] als HTML Template Engine und Twitter's Bootstrap[76] CSS Framework. Außerdem wird JQuery[77] benötigt für das updaten der Echtzeit Logausgaben, sowie für Highcharts [78], für das Tracking der Pingzeiten und natürlich die Front-End Variante von socket.io, da die Kommunikation zum Server über Websockets stattfindet.

Abbildung 52: Network View

Abbildung 52 zeigt die fünft eingebauten Icons, um die Netzwerkzustände zu visualisieren. Die Icons ändern ihre Helligkeit sollten sie aktiv werden. Diese simplen Icons haben den Entwicklungsvorgang besonders in der zur Installationsphase sehr erleichtert.

Die Symbole bedeuten von links nach rechts: Nicht mit dem Server verbunden, mit dem Server Verbunden, Netzwerk Nachricht erhalten, Netzwerk Nachricht bestätigt und Netzwerk Fehler aufgetreten.

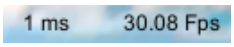


Abbildung 53: Ping and FPS

Auch erfolgt die Überwaching (Abb. 53) der Serverseitig überwachten Ping zum Server und den aktuellen FPS des Clients.

Des weiteren ist geplant den Server zu optimieren um Speicherlöcher zu schließen. Zur Zeit verliert der Server etwa 8KB pro Sekunde, die hier für den Prototypen vernachlässigt werden.

[75] http://jade-lang.com/
[76] http://getbootstrap.com/
[77] http://jquery.com/
[78] http://www.highcharts.com/

7.3 Testgeräte (JR)

Als Testgeräte wurden mit einem Samsung Galaxy S2, Sony Xperia S, Nexus 7, Milestone 2, sowie mit einem Lenovo z580 und einem Acer Aspire M5 getestet. So wurden als Mobile Geräte nicht nur Smartphones, sondern auch Tablets und Laptops zum testen genutzt.

7.4 Optimierungen (JR)

In First Person Shootern, sind Techniken wie Entity Interpolation[79], Input Prediction[79] und Lag compensation[79] relativ bekannt und unbedingt notwendig.

Doch diese Techniken kompensieren Latenzprobleme von Transformationsupdates. Sie werden bei Drag'n Slay nicht benötigt, weil hier hingegen keine Transformationen übermitteln werden und alles notwendige Client-seitig simulierbar ist.

Jedoch erlaubt unser Design es uns dennoch an einigen Stellen zu tricksen um dem Spieler ein viel flüssigeres Erlebnis zu bieten, als es eigentlich korrekt wäre. Angenommen ein Spieler gibt den Befehl seine Flugzeuge von Insel A nach Insel B zu senden. Dann würde der Befehl erst ausgeführt werden, wenn alle Mitspieler dies in ihrer nächsten *'Done'* Nachricht bestätigt haben. Allerdings fliegen die Flugzeuge Client-seitig bereits los, da wir ja das ankommen synchronisiert wird und für die nächsten Schadensberechnungen zwingend notwendig ist. Im Worst-Case, werden die Flugzeuge kurzzeitig weiterhin mit Raketen beschossen, von Inseln von denen sie kamen. Das ist allerdings nicht weiter schlimm, da es dem Spieler visuell kommuniziert wird.

Zukünftig geplant ist die Übertragung mit UDP Sockets, SSL, sowie eventuell Protobuffer zu verwenden um komplette Game States zu übertragen.

8 Businessmodell (JP)

Neben der Erstellung eines Prototypen ist es ebenfalls wichtig sich Gedanken über eine Mögliche Vermarktung zu machen. Hierbei wurden Ideen über eine mögliche Firmenpolitik durchdacht und Überlegt wie der Kunde zum Zahlen animiert werden kann.

Wichtige Partner Obwohl es sich bei Drag'n Slay um eine Bachelor Arbeit handelt, wurden Partner gefunden, welche bei der Entwicklung von Hilfe waren.

[79] `https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking`

So wurde zum Beispiel auf nachfrage und Darbietung der Spielidee bei CodeandWeb eine kostenlose Lizenz von TexturePacker zur Verfügung gestellt, welche für Drag'n Slay genutzt werden darf.

Schwerpunkte und Wertversprechen Die Schwerpunkte der Entwicklung von Drag'n Slay liegen auf dem erzielen einer zufriedenen Spielercommunity auf dem mobilen Markt. Dieses kann nur durch ein zuverlässiges Produkt erreicht werden, was bei möglichst vielen Usern auf unterschiedlichen mobilen Endgeräten keine Probleme erzeugt. Hierbei sollen keine Fehler zu lasten der User entstehen, damit ein positives Firmenbild bestehen bleibt.

Kundenbeziehungen Die Entwicklung von Drag'n Slay soll nach außen für den Kunden sehr offen und durchschaubar sein, da eine nahe Zusammenarbeiten mit den Usern gewünscht ist. So sollen die User immer per Updatelogs über fertige und zukünftige Erneuerungen informiert werden und dabei aktiv eigene Ideen einbringen können.

Kundensegmente Drag'n Slay soll sich sowohl an erfahren so wie Gelegenheitsspieler richten. Geschlecht und Alter soll hierbei in erster Linie keine Rolle spielen. Jedoch ist das alter nach unten durch das im jeweiligen Land geltende Gesetz zur Geschäftsfähigkeit und Jugendfreigabe abhängig.

Kanäle Zur Verbreitung der eigenen Inhalte, Neuigkeiten und zum informieren potentieller User wurde neben einem Facebook-Profil auch ein interaktiver Wordpress-Blog angelegt, in dem neben Arbeitsprozessen auch Designkonzepte öffentlich gezeigt werden. So wurde durch die Kommentarfunktion eine weitere Möglichkeit geschaffen, ein Feedback von potentiellen Nutzern zu aktuellen Bearbeitungsthemen zu erhalten.

Kostenstruktur Für die Entwicklung des Prototypen fallen insbesondere fixe Kosten im Bereich der Benötigten Software an. Es wird zwar versucht zum Großteil mit Open Source Software zu arbeiten, jedoch gibt es im Bereich der Editoren und Grafikprogramme teils keine vergleichbare kostenlose Lösung. Kosten für Gehälter entfallen zunächst und werden nur durch prozentuale Anteile an Drag'n Slay vergütet.

Schlüsselressourcen Die Schlüsselressourcen setzten sich insbesondere aus den Kundenbeziehungen, den Kanälen zum Verbreiten von Informationen und dem Einhalten der Wertversprechen gegenüber der Kunden.

Revenue-Stream Der spätere Nutzer soll nicht gezwungen werden für das Spiel zu

zahlen. Das Ziel ist es eine große Anzahl an zufriedenen Spielern zu erlangen, welche die Entwickler durch den Kauf von visuellen Spielzusätzen unterstützen wollen. Diese Zahlmotivation soll aber durch unterschiedliche Verkaufsstrategien, welche später erleutert werden gestärkt werden.

8.1 Micropayment (JP)

Unter Micropayment, zu deutsch Kleinbetragszahlung versteht man ein Zahlverfahren, bei dem es um Kleinstbeträge geht. Diese fallen im digitalen Zeitalter immer öfter an. Ein Beispiel hierfür sind Käufe einzelner digitaler Musikstücke oder die oft geringen Kaufpreise von digitalen Zusatzinhalten in Spielen.

Micropayment soll mehrere Probleme beseitigen, die bei konventionellen Zahlungsmethoden auftreten würden. Die beiden größten sind die Zusatzkosten und Verzögerungen, die zum Beispiel beim Zahlen mit einer Kreditkarte anfallen würden. Tätigt ein Kunde eine Zahlung mit einer Kreditkarte, so fallen im Normalfall Gebühren von 1,5% - 2% und eine Grundgebühr für den Händler an. werden diese Prozentsätze auf einen Betrag von 0.50 Euro gerechnet, wären diese zu vernachlässigen. Da aber zusätzlich eine Grundgebühr für die Bearbeitung anfällt, würde der Händler am ende sogar eine negative Bilanz ziehen, da die Grundgebühr den Wert übersteigen würde. Da im Micropayment häufig Digitale Währungen genutzt werden, gibt es auch praktisch keine Wartezeiten, bis ein Betrag überwiesen ist. Da ein Kunde bei digitalen Medien erwartet sofort nach dem kauf einen Zugang zu den erworbenen Daten zu erhalten, ist ein schneller und zuverlässiger Geldtransfer von Vorteil.

8.2 Anforderungen an ein Micropayment-System (JP)

Möchte ein Händler seinen Kunden ein eigenes Micropayment-System anbieten, muss er sich im klaren sein was seine Kunden von seinem Bezahlsystem erwarten und welche Herausforderungen für den Betrieb entstehen. Diese werden in der folgenden Tabelle 16 gegenüber gestellt.

Da sämtliche finanzielle Daten sehr sensibel sind, ist ein hoher Sicherheitsstandart unablässig. Daher erwartet jeder Kunde, dass ihm durch die Nutzung kein finanzieller schaden durch Dritte zugefügt werden kann. Hier ist es die Aufgabe des Händlers für eine hohe Sicherheit zu sorgen und im Falle eines Systemfehlers auch zu diesem zu stehen und die Kunden auf einen möglichen Fehler hinzuweisen

Da es im Micropayment genau darum geht viele kleine Beträge zu überweisen, sollten diese Transaktionen für den Kunden kostenlos sein, dass heißt aber auch, das die

Transaktionskosten für den Anbieter ebenfalls sehr gering sein müssen, da diese in die Verkaufspreise mit eingerechnet werden und dabei die Preise nicht in die Höhe treiben sollten.

Anforderungen aus Händlersicht	**Anforderungen aus Kundensicht**
Hoher Sicherheitsstandard	Hohe Sicherheit
Geringe Transaktionskosten	Kostenfreiheit
Hohe Verbreitung und Akzeptanz des Systems	Viele Akzeptanzstellen
kostengünstige Implementierung im Unternehmen	unkomplizierte Software- und Hardwareanforderungen

Tabelle 16: Vereinfachte Tabelle: http://de.wikipedia.org/wiki/Micropayment

Möchte ein Unternehmen ein solches Zahlsystem zum Erfolg führen, muss es für viele Akzeptanzstellen sorgen. Das heißt zum einen das der Kunde am Liebsten ein Micropaymentsystem für alles hätte. Für Drag'n Slay ist das nicht möglich, solange wir nur ein Spiel anbieten. Zum anderen ist es aber auch wichtig möglichst viele Überweisungsdienste anzubieten, über die der Kunde Währung tauschen kann. Insbesondere weil ein Großteil unserer Zielgruppe nicht volljährig ist und damit nicht über eine Kreditkarte besitzt, bietet der Zahlungsumweg über Paysafe Karten oder ein direktes Zahlen über den Google Playstore eine Alternative an.

Die Implementation eines Zahlungssystems ist für ein Unternehmen mit hohen Kosten verbunden und lohnt sich erst, wenn die Entwicklungskosten durch die Einsparungen an nicht zu zahlenden Überweisungs und Servicekosten gedeckt werden. Da der Kunde an dieser Stelle eine leicht zu Bedienende, sichere und schnelle Software erwartet, sollte im Fall einer Eigenentwicklung nicht gespart werden. Erscheint dem Kunden der Weg der für eine Kleinsbetrags Überweisung anfällt zu kompliziert, wird er unter Umständen auf einen Kauf verzichten

8.3 In Game Währung, Cloud Coins (JP)

Beim erstellen einer In Game Währung, auch eGeld genannt, müssen sich vor dem Bereitstellen dieser einige Punkte genau überlegen werden, da späteres ändern der Wertigkeit den Kunden irritieren und vor allem verärgern kann.

Für die Drag'n Slay Cloud Coins haben gibt es einen Wechselkurs von 1 zu 120. Das heißt also für jeden Euro den der Kunde eintauscht erhält er 120 Cloud Coins. Ob dieser Wechselkurs zu Beträgen führt, mit denen der User sich wohl fühlt und gerne

hantiert lässt sich zu diesem Zeitpunkt nicht sagen und wird in der Weiterführung des Projekts während der Beta-Phase ermittelt. während dieser Testzeit wird jedem Nutzer wöchentlich ein Betrag an Cloud Coins übertragen, den er zur freien Verfügung erhält. Um zu Beginn der nächsten Woche neue Cloud Coins zu erhalten muss der User jeweils eine kurze Umfrage über seine Erfahrungen mit dem Bezahlsystem beantworten. Die Fragestellungen der Umfrage werden so gestellt sein, dass sie bei der Auswertung Informationen liefern, die dabei helfen sollen, den Wechselkurs zu optimieren. Dieses soll für den User aber nicht offensichtlich erkennbar sein, da so seine Antworten beeinflusst werden können. Da dieses ein schwieriges Unterfangen ist, sollte diese Tätigkeit am besten an ein Fachkundiges Team ausgelagert werden. Am Ende der Beta muss dann ein fester, funktionierender Wechselkurs für die Cloud Coins festgestellt werden, der nicht mehr geändert wird.

Würde Drag'n Slay in einer größeren Software Schmiede entwickelt werden, in der geplant wird weitere Spiele zu veröffentlichen, so wäre es naheliegend eine Firmenwährung zu entwickeln, welche für alle Spiele der Firma übergreifend gleich ist. So könnten bei späteren Neuentwicklungen Zeit und damit Kosten gespart werden.

8.4 Steigerung der Zahlmotivation (JP)

Um die Zahlmotivation der Kunden zu steigern, setzt Drag'n Slay auf unterschiedliche Strategien. Hierbei steht an erster Stelle den Kunden nicht zum Zahlen zu zwingen. Alle in Drag'n Slay vorhandenen "in game Käufe", dienen nur visuellen-Effekten und Boosts, welche keinen direkten Vorteil innerhalb des Spiels gegenüber anderen Mitspielern erbringen. Die Funktionalität dieser Herangehensweise wird durch die vielen, auf dem Markt erfolgreich existierenden Titel bestätigt. Der Kunde von heute spendet lieber mehrmals Geld gegen kleine "Geschenke", als einen Vollpreis zu Beginn zu bezahlen, selbst wenn hierdurch am Ende höhere Gesamtausgaben entstehen. Hierbei ist es auch hilfreich, wenn eine sogenannte "in game Währung" benutzt wird. So lassen sich zum Beispiel für 10.00 Euro, 1200 Cloud Coins erwerben, welche dann im "in game Shop" gegen hübsche Skins oder Boosts getauscht werden können. Hierbei wird bewusst für die in game Währung ein geringer Eigenwert festgelegt, da so große Zahlen zu Stande kommen. Auf den ersten Blick mag dieses wenig Sinn ergeben, weil es dazu führt das der Kunde mit großen Zahlen rechnen muss. In Wirklichkeit führt bei den meisten Menschen das Ausgeben von großen Zahlen jedoch zu einer Art Macht und Wohlgefühl. Die Tauschfreudigkeit des Kunden kann zusätzlich durch Bonus-Programme angeregt werden. Hierbei unterscheiden wir in dauerhafte und Event gebundene Rabat-Aktionen. Bei den dauerhaften Rabat-Aktionen han-

delt es sich lediglich um einen Auszahlungs-Bonus von 20%, wenn der Kunde mehr als 10.00 Euro eintauscht oder 25% ab 15.00 Euro. Die Intervalle werden hierbei immer kleiner, um den Kunden dazu zu bringen, mehr Geld auszugeben, um in den nächsten Bonus-Bereich zu kommen.

Zusätzlich soll die Kaufbereitschaft der Kunden durch zeitlich begrenzte Rabatt Events gesteigert werden. Bei diesen gibt es über einen begrenzten Zeitraum eine Preisreduzierung auf einen bestimmt Artikel im in game Shop. Da der Kunde Angst hat, einen Nichtkauf mit reduziertem Preis später zu bereuen, kann dadurch die Kaufentscheidung positiv beeinflusst werden.

Eine weitere Möglichkeit den Kunden zu einer größeren Kaufbereitschaft zu bewegen ist es Bundles anzubieten. So werden dem Kunden Pakete angeboten, welche mehrere Kaufgegenstände auf einmal beinhalten. Diese nach Möglichkeit in einem Themen gebundenen Kontext stehenden Pakete sind im Gesamtpreis niedriger als würde der Kunde jeden enthaltenen Artikel einzeln kaufen. Der Trick ist, dass der Kunde auf diesem Weg dazu bewegt wird, mehr zu kaufen als er ursprünglich vor hatte, da er den Eindruck bekommt beim Kauf des Bundles zu sparen. In Wirklichkeit kauft er jedoch am Ende einen Artikel mehr und zahlt nur für diesen einen verringerten Preis. Wenn es bereits geschafft wurde ein größeres Kundenfeld zu erreichen, kann man dieses nutzen um neue Bundles zusammen zu stellen. Durch Kaufanalysen der Kunden, lassen sich beliebte Artikel erkennen und damit Trends im Kaufverhalten der Kunden erahnen. Wird auf diese Trends eingegangen und es werden zum Beispiel zwei beliebte Artikel mit einem weniger beliebten Artikeln im Bundle kombiniert, so kann dieses zu einer Verkaufssteigerung des weniger beliebten Artikels führen. Wichtig dabei ist es dem Kunden immer mehr Artikel anzubieten die er sowieso kaufen möchte, als Artikel die er eigentlich nicht benötigt.

Ebenfalls muss die Möglichkeit des Rücktausches der in game Währung ausgeschlossen werden. Denkbar ist es, dass dieses einen zusätzlichen Gewinn einbringt, da Restbestände einfach liegen bleiben und nicht genutzt werden. Dieses ist aber nicht der Fall, da der finanzielle Gewinn lediglich beim Tausch der Währung gemacht wird. Mit dem Ausschluss der Rückzahlung können Steuerrechtliche Probleme umgangen werden, welche zusätzliche Kosten und Arbeitsaufwand bedeuten würden.

9 Conclusion

Als wir vor der Anmeldung der Bachelorarbeit standen, haben wir uns bereits für eine Grundthematik des Spiels entschieden gehabt. Bevor mit der Implementation des Drag'n Slay Protoypen begonnen wurde, wurden zahlreiche Designschwerpunkte und Gamearts erstellt. Mit den Bilden der Ideen und visuellen Eindrücken wurde dann zunächst eine Marktanalyse durchgeführt um zu überprüfen, ob die Entwicklung eines Prototypen bei den aktuellen Gegebenheiten des Marktes lohnenswert sind.

Mit den durch die Marktanalyse erhaltenen Informationen konnten dann weitere Vertiefungen und Verfeinerungen an Use Cases, Design Pillars und Spielmechaniken getätigt werden, um das Spiel anhand seiner Komplexität und Grafik auf dem Markt einmal zu machen. Bei der Ausarbeitung des Prototypen erhielten wir durch das Lösen von Problemen große Einblicke in die Optimierung unserer Netzwerkkommunikation und durch die begrenzten Ressourcen der mobilen Endgeräte lernten wir unterschiedliche Methoden zum verbessern der Stabilität von Drag'n Slay kennen. Mit dem Fortschreiten des Projekts setzten wir uns auch mit einem möglichen Geschäftmodell und Ingame-Käufen auseinander.

Mit dem Ergebnis des Projekts und dem großen Umfang der neu gewonnenen Informationen sind wir sehr zufrieden, dennoch sind wir sehr motiviert in der Zukunft aus dem Prototypen eine fertiges Produkt zu entwickeln.

Anhang

A GUI

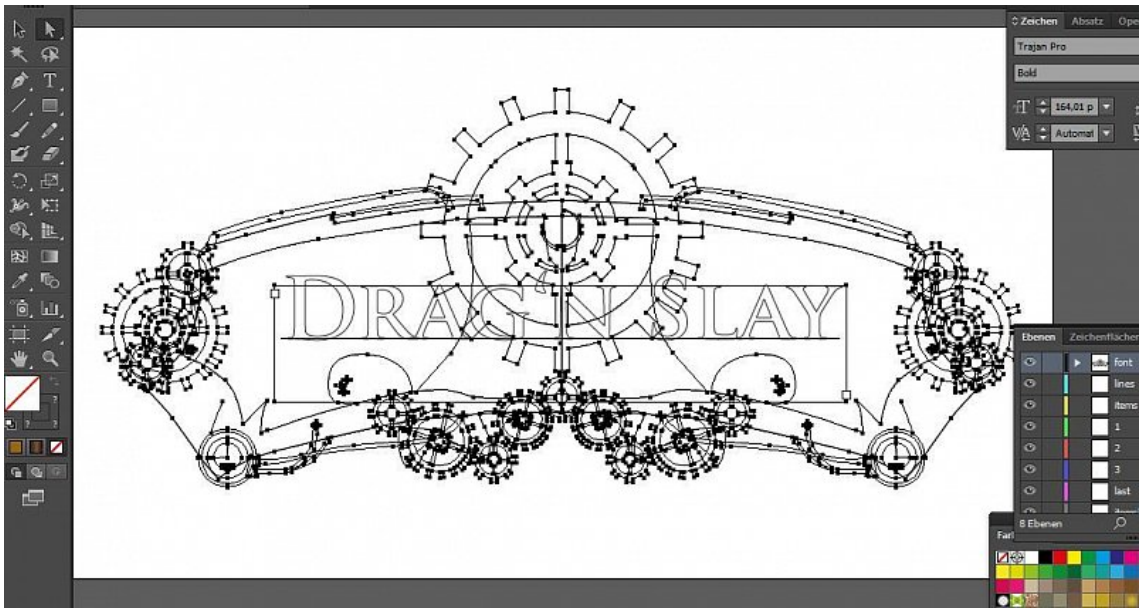


Abbildung 54: Wirreframe des Banners

Abbildung 55: Banner mit und ohne Texturierung

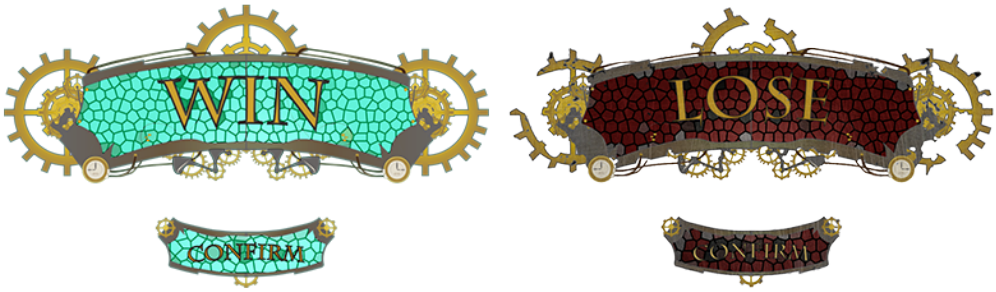


Abbildung 56: Win und Lose Screen

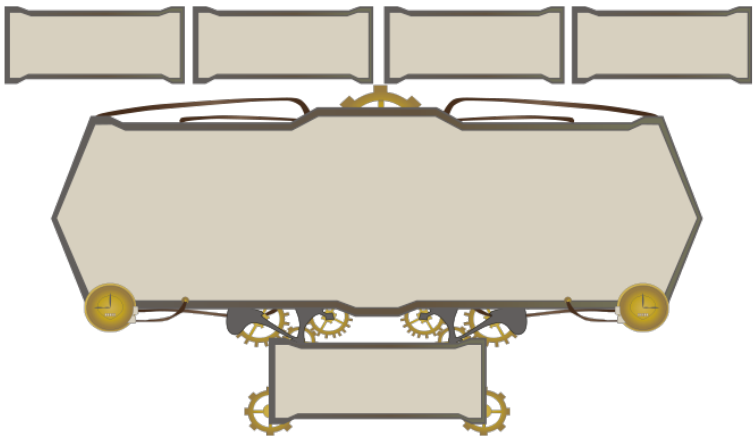


Abbildung 57: UI Konzept des Shops

The Business Model Canvas

Designed for:

Designed by:

On: Day | Month | Year

Iteration: No.

Key Partners

Who are our Key Partners?
Who are our key suppliers?
Which Key Resources are we acquiring from partners?
Which Key Activities do partners perform?

MOTIVATIONS FOR PARTNERSHIPS:
Optimization and economy
Reduction of risk and uncertainty
Acquisition of particular resources and activities

Key Activities

What Key Activities do our Value Propositions require?
Our Distribution Channels?
Customer Relationships?
Revenue streams?

CATEGORIES
Production
Problem Solving
Platform/Network

Key Resources

What Key Resources do our Value Propositions require?
Our Distribution Channels? Customer Relationships?
Revenue Streams?

TYPES OF RESOURCES
Physical
Intellectual (brand patents, copyrights, data)
Human
Financial

Value Propositions

What value do we deliver to the customer?
Which one of our customer's problems are we helping to solve?
What bundles of products and services are we offering to each Customer Segment?
Which customer needs are we satisfying?

CHARACTERISTICS
Newness
Performance
Customization
"Getting the Job Done"
Design
Brand/Status
Price
Cost Reduction
Risk Reduction
Accessibility
Convenience/Usability

Customer Relationships

What type of relationship does each of our Customer Segments expect us to establish and maintain with them?
Which ones have we established?
How are they integrated with the rest of our business model?
How costly are they?

EXAMPLES
Personal assistance
Dedicated Personal Assistance
Self-Service
Automated Services
Communities
Co-creation

Channels

Through which Channels do our Customer Segments want to be reached?
How are we reaching them now?
How are our Channels integrated?
Which ones work best?
Which ones are most cost-efficient?
How are we integrating them with customer routines?

CHANNEL PHASES:
1. Awareness
How do we raise awareness about our company's products and services?
2. Evaluation
How do we help customers evaluate our organization's Value Proposition?
3. Purchase
How do we allow customers to purchase specific products and services?
4. Delivery
How do we deliver a Value Proposition to customers?
5. After sales
How do we provide post-purchase customer support?

Customer Segments

For whom are we creating value?
Who are our most important customers?

Mass Market
Niche Market
Segmented
Diversified
Multi-sided Platform

Cost Structure

What are the most important costs inherent in our business model?
Which Key Resources are most expensive?
Which Key Activities are most expensive?

IS YOUR BUSINESS MORE:
Cost Driven (leanest cost structure, low price value proposition, maximum automation, extensive outsourcing)
Value Driven (focused on value creation, premium value proposition)

SAMPLE CHARACTERISTICS:
Fixed Costs (salaries, rents, utilities)
Variable costs
Economies of scale
Economies of scope

Revenue Streams

For what value are our customers really willing to pay?
For what do they currently pay?
How are they currently paying?
How would they prefer to pay?
How much does each Revenue Stream contribute to overall revenues?

TYPES:
Asset sale
Usage fee
Subscription Fees
Lending/Renting/Leasing
Licensing
Brokerage fees
Advertising

FIXED PRICING
List Price
Product feature dependent
Customer segment dependent
Volume dependent

DYNAMIC PRICING
Negotiation(bargaining)
Yield Management
Real-time-Market

www.businessmodelgeneration.com

Erklärung

Hiermit versichere ich, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)

Erklärung

Hiermit versichere ich, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Datum: .....

(Unterschrift)