TUMLogo_oZ_Outl_SW

FAKULTÄT FÜR INFORMATIK  
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Probably An Island

Projekt für Design und Umsetzung von Planspiele im Wintersemester 2016/17

Dokumentation

## Projektmitglieder

Jean-Luc Etgen, Manuel Neuberger, Felix Wohnhaas, Alexander Schuster, Maximilian Mayer, Maximilian Hennig, Kagan Batuker, Gilles Tanson, Alexander Borchers

## informat_blau_grBetreuer

Matthias Utesch, Loina Prifti

# Allgemeines Konzept

Das Spiel „Probably an Island“ wurde von Grund auf als Lernspiel für Wahrscheinlichkeit konzipiert. Im Fokus waren hier eine kontinuierliche Spielerfahrung und eine unterbewusste Lernerfahrung – uns war wichtig, dass das Konzept der Wahrscheinlichkeit intuitiv klar wird, und der Spielende nicht direkt durch Aufgaben damit konfrontiert wird – viel mehr wird man in dem Spiel dazu angeregt, unterbewusst die korrekte, da wahrscheinlichste Lösung zu finden.

Die Zielgruppe des Spiels waren Schüler der 6. bis 8. Klasse. Das Spiel wurde auch einigen Schülern vorgestellt und aufgrund von verschiedenen Problemen, die im Entwicklungsprozess nicht auffielen, weiter angepasst – ein erwachsener Student kann sich nicht immer in die Denkweise eines 12- bis 16-jährigen hineinversetzen.

## Story

In dem Spiel dreht sich alles um eine Insel, auf welcher der eigene Charakter sich begibt, um einen Schatz zu finden. Dabei müssen einige Rätsel gelöst werden, um schlussendlich an den begehrten Schatz zu kommen. Die Zeit spielt hierbei eine Rolle – umso schneller der Spieler die Rätsel löst, umso besser ist der gefundene Schatz.

## Rätsel

Das Spiel beinhaltet vier Rätsel, die jeweils verschiedene Ansätze bieten, über Wahrscheinlichkeit auf eine Lösung zu kommen. So wird beim ersten Spiel mit Würfeln gearbeitet, das zweite Spiel verwendet Zahnräder. Im komplett eigens konzipierten dritten Spiel werden verschiedenfarbige Felder und verwendet, um Wahrscheinlichkeiten darzustellen. Im letzten Spiel ist vor allem die Zeit ein weiterer wichtiger Faktor, um das Stressdenken und schnelle Entscheidungsfindungen anzuregen.

## Level Design

Um ein abenteuerliches und interessantes Setting zu vermitteln, wurde mit Tempel-Elementen, vielen verschiedenen Pflanzenarten und unterschiedlichsten Lichteffekten gearbeitet. Auch die variierende, selbst erzeuge Musik spielt hierbei eine sehr wichtige Rolle. Die Insel wurde extra so konzipiert, dass das unbeabsichtigte Übergehen von Rätseln schon durch die Levelstruktur unterbunden wird. So ist nach jedem Rätsel ein unüberwindbares Hindernis angebracht, welches dem Spieler keinen Fortschritt ermöglicht, wenn das Rätsel nicht vorher abgeschlossen wird. Konkret wird hier mit einer Brücke, Klippen und einem See gearbeitet.

Die Insel lässt sich theoretisch von dem Spieler nahezu komplett erkunden, der Level ist aber so aufgebaut, dass der Spieler den korrekten Weg sicher findet – sowohl durch Texturen, als auch durch Geländeaufbau, Flora und Fauna. Dieses Design vermeidet Frustration, gibt dem Spieler aber trotzdem ein Gefühl der Freiheit.

# Würfelspiel

## Didaktisches Ziel

Das Spiel soll den Schüler durch den spielerischen Umgang mit Würfeln, ein Gefühl für Wahrscheinlichkeiten vermitteln.

## Aufbau

Nach dem der Spieler vom Strand aus, eine große Ruinenlandschaft durchquert hat erblickt er eine große Tafel mit grüner Schrift, sowie fünf Hebeln (wobei 4 davon einen sechsseitigen Würfel auf der Oberseite haben). Hinter den Hebeln befindet sich eine Falltür, auf deren gegenüberliegenden Seite sich eine verschlossene Tür ist. Über der ganzen Szene thronen 4 Totenschädel mit geschlossenen Mündern.

## Spielziel

Der Spieler muss, mit Hilfe der Würfel die Aufgabe auf der Tafel erfüllen, worauf hin die nächste angezeigt wird. Nach dem insgesamt 3 Aufgaben erfolgreich bewältigt wurden, erscheint auf der Tafel ein Hinweis für das nächste Rätsel und die Tür öffnet sich.

## Würfeln

Um zu würfeln muss der Spieler zuerst die Würfel auswählen die gewürfelt werden sollen. Dies tut er indem, er einen der Hebel, mit Würfel auf der Oberseite, auswählt (linke Maustaste drücken während das Fadenkreuz auf den Hebel zeigt). Das der Hebel nun nach unten zeigt sowie, dass der dem Hebel gegenüberliegende Totenschädel nun geöffnet ist, zeigen an das dieser Würfel ausgewählt wurde. Durch ein erneutes auswählen des Hebel wird die Auswahl dieses Würfels rückgängig gemacht.

Nach dem der Spieler so viele Würfel ausgewählt hat wie er werfen möchte, betätigt er den mittleren Hebel (ohne Würfel auf der Oberseite), wodurch entsprechend viele Würfel auf die geschlossene Falltür fallen. Sobald die Würfel still liegen werden diese ausgewertet und es wird überprüft ob die aktuelle Aufgabe erfüllt wurde. Sollte dem so sein wird die entsprechend nächste Aufgabe angezeigt.

Durch ein erneutes aktivieren des Hebels öffnet sich die Falltür und die Würfel verschwinden. Kurz darauf, schließt sich die Falltür wieder. Die Auswahl an Würfeln bleibt nach dem Wurf erhalten.

## Lösungen für die Aufgaben

Da die Aufgaben aus Wahrscheinlichkeit basieren ist der Ausgang eines Wurfes nicht deterministisch. Die hier angegebenen Lösungen entsprechen dem Aufbau welcher die höchste Wahrscheinlichkeit bietet die Aufgabe zu lösen. Die Lösung ist auch mit einem nicht optimalen Aufbau möglich.

1. *Aufgabe („Würfel eine 7“)*

Damit die Summe der Augenzahlen 7 ergibt, sollte der Spieler im optimal Fall 2 Würfel auswählen.

1 Würfel -> Nicht Möglich

2 Würfel -> 6/36 -> 1/6 -> 16,7%

3 Würfel -> 15/216 -> 5/72 -> 6,9%

4 Würfel -> 20/1296 -> 5/324 -> 1,5%

1. *Aufgabe („Würfel mindestens zwei 5er“)*

Die Aufgabe soll ein Gefühl für das Wort „mindestens“ vermitteln. Die Lösung ist natürlich so viele Würfel wie möglich zu verwenden. In diesem Fall also 4.

1. *Aufgabe („Würfel maximal eine gerade Zahl“)*

Ähnlich wie in Aufgabe 2, soll dem Spieler hier ein Verständnis für das Wort maximal vermittelt werden. Da man mit einem Würfel nie mehr als eine gerade Zahl würfeln kann ist dies hier natürlich die optimale Lösung.

# Zahlenkombinationsspiel

## Funktionsweise

Das Spiel besteht aus vier Zahnrädern, die mit der richtigen Zahlenkombination eine Tür öffnen. Auf zwei Schildern werden dem Spieler Hinweise gegeben, die zum Lösen des Rätsels nötig sind. Drei Zahlen kann der Spieler mithilfe dieser Hinweistexte selbst ermitteln. Die vierte Zahl bekommt der Spieler als Hinweis nach dem Lösen des vorangegangenen Spiels gegeben. Der Spieler ändert die Zahl auf dem Zahnrad, indem er auf dieses klickt. Jedes Zahnrad zeigt die Zahlen 1 bis 9 und sie drehen sich nur in eine Richtung. Wenn der Spieler die richtige Zahlenkombination eingibt, öffnet sich die Tür und er kann zum nächsten Abschnitt des Spiels gelangen.

## Hinweistext auf den beiden Schildern

Schild 1: Für die ersten beiden Zahnräder, rechne dir die Anzahl an Möglichkeiten aus, die du bei drei Zahnrädern hast. Rad 1 entspricht der ersten Stelle deines Ergebnisses, das Rad 2 entspricht der letzten Stelle des Ergebnisses. Hinweis: Wie viele Möglichkeiten hat man bei einem Zahnrad? Wie erhält man dann die Anzahl an Möglichkeiten bei drei Rädern?

Lösung: 9 \* 9 \* 9 = 729 => Rad 1: 7, Rad 2: 9

Schild 2: Die Kombination von Rad 3 entspricht der Wahrscheinlichkeit hier die richtige Zahl zu erhalten. Multipliziere diese Wahrscheinlichkeit mit 18, um die richtige Kombination zu erhalten. Hinweis: Wie viele Zahlen des Rads sind die richtige Lösung?

Lösung: 1/9 \* 18 = 2

## Implementierung

Jedes Rad besitzt eine eigene Rad-Logik, welche das Rad rotiert, die Zahl erhöht, wenn es gedreht wurde und die aktuelle Zahl an die übergeordnete Logik weitergibt. Die Zahlen sind dabei nicht an dem Objekt orientiert, sondern nach der Anzahl der Rotationen. Wenn ein Rad gedreht wird, ist klar, dass auch die Zahl sich erhöht hat. Wenn man die 9 erreicht hat und bei einem weiteren Drehen wieder die 1 erscheint, wird die Zahl zurückgesetzt. Jedes Rad ist dabei eine Repräsentation der aktuellen Zahl in einem Array. Die übergeordnete Logik überprüft die Kombination mit den Zahlen jedes Rads, ob diese mit der Lösung des Rätsels übereinstimmen. Bei der richtigen Kombination öffnet sie die Tür und setzt dafür auch die nötigen Töne. Über Singletons werden die einzelnen Werte der Räder und auch jeder weitere Wert oder Boolean gespeichert, die für eine korrekte Funktionsweise nötig sind. So kann auch von außerhalb jede Funktion auf die Werte zugreifen.

Bei der Vorgehensweise bin ich zuerst von einem einzelnen Rad ausgegangen und habe mir überlegt, was ich tun muss damit sich das Rad dreht und wie ich die Zahlen dabei setzen kann. Denn ich wollte das Spiel so entwickeln, dass man beliebig viele Zahnräder hinzufügen oder entfernen kann. So konnten wir die Zahnräder auch wieder beim letzten Rätsel wiederverwenden und mussten nur ein wenig die Logik verändern, damit es für die neuen Anforderungen funktioniert hat. Danach habe ich die Logik um die Zahnräder herum implementiert, da ich wusste, wie viele Räder ich habe und wie ich auf die Zahlen zugreifen kann. Eine einfach if-Abfrage konnte dann überprüfen, ob der Spieler die richtige Zahl ausgewählt hat.

## Probleme

Die einzigen Probleme, die meine Implementierung verursacht hat war, dass ich die Rotationen zuerst mit EulerAngles umgesetzt habe. Da diese aber nicht mit einer einfachen Abfrage überprüft werden können, da sie eine andere Rotation verursachen, als eine einfache Drehung des Objekts anhand von vorgegebenen Gradzahlen, drehte sich das Rad entweder zu weit oder zu wenig und hat außerdem unterschiedliche Drehgeschwindigkeiten erhalten. Als ich dann das gesamte Object mit der Quaternion Drehung gedreht habe, konnte ich abfragen, um wie viel Grad sich das Rad gedreht hat. Also konnte ich die Räder an exakte Positionen drehen und auch die Zahlen updaten. Danach gab es keine weiteren nennenswerten Probleme.

# Color-Game

## Zielsetzung und Beschreibung

Die Zielsetzung des dritten Minispiel, im folgenden Color-Game genannt, war es neben einem Spiel mit einer Verbindung zu Wahrscheinlichkeiten ein eigenständiges Rätsel auszuarbeiten, das autonom als Spiel funktionieren kann. Dabei kann der Spieler selber Entscheidungen treffen die den gesamten Verlauf des Minispiels beeinflussen sollten. So wurde die Designentscheidung getroffen, dass das Spiel aus mehre Runden bestehen soll.

Als Spielmechanismus wurde ein Programmier-Mechanismus aus Brettspielen wie „Lords of Xidit“, „Robo Rally“ oder „AquaSphere“ als Vorbild genommen. Bei diesen Spielmechanismus soll der Spieler seine Spielzüge nicht einen nach den anderen ausführen und sofort Rückmeldung erhält, sondern erst alle seine Aktionen festlegen bevor diese durchgeführt werden. Um eine Konfliktkomponente einzuführen, soll abwechselnd mit dem Spieler das Programm selbst Spielzüge festlegen, die sich auf die Entscheidungen des Spielers auswirken sollen.

Die Aufgabe ist es dabei auf einem Spielbrett, bestehend aus 16 als Raster angeordneten Feldern, eine Spielfigur von einem Startfeld auf ein Zielfeld zu bewegen. Dabei sollte der Spieler, sowie das Programm je 4 Bewegungen festlegen. Die Schwierigkeit für den Spieler besteht darin mögliche Bewegungen der Computerzüge auszugleichen. Da alle Schritte erst nach der Wahl der letzten Bewegung ausgeführt werden, ist es Aufgabe des Spielers sich die mögliche momentane Position der Spielfigur auf dem Spielfeld durch die zuvor getroffenen Entscheidungen herzuleiten und zu merken. Um das einzelne Minispiel nicht zu lang und schwierig zu gestalten, wurde festgelegt dass die Spielerin nicht mehr als 4 Entscheidungen treffen sollte, was mit den zusätzlichen Bewegungen des Programmes zu 8 Bewegungen in einer Spielrunde führten.

Bei der Auswahl der möglichen Farbtafeln entschied man sich für einen festen Satz an Bewegungsmöglichkeiten von denen bei Spielstart zufällig 4 gewählt werden. Dies mindert die Aussicht der möglichen Situationen in denen der Spieler nicht gewinnen kann. So setzt sich der Spieler beim Color-Game nicht nur mit Wahrscheinlichkeiten auseinander sondern baut seine Fähigkeiten Vorauszuplanen aus und schult sein Gedächtnis.

Um den Schwierigkeitsgrad nicht weiter zu erhöhen, soll die mögliche Auswahl des vom Programm gewählten Feld so erkennbar wie möglich sein, ohne das Spielelement unwirksam zu machen. So war es wichtig in der späteren Umsetzung dafür zu sorgen, dass der Spieler auf einen Blick alle Informationen sehen kann. Ein wiederholtes Wechseln der Blickrichtung um das Spielfeld oder die Auswahl zu sehen sollte vermieden werden, da dieses den Denkprozess unnötig erschweren würde. Eine Variation ist möglich aber für den ersten Kontakt mit den Spiel und der Regeln unnötig und kann an einer anderen Stelle für einen höheren Schwierigkeitsgrad genutzt werden, nachdem das Spiel schon einmal absolviert wurde.

## Entwicklungsstufen des Color-Game

Für die ersten Versionen wurden Konzeptzeichnungen und Beschreibungen des Spielablaufs angefertigt. Danach wurde ein spielbarer Papierprototyp erstellt der auch die Rolle des Programms simuliert. Dieser besteht aus einem Spielfeld, Richtungsmarker, einen Kartenstapel aus 8 Karten mit den Bewegungswahrscheinlichkeiten, sowie Würfel für die zufällige Bestimmung des Start- und Zielfeldes. Die Karten repräsentieren dabei die Wahrscheinlichkeiten welche Richtung das Programm wählt, abhängig davon auf welchem Feld sich die Figur zu dem Zeitpunkt an dem die Karte ausgeführt wird befindet. Die Wahrscheinlichkeiten werden dabei als Farben Grün, Blau, Orange und Magenta dargestellt. Dies ist auch die Prioritäten-Reihenfolge in der sich das Programm für eine bestimmte Bewegungsrichtung entscheidet. Sie wurde festgelegt um das Spiel einfacher zu halten. Falls mehrere Felder der höchsten Priorität zur Wahl stehen, wird eines der Felder zufällig gewählt.

Der Spielablauf konnte mit diesem Prototyp, wie folgt, dargestellt werden. Nach der Bestimmung des Start- und Zielfeldes mit den Würfeln zieht der Spieler abwechselnd eine Bewegungsrichtung und eine Farbkarte von dem vorher gemischten Kartenstapel. Nach der letzten Wahl wird die Spielfigur entsprechend der gewählten Richtungen auf dem Spielfeld bewegt. Sollte sich die Spielfigur nach der letzten Bewegung auf dem Zielfeld befinden, gilt das Spiel als gewonnen, ansonsten kann eine neue Runde gespielt werden.

Der erste digitale Prototyp wurde wie das Endprojekt in Unity programmiert. Dabei war die erste Version eine Umsetzung des Papierprototypens bei der der Spieler alle Spielkomponenten von oben herab betrachtet. Die Interaktion beschränkte sich bei dieser Version auf das wählen einer Bewegungsrichtung mit den Pfeiltasten und das Bestätigen der Wahl mit der Return Taste.

Eine spätere Version wurde als ein eigenes Level in eine Inselumgebung umgesetzt mit der von Unity zur Verfügung gestellten First Person Kontroller und für das Projekt erstellten Funktionen für die Interaktion mit der Umgebung. Durch die neuen Interaktionsmöglichkeiten wurde das wählen der Bewegungsaktion durch Tasten aus dem Spiel entfernt. Durch die neue Bewegungsfreiheit des Spielers musste die einzelnen Spielelemente neu angeordnet werden. Diese sollten in die Umgebung eingearbeitet werden. Zugleich sollte der Spieler bei der Auswahl seiner Bewegungen das Spielfeld, sowie die Farbtafel im Blickfeld haben.

In der nächsten Version wurden das neue 3D Modelle für die Schalter der Bewegungsauswahl eingesetzt. Diese Version des Spieles konnte dann in das Gesamtprojekt exportiert werden und wird dort weiter bearbeitet.

Durch das Gewinnen des Spiels sollte sich ein Weg für den Spieler eröffnen um zum nächsten Gebiet der Insel und dadurch zum nächsten Rätsel zu kommen. Das Hindernis das es zu überwinden galt war ein See der nicht betreten werden konnte. Das Spielbrett wurde dazu in diesen See gesetzt. In der ersten Version stiegen, sobald der Spieler das Spiel gewann, Steinplatten aus dem Wasser. Über diese konnte sich der Spieler bewegen. Da diese aber keinen durchgehenden Weg darstellten musste der Spieler an der richtigen Stelle springen um auf die andere Seite zu kommen. Dies stellte ein weiteres Spielelement dar.

Im Laufe der Entwicklung wurde die Möglichkeit des Springens entfernt um zu verhindern, dass Spieler an ungewünschte Positionen gelangen konnte. Entsprechend musste der Weg der zum nächsten Areal führte bei dem dritten Rätsel geändert werden. Dabei entschied man sich ein Boot einzubauen, das sobald der Spieler das Spiel erfolgreich beendet hat, ihn auf die andere Seite bringen würde. Dies sollte automatisch passieren sobald der Spieler das Boot aktivierte. Der Spieler hat dabei keinen Einfluss auf den Weg den das Boot einschlägt und kann sich bis zur Ankunft auf der anderen Seite nicht bewegen. Die Landung auf der anderen Seite des Sees beendet das dritte Rätsel und der Spieler ist frei sich noch etwas in der Umgebung umzusehen bevor er sich zum letzten Rätsel des Spiels aufmacht.

## Umsetzung im Code

Da es eine Designentscheidung war ein festes Set von Farbfeldern zu nehmen die die Wahrscheinlichkeiten für die Bewegung des Programms repräsentieren, sind diese fest im Code verankert. Zu diesem Zweck ist jede Karte mit der entsprechenden Farbposition als Array repräsentiert. Vier Karten wurden umgesetzt, da es mit der zufälligen Wahl der Reihenfolge und die des Ziel- und Startfelds genug Variation gibt, so dass sich kein Spiel wie das vorherige für den Spieler anfühlt.

In der ersten digitalen Umsetzung wurde im ersten Schritt an der Bewegung gearbeitet. Zu diesen Zweck war es eine Funktion zu schreiben die eine Position übergeben bekommt und die Spielfigur entsprechend bewegt. Da es eine feste Anzahl an Positionen gibt, die abhängig von der Position des Spielbrettes sind, wurden die Positionen fest in den Code gebunden.

Eine weitere wichtige Funktion war es die Bewegungswahl abzuhandeln. Die Wahl der Bewegung des Spielers wird nach jeder Interaktion aktualisiert. Da die Wahl immer aus dem Set der Bewegungen nach links, rechts, oben und unten besteht und die Auswahlschalter immer in der gleichen Ausgangssituation starten, hat man sich dazu entschieden durch dieses Set zu iterieren. Bei jeder Interaktion wird eine Animation der Schalter abgespielt und die aktuelle Auswahl gespeichert. Diese gewählte Bewegung wird weitergegeben sobald der Spieler die Wahl mit einem weiteren Schalter bestätigt. Dabei wird das aktuelle Paar aus Schaltern für die Bewegungswahl und Bestätigungsschalter deaktiviert und das nächste Paar aktiviert.

Bei der Auswertung welche Position vom Programm gewählt wird, werden zuerst alle benachbarten Felder in einen Array zwischengespeichert. Dabei werden die Wahrscheinlichkeiten der Farben als Integer gespeichert, die kleinste Zahl entspricht dabei der höchsten Wahrscheinlichkeit. Die Felder werden sortiert und das mit der höchsten Wahrscheinlichkeit zurückgegeben. Als Sortieralgorithmus hat man sich für einen Bubblesort entschieden, da es sich um eine sehr geringe Menge an Daten handelt. Der Bubbelsort legt bei dieser Menge an Daten eine vertretbare Laufzeit an den Tag, die bei dem Minispiel dem Spieler nicht auffallen wird und deshalb nicht ins Gewicht fällt.

Alle gewählten Positionen, durch den Spieler oder des Programmes, werden in eine Queue gespeichert. Die nächste Position wird dabei durch die aktuelle Position und die getroffene Wahl der Bewegung berechnet. Diese Queue wird nach der letzten Auswahl abgearbeitet und die Spielfigur nacheinander um jeweils eine Position bewegt.

Nach der letzten Bewegung wird überprüft ob die momentane Position der Figur mit dem Zielfeld übereinstimmt. Falls dieses nicht der Fall ist, wird das Spiel zurückgesetzt und der Spieler kann erneut spielen. Dabei bleibt das Zielfeld immer gleich und das Startfeld der Spielfigur ist das Feld der letzten Bewegung der vorherigen Runde. Hat der Spieler hingegen das Spiel gewonnen, werden die Schalter deaktiviert und die mögliche Interaktion mit dem Boot aktiviert.

Der Transport des Spielers nach dem Aktivieren des Bootes wird mit einer Animation in Unity umgesetzt. Die Fähigkeit des Spielers sich zu bewegen wird für diesen Zeitraum außer Kraft gesetzt und wieder aktiviert sobald die komplette Animation abgelaufen ist.

# Endrätsel – Platten mit Nummern

## Konzept

Das Konzept vom letzten Puzzle war ein Rätsel zu entwickeln, wo die Umgebung das Puzzle ist. Ursprünglich war gedacht alle 3 vorherigen Rätselspiele in einem zu vereinen, doch der Aufwand wäre für so ein Rätsel zu viel gewesen.

## Erklärung

Das Ziel vom Spiel ist auf die andere Seite des Spielfelds zu kommen, da dort der Schatz ist. Je schneller der Spieler es schafft die Schatztruhe zu öffnen, desto mehr Geld/Punkte bekommt er. Der spieler kann maximal 500$ und verliert 100$ pro Runde, jedoch bekommt er immer mindestens 10$, wenn er zu lang braucht.

Das Spielfeld besteht aus einem Feld von Platten, das fünf Platten breit und acht tief ist. Auf jeder Platte steht eine Nummer von 1 bis 9. Über dieses Feld von Platten muss der Spieler drüberlaufen, um auf die andere Seite zu kommen. Weiterhin sind am Eingang zum Tempel vier Zahlenräder, die der Spieler anhalten kann, wo auch die Nummern 1 bis 9 draufstehen.

Damit der Spieler eine Platte betreten kann ohne runterzufallen muss er erst die entsprechende Nummer, die auf der Platte steht, bei den Zahlenrädern einstellen. Das heißt, wenn eine 5 auf der Platte steht, die man betreten will, muss mindestens bei einem der Zahlenrädern eine 5 stehen. Runterfallen ist nicht schlimm, da der Spieler dann wieder zu den Zahlenräden versetzt wird; die heruntergefallene Platte verbleibt jedoch bis zum Ende der Runde im Abgrund. Die „sicheren“ Zahlen sind auch nochmal direkt über der Schatztruhe am Ende des Spielfeld hinterlegt.

Dazu kommt noch, dass das Spiel rundenbasiert ist. Jede Runde ist eine Minute lang und am Anfang ändern sich jedes Mal die Nummern auf den Platten. Dadurch wird verhindert, dass ein Spieler nur an den Zahlenrädern rumprobiert, bis er sich einen sicheren Weg erstellt hat.

## Implementierung

Alle 3D Models und ihre Animationen außer den Steinen an der Seite der Arena wurden von Alexander Schuster erstellt. Das Terrain und die Einrichtung des Levels, d.h. Lighting und Level Design, hat Felix Wohnhaas gemacht. Die gesamte Spielelogik wurde von Jean-Luc Etgen implementiert, wobei die Zahlenradlogik von Gilles Tanson erstellt wurde. Die Hintergrundmusik wurde von Kagan Batuker und die Soundeffekte wurden von Maxi Mayer erstellt.

# Leveldesign & 3D-Assets (Alexander Schuster)

## Allgemein

Die Spielekonzeptionierung wurde innerhalb des gesamten Teams besprochen und ausgearbeitet. Insgesamt haben wir uns auf vier Rätsel festgelegt, welche auf einem Inselsetting mit „archäologischem“ Flair verteilt werden sollten. Für die detaillierte Ausarbeitung, sowie der Umsetzung, haben wir uns in Gruppen mit festem Aufgabenbereich eingeteilt.

## Ablauf

Die im Team mir zugewiesene Arbeit bezog sich auf das Design des Spiels. Dies umfasst zum einen das Entwickeln von spezifischen Assets, welche für das Funktionieren eines Rätsels benötigt werden, als auch das Erstellen der Spieleumgebung. Im Verlauf der Umsetzung habe ich mich zuerst auf Asset Wünsche meines Teams konzentriert um eine erste Grundlage für die verschiedenen Rätsel zu schaffen. Nach Abschluss dieser Arbeit habe ich mich um das grundsätzliche Design gekümmert, welches ich anhand meiner Mitschriften aus der Vorbereitungsphase entwickelt habe. Aufgrund des verhältnismäßig knappen Zeitraums für das Projekt habe ich mich dazu entschieden die Umgebung modular aufzubauen. Dies hat den Vorteil, dass man Asset-Arbeiten relativ schnell in den Entwicklerprogrammen erledigt, wodurch man mehr Arbeitszeit in die Szenenentwicklung in Unity investieren kann. Zudem vereinfacht das modulare Aufbauen einer Szene die Planung in Unity, allerdings führt dies auch zu einer schlechteren Performanz aufgrund der erhöhten Polygonanzahl und einer repetitiven Umgebung.

## Probleme

Die Benutzung von Git zusammen mit Unity hat sich als problematisch herausgestellt, da es beim Level Aufbau zu folgenreichen Merge-Konflikten in Unitys internen Skripten kommt. Deswegen rate ich bei Fortführung des Projekts ein ausgiebiges Setup von Git für die Benutzung von Unity. Die dabei entstandenen Probleme konnte ich aufgrund des Zeitmangels leider nur Notdürftig lösen. So musste ich einige der modularen Teile der Tempelanlagen zusammenfassen, wodurch sich das Abändern solcher Gruppen als sehr arbeitsintensiv gestalten dürfte.

## Weiterentwicklung

Bevor das Projekt weiterentwickelt werden kann, sollte die Ordnerstruktur für die Game Assets neu angelegt, und übersichtlicher gestaltet werden. Zudem können, für eine verbesserte Performanz, die Collisionboxes für die modularen Assets angepasst werden. Außerdem haben sich die abgeschrägten Kanten der Bodenplatten als eher hinderlich für das Modulare designen erwiesen.

# Sounddesign

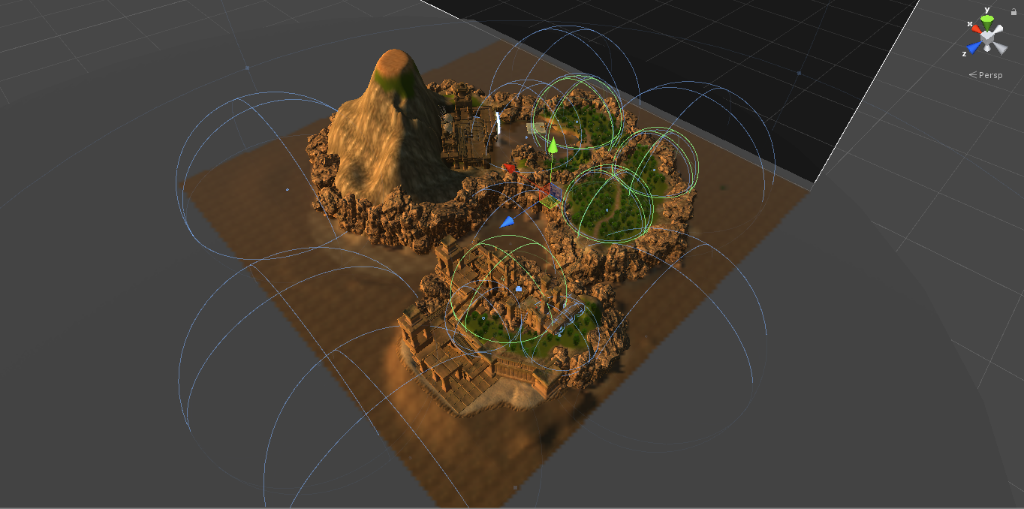
## Main Themes und Backing Tracks

Da das Spiel von Grund auf in einzelne Teile, nämlich die vier Rätsel, eingeteilt war, kam der Grundgedanke auf, für jedes Rätsel ein eigenes Lied zu kreieren. Wenn der Spieler zu Anfang strandet, spielt noch kein Lied. Die entsprechenden Songs starten immer abhängig von der Position des Spielers und seiner Nähe zu dem nächsten Rätsel. Die Lieder werden mittels eines Crossfades gewechselt, um keine Unterbrechungen zu erzeugen. Die Organisation übernimmt das Audio Handler Skript, dass in der Szene am Environment Audio Gameobject hängt. Für die vier Rätsel wurden folgende Lieder komponiert:

* Würfelrätsel – „Island Theme“
* Zahlenschlossrätsel – „Island Groove“
* Brettspiel – „Island Riddle“
* Finales Rätsel – „Island Darkness“

Jedes der Lieder wurde in „FL Studio 12“ am PC erstellt und eigens für das Spiel komponiert. Dabei wurde versucht, das Flair einer einsamen Tropeninsel einzufangen, indem vor allem Instrumente wie Rasseln, Stockschläge, Mundharmonika, Xylophon, Glocken oder Panflöten verwendet wurden. Die Lieder sind außerdem so aufgebaut, dass sie nahtlos in Schleife gespielt werden können, um die Immersion eines endlosen Hintergrundliedes aufrecht zu erhalten. Dabei wurde aber dennoch auf Höhepunkte in den Liedern nicht verzichtet. Weiterhin reflektieren die Songs die aktuelle Position des Spielers auf der Karte. Z.B. verwendet „Island Theme“ einen Synthesizer, der langsam und gemächlich einen Akkord nach dem anderen im immer selben Rhythmus spielt. Dies symbolisiert die Wellen am Strand, die auf Sandbänke auflaufen. „Island Groove“ hingegen setzt auf mehr Rhythmus durch Holzinstrumente, was die Position des Spielers im Wald, in dem sich das zweite Rätsel befindet, widerspiegelt. „Island Darkness“ verwendet einen Synthesizer, der immer auf die eins spielt und somit die tickende Zeit darstellt, die den Spieler im letzten Rätsel einschränkt. Die Flöte in „Island Riddle“ spielt zudem die Vögel, die am Rande des Sees des dritten Rätsels zwitschern.

## Environment Sound Effects

Umgebungseffekte wurden auf der ganzen Insel verteilt, um die Immersion einer tropischen Insel weiter zu vertiefen. Einerseits kann man am Strand das Rauschen des Meeres vernehmen, andererseits zwitschern im Wald Vögel und man kann Enten am Ufer des Sees hören. Dazu wurden verschiedene Audioquellen auf der Insel verteilt, welche nebenstehender Grafik zu entnehmen sind. Dabei stellen die großen blauen Sphere-Bounding-Volumes die Ozean Sound Bereiche, die kleineren blauen die Wald Sound Bereiche und die kleinen grünen die Theme Song Bereiche dar. Je weiter man vom Mittelpunkt einer Quelle entfernt ist, desto leiser wird diese Quelle für den Spieler.

## Weitere Sound Effekte

Für sich bewegende Objekte in der Welt wurden eigens aufgenommene und zusammengestellte Effekte kreiert. Dabei sind folgende Effekte entstanden:

* 12 verschiedene Sounds, wenn ein Würfel auf dem Boden aufschlägt
* 2 Sounds für stoppende Würfel (mit Aufnahmeprogramm abmikrofoniert)
* Gong Effekt für das Reset-Signal im finalen Rätsel
* Zahlenschloss Rattern und Klicken
* Riegel Öffnen Geräusch
* Hebel Geräusch für alle zu aktivierenden Hebel
* „Knirschgeräusch“ für sich verschiebende Steinplatten/ Statuen
* Falltür-Klappgeräusch
* Eigens komponiertes Schatztruhen Jingle, erstellt mit FL Studio 12

Alle Effekte wurden mit einem Rode Mikrofon und Audacity aufgenommen und dann nachbearbeitet oder zusammengeschnitten. Viele Geräusche sind das Ergebnis der Komposition anderer Geräusche aus der Welt, z.B. ist das Falltür Klappgeräusch durch eine schlagende Tür eines Gartenhäuschens und das Quietschen einer Haustüre entstanden. Die Soundeffekte wurden dann vom Designer in die Animationen der einzelnen Objekte eingebunden, oder vom Engineer zur rechten Zeit abgespielt.

Schrittgeräusche erzeugen zudem an verschiedenen Stellen auf der Karte Hall, indem Reverb-Zones installiert wurden. Diese finden sich bevorzugt in den Tempeln und in Höhlen.

## Aufgabenverteilung

Für Sound war das Team bestehend aus Kagan Batuker und mir, Maximilian Mayer, zuständig. Wir haben getrennt von den anderen die Musik komponiert und die Sound Effekte auf Bedarf der Designer hin erstellt, sowie diese dann anschließend nach Fertigstellung der grafischen Oberfläche eingebunden.