

# Bachelorarbeit

## Design und Implementierung eines Resource Management Games

**Justin Nicola**

Justin Nicola  
[065369@student.uni-kassel.de](mailto:065369@student.uni-kassel.de)  
Matrikelnummer: 35441124  
8. Fachsemester  
Studienfach: Informatik, PO2018

## **Eidesstattliche Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit den nach der Prüfungsordnung der Universität Kassel zulässigen Hilfsmitteln angefertigt habe. Die verwendete Literatur ist im Literaturverzeichnis angegeben. Wörtlich oder sinngemäß übernommene Inhalte habe ich als solche kenntlich gemacht. Hiermit versichere ich, dass diese Version inhaltlich mit dem vorab elektronisch übersandten Exemplar übereinstimmt.

---

Kassel, 29.09.2022, Justin Nicola

# Glossar

# Abbildungsverzeichnis

1	Screenshot aus Age of Empires ([34]) . . . . .	13
2	Screenshot aus Civilization ([35]) . . . . .	15
3	Screenshot aus Sim City 2000 ([37]) . . . . .	18
4	Game Over in Sim City 2000 ([61]) . . . . .	18
5	Generierter Planet, Screenshot aus RimWorld . . . . .	19
6	Ausgewählter Startpunkt für die Kolonie, Screenshot aus RimWorld . . . . .	20
7	Auswahl der Startkolonisten, Screenshot aus RimWorld . . . . .	21
8	Stimmungsbalken und -einflüsse, Screenshot aus RimWorld . . . . .	22
9	User Interface, Screenshot aus RimWorld . . . . .	23
10	Baumenü, Screenshot aus RimWorld . . . . .	24
11	Prioritäten, Screenshot aus RimWorld . . . . .	24
12	Befehle, Screenshot aus RimWorld . . . . .	25
13	Die Vier Phasen des Interviews . . . . .	25
14	Pollenpakete einer Honigbiene ([25]) . . . . .	33
15	(a) Königin, (b) Arbeitsbiene, (c) Drohne. (K) Kopf, (B) Brust, (H) Hinterleib, (A) Auge, (F) Fühler ([26, S.2]) . . . . .	34
16	Verschiedene Stadien einer Honigbiene, ausgewachsen (links), Puppe (mitte), Larve (rechts) ([25]) . . . . .	35
17	Überblick über den Lebensverlauf einer Biene jeder Kaste ([75]) . . . . .	36
18	Die minimale und maximale Lebenserwartung einer Biene jeder Kaste ([75]) . . . . .	37
19	Aufbau eines Square Grids ([7]) . . . . .	38
20	Aufbau eines Hex Grids ([7]) . . . . .	39
21	Grobe Skizze des Gameplay Loops . . . . .	41
22	Fertig implementierte Prioritätenliste . . . . .	42
23	Ressourcenverlauf und mögliche Umwandlungen . . . . .	45
24	Skizze des Informationsfensters einer ausgewählten Biene . . . . .	46
25	Simplifiziertes UML-Klassendiagramm der Bee.cs . . . . .	47
26	Reproduktionsverhalten der Bienen . . . . .	48
27	Vier verschiedene Stadien des Bienenmodells von links nach rechts: Ei, Adult, Puppe und Larve . . . . .	52
28	Zeitsteuerung Zustandsdiagramm . . . . .	53
29	Erster Prototyp des Spielinterfaces . . . . .	54
30	Start Screen des Spiels . . . . .	56

# Tabellenverzeichnis

1	Age of Empires Eigenschaften ([34, 4, 3, 5]) . . . . .	13
2	Civilization Eigenschaften ([70, 35, 64, 41]) . . . . .	15
3	Sim City 2000 Eigenschaften ([37]) . . . . .	17
4	Jeweilige Antworten der Probanden auf die Einleitungsfragen . . . . .	27
5	Übersicht aller aufgestellten Hypothesen der durchgeföhrten Interviews	31
6	Verfügbare konkrete Ressourcen . . . . .	44
7	Mögliche Traits einer Biene . . . . .	50

# Quellcodeverzeichnis

1	The Sumerian Game, Ausschnitt	8
2	World Generation	40
3	Erstellung einer neuen Anweisung	43
4	Zuweisung einer Biene von einer offenen Anweisung	43

# Design und Implementierung eines Resource Management Games

Justin Nicola, Mat. 35441124, Informatik PO2018

21. September 2022

## 1 Einleitung

### EINLEITUNG SCHREIBEN

#### 1.1 Geschichte

Bereits seit dem Jahr 1964 existieren Ressource Management Games. Das erste verzeichnete Spiel trägt den Titel „The Sumerian Game“, wurde entwickelt von Mabel Addis und spielt in dem antiken sumerischen Stadtstaat Lagash. Die Aufgabe des Spielers war es, in drei verschiedenen Segmenten mit jeweils eigenen Runden, die Bevölkerung und die Ressourcen so anzurichten, dass ein erfolgreiches Überleben der Einwohner gesichert ist, trotz zufälliger Katastrophen oder Events [78]. Ein Ausschnitt des Spielgeschehens kann [Quellcode-Ausschnitt 1](#) entnommen werden. Seitdem hat sich in der Branche des Game Developments und der Kategorie der Resource Management Games einiges getan, von SimCity über Anno, bis hin zu RimWorld oder Factorio. Es gibt verschiedenste Unterkategorien von Resource Management Games, welche alle eigene Siegbedingungen, Tücken und Eigenschaften besitzen, wie auch typische UI-Elemente und -Anordnungen. Das momentan am besten verkauftes Resource Management Game auf der Online Spiele Plattform *Steam* ist *Cities: Skylines* mit einem Nettoumsatz von \$88 Millionen Dollar, und wurde im März 2015 veröffentlicht [54].

```

1 Total population now 500
2 Total farm land under cultivation, acres 600
3 Total grain in inventory, bushels 900
4 one season old 900
5 two seasons old 0
6 three seasons old 0
7 Total grain just harvested, bushels 13000
8
9 Total resources, harvest + inventory 13900
10
11 You must now decide how to use your resources.
12
13 How many bushels of grain do you wish to FEED your people?
14 1000
15 How many bushels of grain do you want PLANTED for the next crop?
16 2000
17 This means that 10000 bushels must be placed in storage. Is this all
18 right? Do you wish to 1-let your decision stand or 2-revise them?

```

*Quellcode-Ausschnitt 1: The Sumerian Game, Ausschnitt*

## 2 Motivation

Videospiele integrieren sich mehr und mehr in den Alltag eines jeden, und ist auch seit meiner Kindheit ein Teil der Freizeit. Das erste Videospiel, das ich jemals spielen durfte, war *Age of Empires*, kurze Zeit später auch *Starcraft II* und *Anno 1701*. Alle diese Spiele haben als zentrale Eigenschaft die Verwaltung von Ressourcen und erfordern damit ein taktisches Geschick und kreatives, effizientes Denken, was mich als Kind faszinierte. Der Erfolg des Spieles stand oder fiel mit der eigenen Effizienz, man musste erst Scheitern und aus den Fehlern lernen, um Fortschritte zu machen und bei dem nächsten Versuch die Hürde zu überwinden. Seitdem begeistert mich dieses Genre mit stetig neuen Ideen und Konzepten, welche neue Herausforderungen und Erfolge bieten. Die Palette der Spiele ist selbstverständlich deutlich zu groß, um in diesen wenigen Seiten erfasst werden zu können, aber diese Arbeit soll einen kleinen Einblick in das Genre an sich bieten, wie auch die Konzepte, die Entwicklung der Ideen und die letztendliche Implementierung.

## 3 Definition

Um die Kategorie der Ressource Management Games zu definieren, ist es essenziell zu erfassen, welche Eigenschaften ein Spiel haben muss, damit es als Ressource Management Game kategorisiert werden kann. Laut [73] ist Ressource Management „[...] the practice of planning, scheduling, and allocating people, money, and technology to a project or program. In essence, it is the process of allocating resources to achieve the greatest organizational value.“. Demnach sind nicht nur die Ressourcen von großer Bedeutung, sondern auch der Prozess des Zuweisens der jeweiligen Ressourcen, um

den daraus größten Nutzen für das derzeitige Ziel zu gewinnen. [55] geht einen Schritt weiter, und beschreibt das Sammeln und Überwachen der Ressourcen als zentrales Element. Außerdem spielt der Informationsgehalt des Spielers eine Rolle, jedes komplexe Resource Management Game „[...] will involve imperfect decisions, leading to interesting strategy.“ [55]. Im Zentrum dieser Kategorie stehen also die *Ressourcen*, das *Verwalten* dieser Ressourcen innerhalb einer *Ökonomie*, und der zu einem gegebenen Zeitpunkt vorhandene *Informationsgehalt* des Spielers.

### 3.1 Ressourcen

Der erste fundamentale Bestandteil sind die *Ressourcen*. Als Stützpfiler von Ökonomien kann jegliches Konzept, was numerisch erfasst und gemessen werden kann als Ressource gelten. Speziell auf Spiele bezogen gibt es also etliche Möglichkeiten eine Ressource zu etablieren. So sind Geld, Energie oder eigene Einheiten verschiedene Ressourcen, aber nach genannter Definition auch gegnerische Einheiten, Zeit und Geschwindigkeit. Statische und nicht interaktive Objekte oder Gebäude hingegen sind keine Ressourcen [1]. Es ist jedoch keine *Kontrollierbarkeit* gefordert, so ist *Zeit* auch eine Ressource, auch wenn diese nie mehr werden kann und vom Menschen nicht beeinflussbar ist.

Eine Ressource gilt als *greifbar* (engl. *tangible*), wenn diese Ressource in der physischen oder virtuellen Welt tatsächlich existiert und wahrgenommen werden kann. Die Ressource besitzt also eine eindeutige Koordinate im Raum. Beispielsweise wären wiederum Einheiten, Bäume oder Eisenminen. Im Gegensatz dazu stehen *nicht greifbare* (engl. *intangible*) Ressourcen. Darunter fallen alle Ressourcen, die keine Koordinate im physischen oder virtuellen Raum haben. Lebenspunkte einer Einheit sind beispielsweise keine greifbare Ressource.

Diese beiden Gegensätze können in Wechselwirkung stehen, so kann ein vom Spieler eingesammelter Gegenstand (greifbar) dazu führen, dass sich die Lebenspunkte erhöhen (nicht greifbar), wodurch der eingesammelte Gegenstand aus dem Raum verschwindet. Dadurch entsteht eine Art Transition von einer greifbaren zu einer ungreifbaren Ressource [1].

Ressourcen können weiterhin gruppiert werden in *konkret* (engl. *concrete*) oder *abstrakt* (engl. *abstract*). Konkrete Ressourcen sind dabei jegliche Form von Ressourcen, welche der Spieler tatsächlich als diese wahrnimmt, also etwa Holz, Lebenspunkte oder Bäume. Abstrakte Ressourcen hingegen werden nicht vom Spieler als solche erkannt und nicht gezeigt, sondern für Hintergrundberechnungen verwendet. Ein klassisches Beispiel einer solchen Ressource ist der Wert der einzelnen Spielsteine beim Schach. Ein Bauer hat einen numerischen Wert von 1, die Königin hingegen einen Wert von 9. Der König hat als einziger Spielstein keinen Wert, da mit dessen Fall das Spiel endet [14]. Diese Werte werden im Spiel nicht thematisiert und auch nicht in dessen Konzept eingebunden, sondern existieren einzig für den Zweck der Wertebestimmung eines Spielers.

### 3.2 Ökonomie

Eine *Ökonomie*, oder auch *Wirtschaft*, ist ein System und „[...] besteht aus Einrichtungen, Maschinen und Personen, die Angebot und Nachfrage generieren und regulieren.“ [18]. Diesem System sind vier verschiedene Funktionen inne [2, 63]:

## Quellen

Als *Quelle* (engl. *source*) wird eine Mechanik beschrieben, welche neue Ressourcen generiert. Ob dabei durch Konditionen ausgelöst oder in kontinuierlichem Fluss ist dabei nicht von Bedeutung. Ein Beispiel einer solchen Quelle ist die natürliche Lebensregeneration von Einheiten, wobei diese Quelle erst mit Erfüllung einer Kondition neue Lebenspunkte generiert, da für gewöhnlich die Lebenspunkte zuerst unter dem Maximum liegen müssen.

## Abflüsse

Als *Abflüsse* (engl. *drains*) werden die zu Quellen gegenteiligen Mechaniken beschrieben. Ein Abfluss entfernt also bestimmte Ressourcen aus dem Spiel. In Ego-Shootern ist die Munition hierfür ein gängiges Beispiel, da geschossene Kugeln in keine andere Ressource umgewandelt werden, sondern durch einen *Abfluss* aus dem Spiel entfernt werden.

## Umwandler

Ein Umwandler (engl. *converter*) ist die Mechanik einer Transition zwischen verschiedenen Ressourcen. So können Bäume in einer bestimmten Relation zu Holz umgewandelt werden. Für gewöhnlich kann auf diese Relation vom Spieler Einfluss genommen werden, in dem durch beispielsweise technologische Verbesserungen mehr Holz pro Baum erhalten werden kann.

## Händler

Die Mechanik des *Händlers* (engl. *trader*) beschreibt den Austausch von Ressourcen. Anders als ein Umwandler werden jedoch keine Ressourcen erschaffen oder zerstört, sondern lediglich vorhandene ausgetauscht. Diesem Austausch liegt eine *Austauschregel* zugrunde, so kann ein beispielsweise in einem virtuellen Laden vorhandener Gegenstand ein gewisser Betrag an Währung kosten. Durch das Durchführen dieses Austausches erhält das Gegenüber dann den in der Austauschregel geforderten Betrag der Währung, und man selber erhält den geforderten Gegenstand.

### 3.3 Informationsgehalt

Ein weiterer Pfeiler von Ressource Management Games ist der zu einem Zeitpunkt gegebene *Informationsgehalt*. Um das Spiel spannender zu gestalten, kann es sinnvoll sein, dem Spieler bestimmte Informationen vorzuenthalten. Der Spieler ist folglich dazu gezwungen, Entscheidungen zu treffen basierend auf den derzeit gegebenen Informationen beziehungsweise dem, was der Spieler glaubt zu wissen [44], was zu interessanten Ansätzen führen kann und Variation in das Spielgeschehen bringt [55]. Ein Beispiel von fehlenden Informationen ist der sogenannte *fog of war*. Ursprünglich aus einem militärischen Kontext stammend hat dieses Wort große Bedeutung in der Gaming Branche gefunden, speziell in Spielen des Genres *Real Time Strategy* (Echtzeitstrategie). Für gewöhnlich ist in diesem Genre der größte Teil der Karte am Anfang verdeckt. Der Spieler

kann sich dazu entscheiden, die Karte zu erkunden und dadurch Informationen zu sammeln, zum Beispiel die Position des Gegners. Fallen bereits aufgedeckte Teile der Karte außerhalb des Sichtfelds der Einheiten des Spielers, so werden sie durch den *fog of war* verdeckt [31]. Der Spieler sieht lediglich den letzten Zustand der Karte, als die Einheiten dort noch Sicht hatten. Um also die Informationen zu aktualisieren ist der Spieler dazu angehalten, erneut Sicht darauf zu erhalten. Diese Mechanik entzieht dem Spieler also bewusst Informationen und spornt dazu an, Entscheidungen über Erkundung und Einheitenplatzierung zu treffen.

Eine weitere Möglichkeit, bewusst Informationen vorzuenthalten und den Spieler zu Entscheidungen zu bewegen sind *Events*. Ein Event im Kontext von Game Design ist eine vorprogrammierte Erfahrung, die dem Spieler widerfahren wird. Welche Auswirkungen das auf das Spielgeschehen haben wird ist nicht inbegriffen, sollte aber bei der Erstellung solcher Events berücksichtigt werden. Dem Spieler wird im Voraus meist jedoch nicht mitgeteilt, welche Events stattfinden werden, in welcher Reihenfolge oder zu welchem Zeitpunkt. Die Kernidee ist, dass der Spieler adaptiv darauf reagiert, wodurch eine hohe Vielfalt an möglichen Spielverläufen entsteht und ein Spiel unter Umständen *Wiederspielbarkeit* (engl. *replayability*) verleiht, sodass der Spieler ein Spiel auch mehrfach spielen möchte und keine Langeweile nach bereits kurzer Zeit einsetzt. Diese Art des Entscheidungzwangs bricht also die Monotonie und birgt Potenzial für Strategie [12]. Ein Beispiel eines solchen Events wäre in einem Städteaufbauspiel eine plötzliche Hungersnot oder der Ausbruch eines Feuers, wodurch der Spieler dazu angehalten wird, Ressourcen umzulagern oder ähnliche Entscheidungen zu treffen.

### 3.4 Verwaltung

Die *Verwaltung* beziehungsweise in diesem Kontext und in dieser Arbeit referenziert als *Management*, stellt den letzten Pfeiler eines Resource Management Games dar. Die Verwaltung von Ressourcen beschreibt die essenzielle Tätigkeit der Verwendung der bereits zuvor erläuterten ökonomischen Funktionen zum Erfüllen eines bestimmten Zwecks. Diese Verwaltung ist jedoch nicht in den ökonomischen Funktionen begrenzt, sondern kann sich auch auf die Positionsveränderung der Koordinaten einer konkreten Ressource beziehen, etwa einer Einheit, die die Position wechselt. Der Spieler besitzt also eine gewisse Freiheit, Entscheidungen zu treffen über die Verwendung der gegebenen Ressourcen. Kombiniert mit dem zuvor betonten *Entscheidungzwang* und der geforderten Adaptivität des Spielers hat der Spieler keine andere Wahl, als zu Verwalten, vorausgesetzt der Spieler setzt sich als Ziel, das Spiel nicht zu verlieren. Bricht also ein Feuer in einer vom Spieler gebauten Stadt aus, und es ist keine Feuerwehr vorhanden, die auf dieses Event reagieren kann, könnte es sinnvoll sein, die vorhandenen Ressourcen, gegebenenfalls Geld, in eine Feuerwehr zu investieren, um damit auf den Entscheidungzwang zu reagieren.

## 4 Beispiele

Für die Sektion der Beispiele werden sich primär Klassiker angeschaut, welche als Grundlage und Überblick des Genres dienen sollen. Als Klassiker werden im folgenden Spiele genannt, welche bereits etwas älter sind, und maßgeblich beteiligt waren an

der Bildung und Entwicklung dieses Genres. Vorwiegend haben die genannten Spiele also großen Anklang in der Community gefunden und wurden sehr gut aufgenommen. Der Zeitraum beschränkt sich folgend auf die Jahre 1980 - 2010. Die genannten Spiele sind dabei lediglich ein Ausschnitt aller vorhandenen Klassiker und Meilensteine des Genres. Ein weiteres Beispiel eines Resource Management Games, welches erst kürzlich (2018) erschien, wird in der folgenden Sektion *Analyse* genauer untersucht und dabei maßgeblich für die Entwicklung des Prototypen sein.

## 4.1 Age of Empires

Das Spiel *Age of Empires* ist ein von *Ensemble Studios* entwickeltes und von *Microsoft* publiziertes, historisches Echt-Zeit-Strategie-Spiel für Einzel- und Mehrspieler, welches in Amerika im Jahr 1997 erschien. Die verwendete *game engine* ist *Genie*, welche hauptsächlich 2D *sprites* verwendet [34]. *Age of Empires* ist dabei der erste Teil der Reihe, mit drei weiteren Nachfolgern *Age of Empires II*, *Age of Empires III* und *Age of Empires IV*, und einem Ableger mit mythologischem Hintergrund *Age of Mythology* [3]. Das Spiel wird in einer isometrischen Perspektive dargestellt (vgl. Abbildung 1). Es gibt verschiedenste Ressourcen und Einheiten, welche verschiedene Taktiken ermöglichen mit wiederum verschiedenen Konterstrategien. Ressourcen sind dabei finit, was bedeutet, dass ein gefällter Baum nicht wieder nachwachsen wird. Eine Kernkomponente des Spiels sind die verschiedenen Zeitalter, in welche der Spieler voranschreiten kann. Damit werden jeweils neue Technologien und Einheiten freigeschaltet, welche auf dem Weg zum Sieg hilfreich sein könnten. Die Zeitalter gliedern sich dabei auf in *Altsteinzeit*, *Jungsteinzeit*, *Bronzezeit* und *Eisenzeit* [34]. Es gibt dabei 12 verschiedene Völker, die an historische Völker angelehnt sind. Diese sind *Ägypter*, *Assyrer*, *Babylonier*, *Chosonen*, *Griechen*, *Hethiter*, *Minoer*, *Perser*, *Phönizier*, *Shang*, *Sumerer* und *Yamato*. Dabei hat jedes Volk eine andere Gesamtauswahl aus dem Technologiebaum. Alle wichtigen Eckdaten zu dem Spiel sind in Tabelle 1 einsehbar.

### Siegbedingungen

- Alle Gegenspieler eliminieren mittels Militär
- Alle *Artefakte* / Runen erobern
- Ein *Weltwunder* bauen und erfolgreich bis Ende verteidigen

### Ressourcen

- Nahrung
- Holz
- Stein
- Gold

[4]

Tabelle 1: Age of Empires Eigenschaften ([34, 4, 3, 5])

Erscheinungsjahr	1997
Entwickler	Ensemble Studios
Publisher	Microsoft
Multiplayer	Ja
Ressourcen	Nahrung, Holz, Stein, Gold
Siegbedingungen	Domination, Artefakte, Weltwunder
Spielbare Völker	12
Perspektive	Isometrisch
Technologien	53



Abbildung 1: Screenshot aus Age of Empires ([34])

## Rezensionen

PC Games	93% [11]
PC Player	5/5 [49]
Power Play	84% [51]
GameRankings	87,1% [27]
IGDB	85% [34]

## 4.2 Sid Meier's Civilization

Das 1991 erschienene Spiel *Sid Meier's Civilization* ist ein von *MicroPose* entwickeltes und publiziertes, rundenbasiertes Strategiespiel [35]. *MicroPose* ist ein 1982 von Bill Stealey und Sid Meier gegründetes Softwareunternehmen, wovon von letzterem auch der Name abgeleitet wird [21]. Man startet im Jahr 4000 B.C. und bewegt sich zeitlich bis ins Informationszeitalter, während man neue Städte gründet, Technologien erforscht und anderen Gegenspielern beziehungsweise Reichen und deren Herrschern begegnet,

Diplomatie führt und gegebenenfalls auch Kriege. Unter den spielbaren Herrschern sind unter anderem *Alexander der Große*, *Napoleon* und *Julius Cäsar*. Das Spiel besitzt einen nicht-linearen Technologiebaum mit den wichtigsten Errungenschaften der Menschheit, darunter das Rad oder Navigation, verschiedene baubare Wunder, zum Beispiel die Pyramiden oder die Große Mauer und besitzt fünf verschiedene Schwierigkeitsgrade, mit denen der Spieler die Herausforderung selber setzen [15]. Je nach Fokus des Spielers ist also jedes Match ein neues, und es gibt durch verschiedene Ressourcen und Entscheidungsmöglichkeiten einige Strategien denen sich der Spieler bedienen kann.

Das Spiel erhielt fünf weitere Nachfolger *Civilization II*, *Civilization III*, *Civilization IV*, *Civilization V* und *Civilization VI* [40], wobei sich manche Teile stark von anderen Unterscheiden, etwa in der Perspektive oder der Geometrie des Spielfeldes. So wurde von *Civilization IV* auf *Civilization V* das Spielfeld von einem *Square Grid* auf ein *Hex Grid* umgestellt [70]. Neben den Nachfolgern gab es einige Ableger, darunter *Sid Meier's Civilization: Beyond Earth*, welches 2014 erschien und zwei Erweiterungen erhielt. Dieser Teil spielt, anders als alle anderen, auf einem fremden Planeten und hat eine Sci-Fi Thematik, statt wie üblich, eine historische [36]. Das Spiel besitzt, im Gegensatz zu allen anderen Nachfolgern, noch eine Vogelperspektive, statt wie später, eine isometrische (vgl. Abbildung 2). Außerdem sind auch politische Auswahlmöglichkeiten vorhanden. Die wichtigsten Eigenschaften sind in Tabelle 2 zusammengefasst.

## Siegbedingungen

- Alle Gegenspieler eliminieren mittels Militär
- Ein Raumschiff bauen und Alpha Centauri als erster erreichen
- Überleben bis die Zeit ausgelaufen ist [64]

## Ressourcen

- Kohle
- Fisch
- Wild
- Wild (Tundra)
- Edelsteine
- Gold
- Pferde
- Oasen
- Öl
- Robben

[41]

Tabelle 2: Civilization Eigenschaften ([70, 35, 64, 41])

Erscheinungsjahr	1991
Entwickler	MicroPose
Publisher	MicroPose
Multiplayer	Nein
Ressourcen	Kohle, Fisch, Wild, Wild (Tundra), Edelsteine, Gold, Pferde, Oasen, Öl, Robben
Siegbedingungen	Domination, Space Race, Time
Spielbare Völker	14
Perspektive	Vogel
Technologien	67



Abbildung 2: Screenshot aus Civilization ([35])

## Rezensionen

IGDB	93% [35]
AllGame	5/5 [57]
Game Informer	8.5/10 [58]
Next Generation	4/5 [59]

## 4.3 Sim City 2000

Sim City 2000 ist ein im Jahr 1993 erschienenes Städtebauspiel für Einzelspieler, welches in das Genre *Simulation* fällt [66]. Das Spiel wurde ursprünglich von *Maxis* für den Mac entwickelt und publiziert, wurde in den kommenden Jahren jedoch für weitere Konsolen herausgegeben, darunter *SNES*, *PlayStation* und *Nintendo 64*. Im Jahr 2005 erschien es schließlich für den PC unter Windows [37]. Der Spieler startet auf einer ausgewählten Karte, welche ein *Square Grid* mit verschiedenen Höhen und Terrain besitzt. Die dabei zentrale Ressource ist Geld in Form von US Dollars. Der Spieler kann mit diesem Geld aus einer breiten Palette an Gebäuden wählen, darunter *Schulen*, *Bibliotheken*, *Krankenhäuser* und *Kraftwerke* [37], welche die Bedürfnisse der

Einwohner befriedigen und für Recht und Ordnung in der Stadt sorgen. Diese Gebäude besitzen einen Radius, was ein weiteres strategisches Element dieses Spiels darstellt, da auf möglichst sinnvolles Platzieren der jeweiligen Gebäude geachtet werden muss. Für Wohn-, Industrie- und Kaufhausgebäude markiert der Spieler Zonen, statt die Gebäude einzeln zu setzen. Die Gebäude dieser Zonen werden dann, je nach Bedarf, Stück für Stück aufgebaut oder wieder verlassen. Der Bedarf jeweiliger Zonen ist in [Abbildung 3](#) anhand des grünen (Wohngebäude), des blauen (Kaufhausgebäude) und des gelben (Industriegebäude) Balkens am linken Auswahlbalken erkennbar. Zeigt ein Balken dabei in die Richtung des +, so signalisiert das *Bedarf*, zeigt der Balken jedoch in die Richtung des -, ist *Überschuss* vorhanden. Zwischen diesen Zonen herrscht eine Relation, so benötigen Kaufhausgebäude die Industriegebäude als Produktionsquelle, und Wohngebäude als Käufer. Die Wohngebäude wiederum benötigen Arbeitsplätze, sowohl in Kaufhausgebäuden als auch in Industriegebäuden. Die Industriegebäude benötigen die Arbeitskräfte der Wohngebäude und Abnehmer für die produzierte Ware in Form von Kaufhausgebäuden. Diese Relation wird wie folgt berechnet (R:C:I beschreibt die Relation zwischen Wohngebäuden zu Kaufhausgebäuden zu Industriegebäuden):

- Unter 10.000 Einwohnern: R:C:I = 4:1:3
- Zwischen 10.000 und 60.000 Einwohnern: R:C:I = 4:2:2
- Über 60.000 Einwohner: R:C:I = 4:3:1 [65]

Die Wohngebäude brauchen *Strom*, *Wasser*, *Bildung*, *Sicherheit* und *Freizeitaktivitäten*, welche allesamt durch platzierbare Gebäude gegeben werden können. Bildung ist eine Kernkomponente des Spiels, denn je nach Bildungsgrad der Stadt steigt oder fällt die Kriminalitätsrate und entscheidet auch darüber, welche Industrien blühen oder bankrottgehen. Der Bildungsgrad wird gemessen am EQ (*education quotient*), so erhöht beispielsweise eine Schule den EQ von den 5 bis 20 Jahre alten Bewohnern. Es gibt insgesamt 156 Gebäude, welche in 8 Kategorien eingeteilt werden können, darunter verschiedene Arten von Wohngebäuden, Kaufhausgebäuden oder Industriegebäuden, wie auch verschiedene Parks, Statuen oder Kraftwerke [39].

Ein weiteres zentrales Element des Spiels ist das *Budget*, welche am Ende jedes Spieljahres adjustiert werden kann. So kann man die *Steuern* erhöhen, die die Einwohner zahlen müssen, wie auch die *Ausgaben* für verschiedene Institutionen anpassen. So kann beispielsweise das Budget der Polizei verringert werden oder das Budget der Feuerwehr erhöht werden, woraus jeweils Effektivitätsboni oder -mali folgen [62].

Das Spiel hat keine Siegbedingungen, die Ziele setzt sich der Spieler selber. Mögliche Ziele sind dabei eine möglichst schöne Stadt, eine Stadt mit möglichst vielen Einwohnern oder man spielt vor sich hin und versucht ein Problem nach dem anderen zu lösen. Auch wenn das Spiel an sich nicht gewonnen werden kann, gibt es dennoch einen Endzustand und eine Art Gewinnsequenz. Der Endzustand ist ein Game Over, bei dem der Spieler durch \$100.000 Dollar Schulden aus „seinem Büro eskortiert wird“, siehe [Abbildung 4](#). Die mögliche Gewinnsequenz, die einem Sieg am ehesten kommt, ist durch baubare Arkologien bedingt. Arkologien sind sich selbst erhaltende Städte mit einer hohen Bevölkerungsdichte innerhalb eines meist hohen Gebäudes. Der Begriff wurde in den 50er Jahren von dem italienisch-amerikanischen Architekten Paolo Soleri geprägt, jedoch wurde bis heute noch keine echte Arkologie gebaut, auch wenn es dazu

Tabelle 3: Sim City 2000 Eigenschaften ([37])

Erscheinungsjahr	1993
Entwickler	Maxis
Publisher	Maxis
Multiplayer	Nein
Ressourcen	Dollars
Siegbedingungen	Keine
Endzustände	Game Over
Perspektive	Isometrisch

bereits einige Experimente, beispielsweise in Arizona, gibt [71]. Nachdem der Spieler 301 sogenannte *launch arcos* gebaut hat und das Jahr 2051 bereits angebrochen wurde, erscheint dem Spieler eine Nachricht, in Englisch „*The Exodus has begun*“, woraufhin alle gebauten Arkologien explodieren und dem Spieler suggeriert wird, die Schiffe wären aufgebrochen, um neue Planeten zu besiedeln [8].

Auch wenn das Spiel ein reines *Singleplayer* Game ist, also lediglich ein menschlicher Spieler gleichzeitig spielen kann, spielt man dennoch gegen max. vier weitere Computerspieler, auch genannt *KIs* (Künstliche Intelligenzen). Die maximal vier weiteren angrenzenden Karten der jeweils zu Anfang ausgewählten Karte werden von *KIs* besiegt und ebenfalls bebaut. Diese Nachbarstädte können mittels gegebener Transportmöglichkeiten, zum Beispiel Flugzeug oder Zug, Gewinn bringen, da die Bewohner der Nachbarstädte in beispielsweise den Kaufhäusern Geld ausgeben [69].

Eine weitere Eigenheit des Spiels sind die sogenannten *Katastrophen* beziehungsweise *Desaster*. Diese können vom Spieler komplett ausgeschaltet werden, falls gewollt. Die meisten Desaster können zufällig und natürlich passieren, darunter Feuer, Aufstände oder Fluten. Alle Desaster können ebenfalls vom Spieler selbst initiiert werden, um die Grenzen der eigenen Stadt auf die Probe zu stellen. Manche Katastrophen sind verkettet, so können Aufstände zu Feuer führen, oder Erdbeben zur Explosion von Kernkraftwerken führen, welche wiederum zu Feuern und Aufständen führen kann [69]. Das Spiel hatte einen Vorgänger *Sim City*, welcher bereits 1989 erschien, und erhielt neun Nachfolger, darunter *Sim City 3000*, *Sim City 4* und den neusten Teil *SimCity: BuildIt*, welcher lediglich für mobile Endgeräte verfügbar ist und 2014 erschien. Der letzte Teil für den Computer erschien 2013 und trägt den Titel *SimCity 2013* [60].

Wichtige Eckdaten sind in Tabelle 3 vorzufinden.

## Ressourcen

- Dollar

## Rezensionen

IGDB	78% [37]
AllGame	4.5/5 (PC) [6]
MacUser	4.5/5 (Mac) [43]
Sega Saturn Magazine	86% (SAT) [42]

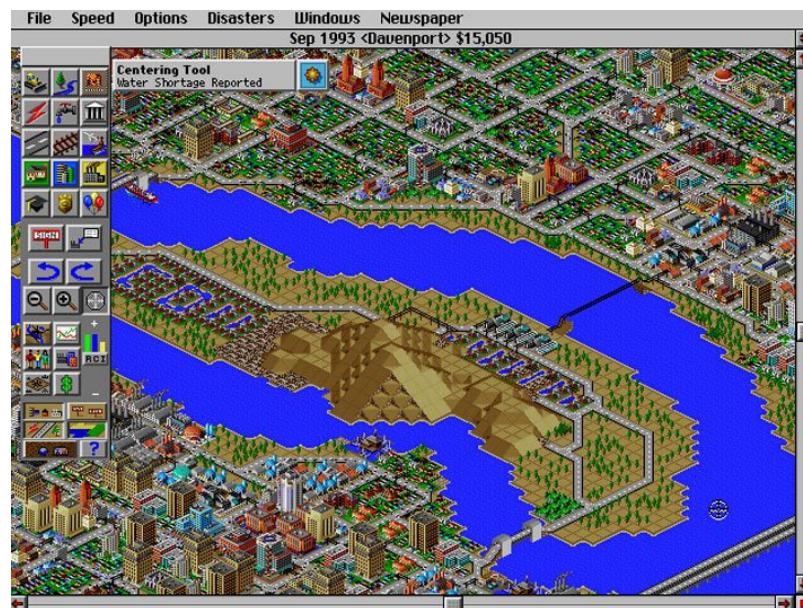


Abbildung 3: Screenshot aus Sim City 2000 ([37])

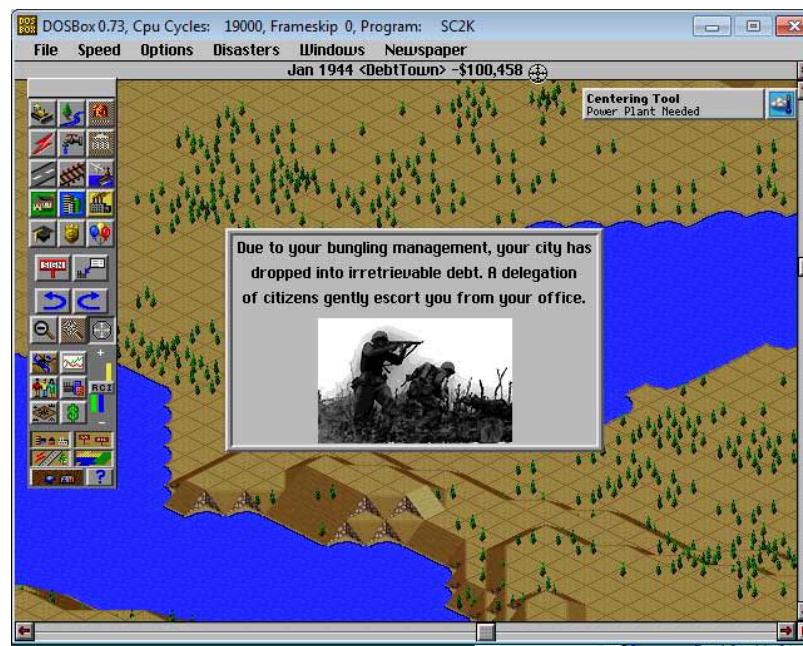


Abbildung 4: Game Over in Sim City 2000 ([61])



Abbildung 5: Generierter Planet, Screenshot aus RimWorld

## 5 Analyse

Im Folgenden wird eine IST-Analyse des Spiels *RimWorld* durchgeführt, um anhand eines Colony Management Games herauszufinden, welche Mechaniken und Eigenschaften dieses Spiel auszeichnen, und welche davon gegebenenfalls übernommen werden könnten für den Prototyp eines neuen Spiels. Dazu wird zu aller erst das Spiel näher untersucht, darunter die Mechaniken, die Kernelemente und das User-Interface. Anschließend werden Leitfaden-Interviews durchgeführt, dessen Ergebnisse und Transkripte verwendet werden, um Hypothesen für den Prototypen aufzustellen. Diese Hypothesen werden letztendlich mit einer Umfrage geprüft und mit der Implementierung des Prototypen schlussendlich umgesetzt.

### 5.1 RimWorld

Das Strategiespiel *RimWorld* ist ein im Jahr 2018 erschienenes Sci-Fi *Colony Management* Game und wurde von *Ludeon Studios* entwickelt und publiziert. Im normalen Szenario startet man mit drei Überlebenden eines Raumschiffabsturzes und versucht seine Kolonie aufzubauen und am Leben zu halten. Das Spiel hat etliche Mechaniken und Tücken, und durch den *AI Storyteller*, welcher eine Vielfalt an Events plant und durchführt, ist eine hohe *replayability* gegeben. Dieser kann auf eine gewünschte Schwierigkeit gestellt werden, von friedfertig bis unfair. Nachdem man den gewünschten Storyteller ausgewählt hat kann man den Planeten generieren (vgl. Abbildung 5) und seinen Startpunkt innerhalb der Spielwelt bestimmen (vgl. Abbildung 6).

#### Spielwelt

Die Welt ist für gewöhnlich eine Pangaea-artige Landmasse auf einem dreidimensionalen, blauen Planeten mit einem Äquator und einem Pol (vgl. Abbildung 5). Der Planet ist aufgeteilt in hexagonale Kacheln, welche jeweils eine Karte symbolisieren. Die Kacheln haben dabei jeweils verschiedene Eigenschaften, die dann auf der gesamten Karte, welche der Spieler aussucht, gelten. Auf Abbildung 6 lassen sich am linken



Abbildung 6: Ausgewählter Startpunkt für die Kolonie, Screenshot aus RimWorld

Rand die meisten Eigenschaften anzeigen. Darunter die Terrain-Eigenschaften, wie viel Berge es gibt und damit auch Erzquellen, welche Steinarten es gibt und die Höhe der Karte. Außerdem ist ein maßgeblicher Faktor die Extreme der Temperaturen, zu welchen Zeiten man Felder anbauen kann und wie viel Regen für gewöhnlich fällt. Der Name der Kacheln, in diesem Fall „Temperate Forest“, gibt außerdem Aufschluss darüber, wie bewaldet die Kachel ist. Je nach Distanz zum Pol oder Äquator ändern sich diese Eigenschaften. Außerdem besitzen manche Kacheln einen Fluss oder bestehen gänzlich aus Wasser oder Eis, oder haben eine Anbindung zu einer Straße (grau) oder einem Feldweg (braun). Wählt man eine Kachel aus und drückt auf „Next“ am unteren Bildschirmrand, gelangt man in die Auswahl der Kolonisten.

## Kolonisten

Das Herz des Spielgeschehens sind die Kolonisten, die, mit manchen Ausnahmen, ihre zugewiesenen Aufgaben erfüllen. Die drei Startkolonisten kann man so lange neu generieren, bis man zufrieden ist. Es gibt dabei mehrere Eigenschaften, die ein Kolonist mit sich bringt. Die darunter fundamentalsten Eigenschaften sind die *Skills*, welche mittig in Abbildung 7 zu sehen sind. Es gibt 12 verschiedene Skills, wovon jeder Kolonist ein bestimmtes Level hat. Diese werden zwischen 0 - 5 generiert, wonach noch die *Hintergründe* (engl. *backstories*) und *Eigenschaften* (engl. *traits*) der jeweiligen Kolonisten addiert werden [16]. Manche Hintergründe addieren oder subtrahieren Level von Skills, manche machen manche Tätigkeiten und damit einhergehende Skills auch unmöglich, darunter der Hintergrund *Romanschriftsteller* (engl. *novelist*), welcher es dem Kolonisten unmöglich macht, einfache Arbeiten zu erledigen, etwa Putzen oder Gegenstände umhertragen [9]. Eigenschaften sind analog zu den Hintergründen zufällig generiert, beziehen sich größtenteils jedoch auf soziale Interaktionen oder bringen bestimmte Boni oder Mali für die *Stimmung* des Kolonisten, darunter Sexualitäten, wie schnell der Kolonist lernen kann, oder exotischere Eigenschaften wie *Kannibale* oder *Psychopath* [67]. Die Flammen neben dem Level der Skills stehen für die *Passion* des Kolonisten für bestimmte Skills. Ist keine Flamme neben dem Level zu sehen, so lernt der Kolonist diese Fähigkeit beim Durchführen davon mit einem Faktor von 35%. Ist eine Flamme



Abbildung 7: Auswahl der Startkolonisten, Screenshot aus RimWorld

vorhanden, ist der Kolonist *interessiert* und hat einen Lernfaktor von 100%. Sind zwei Flammen vorhanden, *brennt* der Kolonist für diese Tätigkeit und lernt mit einem Faktor von 150%. Da viele verschiedene Tätigkeiten zu bestimmten Zeitpunkten ausschlaggebend für das Überleben der Kolonie sein könnten, kann es sinnvoll sein, eine breite Palette verschiedener Passionen und hochrangigen Skills zu haben. Ältere Kolonisten haben für gewöhnlich höhere Skills, sind jedoch davon öfter von *Gesundheitsproblemen* betroffen, etwa Verlust von Seh- und Hörvermögen, oder langsamere Laufgeschwindigkeit. Außerdem haben manche Startkolonisten bereits zuvor gebildete *Beziehungen* zu manch anderen, welche bestimmte Interaktionen vereinfachen oder erschweren.

## Spielgeschehen

Der Spieler wird nach der Anfangssequenz des Absturzes der Kolonisten nicht an die Hand genommen. Ein großer Teil des Spieles besteht darin, Strategien für das Überleben der Kolonie zu finden. Für gewöhnlich beginnt man damit, erste Teile der zukünftigen Basis zu bauen, wie diese aussehen soll und aus welchem Material diese besteht ist, wie so oft in diesem Spiel, dem Spieler selbst überlassen. Jeder Kolonist hat eine *Stimmung* (engl. mood), welche es gilt im Auge zu behalten (vgl. Abbildung 8). Sollte die Stimmung zu lange zu niedrig sein, wird der Kolonist in einen nicht kontrollierbaren Zustand fallen, auch engl. *mental break* genannt, wovon es verschiedene gibt. Darunter *tantrum*, wobei der Kolonist alle möglichen Gebäude und Strukturen in seiner Sichtweite attackiert und möglicherweise zerstört, oder *given up and leaving*, wobei der Kolonist die Spielerkolonie verlässt und aus dem Spiel entfernt wird [45]. Um diese Zustände zu vermeiden muss der Spieler sinnvoll die zu Anfang gegebenen Ressourcen nutzen, und für Nahrung, Unterschlupf, Wärme, Licht und etliche weitere Gegebenheiten sorgen. Je nach gewählten AI Storyteller passieren selten oder öfter Events, welche das Spielgeschehen maßgeblich beeinflussen können. Darunter sind *raids*, wobei fremde Kolonien



Abbildung 8: Stimmungsbalken und -einflüsse, Screenshot aus RimWorld

die Spielerkolonie angreifen, Gebäude zerstören, Reichtümer stehlen oder Kolonisten verschleppen, oder ein zufälliger Ansturm an Biebern, welche sämtliche Bäume auf der Karte Stück für Stück abholzen [24]. Um gegebene Probleme zu Lösen kann der Spieler neue Technologien erforschen, welche letztendlich bis zum Bau eines Raumschiffes führen, was das vorprogrammierte Endziel des Spiels ist. Auf dem Weg zum Bau des Raumschiffes wird der Spieler also konstant vor neue Probleme, Launen der Kolonisten und Ressourcenmanagement gestellt, wodurch eine hohe replayability gegeben ist.

## User Interface

Das UI von RimWorld ist recht gefüllt, denn in jeder Ecke des Bildschirms finden sich direkt mehrere verschiedene Informationen, welche der Spieler erst kennenlernen und verstehen muss. Im Folgenden wird sich stark auf Abbildung 9 berufen, anhand dessen das UI analysiert wird. Das UI ist zum besseren Verständnis in 13 verschiedene Stücke aufgeteilt, welche allesamt eigene Funktionen und einen bestimmten Mehrwert bieten.

Der erste Bereich des UI (1) zeigt die Ressourcen im Besitz des Spielers. Nicht alle Ressourcen werden vom Spieler besessen, sondern erst jene, die im definierten Lager vorhanden sind. Der zweite Bereich (2) ist für kritische Meldungen, die sofortiges Handeln erfordern, in diesem Fall ein medizinischer Notfall, welcher zum Tod des Kolonisten führen kann. In Bereich (3) werden alle Kolonisten, die Teil der eigenen Kolonie sind, angezeigt. Man kann dort sowohl Namen, als auch Aussehen, derzeitige Lebenspunkte und momentane Stimmung ablesen (hellblaue Füllung im Hintergrund des Quadrats). Im vierten Bereich (4) ist gerade ein Event zu sehen, welches zufällig generiert wurde. In diesem Fall wird die Spielerkolonie von einer Händlerkarawane besucht, welche ein eigenes Inventar hat und mit dem Spieler handeln kann, also auch ein *Händler* im Sinne einer ökonomischen Funktion. Bereich (5) zeigt einige Zustände der Spielerkolonie an, darunter derzeitig hungernde Kolonisten, oder dass ein Jäger keine Waffe zum Jagen besitzt. Sehr dringende Zustände mit sofortigem Handlungsbedarf werden, wie in der Grafik zu erkennen, rot hinterlegt dargestellt. In Bereich (6) werden ausschließlich vom AI Storyteller generierte Events mitgeteilt, also Hitzewellen, Angriffe oder, wie in diesem Fall, Besuche von Händlerkarawanen. Im siebten Bereich (7) erkennt man meteorologische Informationen zu derzeitigen Wettergegebenheiten, der Temperatur, den derzeitigen Monat und die Jahreszeit. Bereich (8) beinhaltet die Elemente zur Steue-



Abbildung 9: User Interface, Screenshot aus RimWorld

rung der Spielgeschwindigkeit. Bereich (9) zeigt verschiedene Optionen zum Anzeigen bestimmter Informationen. Man kann sich beispielsweise noch die Fruchtbarkeitsgrade des Bodens einblenden lassen, oder die Schönheit der Umgebung aus der Sicht eines Kolonisten. Der zehnte Bereich (10) stellt die Menüleiste dar, unter welcher man die meisten Aktionen im Spiel durchführen kann. Unter dem Reiter *Architect* finden sich zum Beispiel Möbel und Strukturen (vgl. Abbildung 10), welche man platzieren kann. Der am besten passende Kolonist wird dann dieser Tätigkeit zugeordnet. Bereich (11) zeigt einen im Bett liegenden Kolonisten, welcher zuvor verwundet wurde und sich nun regeneriert. Bereich (12) zeigt Informationen über das Element, auf welche die Maus gerade liegt. Dort können interessante Informationen bereitgestellt werden, zum Beispiel auf dem Boden befindlicher Dreck oder Blut, welches aufgeräumt werden sollte. Der letzte, nicht markierte Bereich ist der restliche Bereich des Bildschirms, wo sämtliches Spielgeschehen simuliert wird. Man erkennt den Anfang einer Kolonie, bestehend aus einigen Holzbaracken, einer Küche, einem Raum mit Tisch, um dort zu essen, ein kleines Lager und einen Produktionsraum, wo die Kolonisten die meiste ihrer Arbeit verrichten.

## Prioritäten

Ein Großteil des Spielgeschehens läuft über die Prioritäten der Kolonisten. Man kann für jegliche mögliche Tätigkeit eine manuelle Priorität setzen (vgl. Abbildung 11), wodurch die Kolonisten erst bestimmte Tätigkeiten verrichten, bevor manch andere angefangen werden. Die Prioritäten reichen dabei von 4 - niedrige Priorität, bis 1 - hohe Priorität. Durch Linksklick auf ein Prioritätenfeld erhöht man die Priorität, durch Rechtsklick erniedrigt man sie. Man kann ebenfalls die Priorität auf *nichts* setzen, wodurch die Tätigkeit unter keinen Umständen ausgeführt wird. Es kann daher sinnvoll sein, die Prioritäten höher zu setzen bei überlebenswichtigen Dingen wie Feuerlöschen oder me-



Abbildung 10: Baumenü, Screenshot aus RimWorld



Abbildung 11: Prioritäten, Screenshot aus RimWorld

dizinischer Hilfe, falls darin geübt.

## Steuerung der Kolonisten

Anders als traditionelle Resource Management Games (Age of Empires, Civilization), steuert man seine Einheiten nicht *direkt*, in dem man diese auswählt und dann auf das gewünschte Feld klickt, sondern *indirekt*. Man wählt zuerst die auszuführende Tätigkeit aus der Liste der möglichen Befehle aus (vgl. Abbildung 12), markiert den Bereich für den Befehl und der nächstbeste Kolonist, der diese Aufgabe erfüllen kann und die höchste Priorität dafür hat, wird zugewiesen sich darum zu kümmern. Das macht die zuvor genannten Prioritäten unerlässlich, um einen reibungslosen Spielverlauf zu haben. Der Vorteil dieser Mechanik ist ganz klar, dass sich das Management der Kolonisten eher anfühlt wie das einer Ameisenkolonie, statt dem herkömmlichen direkten Kommandieren der einzelnen Einheiten. Der klare Nachteil davon ist, dass man nicht immer die gewünschten Dinge erledigen lassen kann, die man gerade hätte. In manchen Situationen sind andere Tätigkeiten prioritär, wodurch man die Prioritäten stetig im Auge behalten muss und öfter mal anpasst, sodass auch wirklich das erledigt wird, was gerade notwendig ist.

## Ressourcen

Das Spiel hat eine große Vielfalt an verschiedener Ressourcen. Darunter fallen Grundbedarfsgüter, wie eine einfache Mahlzeit, Heu oder Reis. Es gibt von einigen Rohstoffen allerdings verschiedenste Ausführungen, so hat beispielsweise Leder 20 verschiedene



Abbildung 12: Befehle, Screenshot aus RimWorld



Abbildung 13: Die Vier Phasen des Interviews

Variationen (Schweinehaut, Hundeleder, Fuchsfell, ...). Es gibt weiterhin sehr exotische Güter, zum Beispiel verschiedene bionische Körperteile, mit welchen man fehlende ersetzen kann, oder Nanobots, die fehlende Organe wiederherstellen können [56]. Es gibt eine Vielfalt an verschiedener, zubereitbarer Mahlzeiten, welche verschiedene Boni auf die Stimmung geben und jeweils verschiedene Rohstoffe benötigen.

## Haltbarkeiten

Viele greifbare Ressourcen, welche auch in der realen Welt aufgrund von Korrosion zerstört würden oder als Lebensmittel mit der Zeit ablaufen, besitzen eine Haltbarkeit. Diese sinkt stetig mit vergehender Zeit von 100 auf 0. Ist diese auf 0 gesunken, verschwindet die Ressource aus dem Spiel, damit stellt diese Mechanik der Haltbarkeit ein Drain dar. Manche Ressourcen benötigen ein Dach oder einen Raum, damit die Haltbarkeit nicht weiter sinkt, andere wiederum und darunter primär Lebensmittel, benötigen eine sehr niedrige Temperatur, damit sie konserviert werden können.

## 5.2 Interviews

Für die Ermittlung gut funktionierender Mechaniken und Konzepte wird auf die Methode des kontextuellen Interviews zurückgegriffen [32]. Dazu wird ein Leitfadeninterview angefertigt, um qualitative Daten von den Teilnehmern zu extrahieren und daraus Hypothesen anzufertigen [10]. Nachdem die Hypothesen erarbeitet wurden, werden diese mittels einer Umfrage in einem breiten Umfang überprüft. Es werden drei Personen ausgewählt, wobei die Erfahrung aller Personen in Bezug auf das Spiel und das Genre generell variiert. Diese Personen, im Folgenden auch *Probanden* genannt, werden in Einzelgesprächen dazu gebeten, dass Spiel eine Stunde lang zu spielen. Einzelgespräche beziehungsweise -interviews sind Gruppengesprächen vorzuziehen, da die einzelnen Probanden einen entspannteren Zeitplan haben, sich gänzlich auf das Spiel und die Fragen fokussieren können und keine Gefahr laufen, abgelenkt zu werden oder mit anderen Probanden in Gespräche zu gelangen [38]. Das Interview gliedert sich in vier Phasen (vgl. Abbildung 13).

### 5.2.1 Versuchsaufbau

Im Zentrum des Interviews steht, neben den Einleitungs- und Abschlussfragen, das Videospiel *RimWorld*, welches für eine Stunde lang von den Probanden gespielt wird. Das Spiel läuft zu dem Zeitpunkt der Interviews auf der Version 1.3.3387, ist auf die englische Sprache eingestellt und wird über die Online-Vertriebsplattform für Computerspiele *Steam* ausgeführt. Das Interview ist, wie bereits erläutert, in vier Phasen gegliedert und wird in der Wohnung des Interviewers durchgeführt. Wichtig ist, dass alle Probanden leicht andere Qualifikationen besitzen, sodass ein breites Bild über den Zustand des untersuchten Spiels erstellt werden kann. Es ist fundamental zu erkennen, wie ein Spiel von Kennern, wie auch von Neulingen empfunden wird. Jedes Interview wird dabei mit einem Mikrofon aufgenommen und anschließend transkribiert. Die Transkripte sind wörtlich und nicht lautsprachlich angefertigt. Dies bietet den einfachen Vorteil, dass die später aufgestellten Hypothesen mit Textbelegen der Aussagen der jeweiligen Teilnehmer aufgestellt beziehungsweise belegt werden können. Allerdings werden die Namen der Teilnehmer nicht genannt und stattdessen durch ein Pseudonym ersetzt [38, S.97]. **DIE ERSTELLTEN TRANSKRIPTE MÜSSEN NOCH ANGEHANGEN WERDEN UND REFERENZIERT!**

**Einleitung** Um die erhobenen Daten richtig kategorisieren zu können, werden als erstes grundlegende Daten zu der Person an sich erfasst:

- F1: Wie alt bist du?
- F2: Nach welchem Geschlecht identifizierst du dich?

**Aufwärmen** Im zweiten Schritt des Interviews werden etwas spezifischere Fragen zu der Erfahrung und dem Verhalten bezüglich Videospielen gestellt:

- F3: Hast du bereits Erfahrung in Videospielen?
- F4: Hast du bereits Erfahrung in Resource Management Games?
- F5: Hast du RimWorld bereits zuvor gespielt?

**Beobachtung** Die Phase der Beobachtung beschreibt die Spielphase. Der Proband wird dazu aufgefordert, das Videospiel *RimWorld* zu spielen. Es ist von fundamentaler Wichtigkeit für diese Phase, dass die beobachtende Person keinerlei Versuch unternimmt, sich in das Spielgeschehen oder das Verhalten der beobachteten Person einmischt. Es wird auch unterlassen, begangene Fehler aufzuklären oder in irgend einer anderen Art und Weise zu helfen, falls nicht anders möglich. Außerdem ist es wichtig zu erwähnen, dass nicht alle Fragen zwangsläufig gestellt werden. Diese Fragen dienen lediglich als Katalog möglicher Fragen, die situativ gestellt werden können.

- F8: Welche Emotion wird gerade verspürt?
- F9: Ist das Spiel für den Probanden immersiv?
- F10: Hat der Proband Verlustängste bezüglich der Kolonisten?

*Tabelle 4: Jeweilige Antworten der Probanden auf die Einleitungsfragen*

	Proband A	Proband B	Proband C
Alter	19	23	24
Geschlecht	♀	♂	♂
Erfahrung Videospiele	✗	✓	✓
Erfahrung Resource Management	✗	✓	✓
Erfahrung RimWorld	✗	✗	✓

- F11: Wieso machst du \_\_\_\_?
- F12: Weißt du, was du nun tun kannst?
- F13: Was ist dein nächstes Ziel?
- F14: Verwendet der Proband die Prioritätenliste?
- F15: Kommt der Proband gut mit der indirekten Anweisungsmechanik zurecht?

**Abschluss** In der Abschlussphase wird der Proband gebeten, eine eigene Meinung abzugeben. Außerdem können Fehler aufgeklärt werden und inhaltliche Fragen beantwortet werden, ohne dass sie nun das Verhalten im Spiel verzerren.

- F16: Was hast du als gut empfunden?
- F17: Was hast du als schlecht empfunden?
- F18: Wie findest du das indirekte Anweisen?
- F19: Was sagst du zu der Ressourcenvielfalt?
- F20: Was hältst du von dem Prioritätensystem?
- F21: Wie findest du die vielen zufällig generierten Umstände?

### 5.2.2 Probanden

Für die Interviews werden exakt drei Personen ausgewählt, welche sich allesamt in ihrer Erfahrung in Videospielen und auch in ihrer Erfahrung in RimWorld unterscheiden. Es ist damit zu erhoffen, dass ein möglichst breites Spektrum an Informationen extrahiert werden kann. Da Proband C sich zu dem Zeitpunkt des Interviews nicht vor Ort befand, wurde mit dieser Person stattdessen über Discord ein Meeting gehalten, in welchem der Bildschirm des Probanden dem Interviewer geteilt wurde. Der Prozess lief ungewöhnlich unkomplizierter ab, da beide Personen bereits vertraut mit der Software waren.

### 5.2.3 Ablauf

Jegliches Interview beginnt mit den Einleitungsfragen, wobei die Antworten der Probanden zusammengefasst in [Tabelle 4](#) zu finden sind. Die Altersspanne der Teilnehmer liegt zwischen 19 und 24 Jahren, mit einem Altersdurchschnitt von 22 Jahren. Zwei der drei Probanden sind männlich, ein Teilnehmer ist weiblich. Die beiden männlichen Teilnehmer weisen beide bereits Erfahrung in Videospielen und Resource Management Games auf, wobei lediglich einer der beiden männlichen Teilnehmer bereits RimWorld zuvor gespielt hat. Alle Teilnehmer wurden darüber in Kenntnis gesetzt, dass ihre Stimmen aufgezeichnet werden, und waren damit einverstanden. Die im Folgenden aufgestellten Hypothesen werden mit [HX] gekennzeichnet, wobei das X eine fortlaufende Nummerierung der jeweiligen Hypothesen darstellt. Die Referenzen auf die Transkripte werden mit [X, Z.Y] dargestellt, wobei X in diesem Fall das Pseudonym des jeweiligen Probanden ist (A, B, C), und Y die jeweilige Zeile in dem gegebenen Transkript.

Zu Beginn fällt auf, dass der Startbildschirm des Spiels sehr viele Buttons enthält, und die Option „New Colony“ für Neulinge nicht unbedingt eindeutig assoziiert wird mit dem Beginn eines neuen Spiels [A, Z.27-30]. Es wäre daher sinnvoll, die Anzahl der Knöpfe auf ein Minimum zu reduzieren [H1]. Die Steuerung und die einzelnen Symbole auf der generierten Weltkarte sind nicht eindeutig assoziierbar [A, Z.39-48][B, Z.38-54], eine gewisse, gut sichtbare Erklärung oberhalb der Weltkugel wäre eine hilfreiche Ergänzung [H2]. Für spezielle, stark herausfordernde Gegner oder Umgebungseigenschaften empfiehlt sich die Möglichkeit, diese auch ausstellen zu können [H3]. Das gibt dem Spieler mehr Kontrolle über mögliche Frustration [C, Z.37-40]. Außerdem fällt auf, dass, gerade für Neulinge, die englische Sprache in Videospielen etwas speziell sein kann. Diese Spiele enthalten Begriffe, welche im schulischen oder alltäglichen Kontext eher weniger auftreten und daher erst gelernt werden müssen. Es ist daher sinnvoll, möglichst viele Sprachen aufgrund der höheren Inklusion und den dadurch niedrigeren Frust zu implementieren [H4][A, Z.49]. Die Auswahl der Startkachel hängt scheinbar sehr stark mit der Erfahrung des Probanden zusammen, Proband A wählt den Startpunkt zufällig [A, Z.41-48], um möglichst schnell das eigentliche Spiel zu beginnen, während Proband B den Startpunkt abhängig von den benachbarten Fraktionen macht, da diese als positiv assoziiert werden [B, Z.56-59]. Proband C ist der einzige Proband, welcher sich die Eigenschaften der einzelnen Kachel anschaut und basierend auf konkreteren Informationen die Entscheidung fällt, darunter die Anbauzeit und die Terrain-Beschaffenheit [C, Z.43-44]. Dieser Lernprozess und die damit verbundene, gewonnene Erfahrung, ist nicht schlecht. Es lässt Raum für Verbesserung und Lerneffekte. Dem Spieler sollte daher nicht mitgeteilt werden, welche Kacheln sinnvoller wären, als andere [H5]. In der Auswahl der Kolonisten fällt auf, dass selbst Spieler, welche bereits Erfahrung in anderen Spielen des Genres haben, leicht überfordert mit der Anzahl und Diversität der Eigenschaften der Kolonisten haben können [B, Z.74-77]. Für den Umfang von RimWorld ergibt diese Entscheidung durchaus Sinn, jedoch ist wichtig zu erkennen, dass hier gegebenenfalls unnötige Komplexität hinzugefügt wird, welche Spieler frustrieren kann. Daher sollten die Traits auf eine kleinere Menge reduziert werden und intuitiv benannt und gestaltet sein, damit sowohl neue, als auch erfahrene Spieler dieses System als gut empfinden [H6]. Mit der Zeit entstehen eigene Taktiken, welche Traits gut und welche als schlecht empfunden werden [B, Z.67-71][C, Z.46-53], was eine durchaus positive Entwicklung ist. Es sollte daher nicht klar erkennbar sein, welche Traits die

besten, und welche die schlechtesten sind, der Spieler sollte dies über Zeit und durch mehrfaches Ausprobieren lernen [H7]. Die in Abbildung 9 in Bereich (5) dargestellten Hinweise auf den Zustand der Kolonie erweisen sich als sehr hilfreich, da jeder der drei Probanden diese als Informationsquelle verwendet [B, Z.82]. Es könnte daher hilfreich sein, kleine, nicht zu viel verratende, Informationen über negative Eigenschaften an den Spieler preiszugeben [H8]. Somit wäre der Spieler nicht komplett auf sich alleine gestellt, aber dennoch nicht an die Hand genommen, sodass ein Lernprozess stattfinden kann. Das Lagersystem in RimWorld scheint für den Großteil der Probanden nicht ersichtlich zu sein, Proband A findet keine Möglichkeit zu lagern, während Proband B erst nach einer sprachlichen Klärung des Begriffes „Stockpile“ die Lagermechanik beginnt zu begreifen [B, Z.89-92]. Statt einem versteckten Lagersystem sollte das Lager deutlich ersichtlicher dem Spieler präsentiert werden, in Form einer baubaren Struktur, analog zu einer Kiste oder einem Ablageplatz [H9]. Eine weitere nicht ersichtliche Information ist die momentane Uhrzeit (vgl. Abbildung 9, Bereich (7)), welche zentraler dargestellt werden sollte [H10][B, Z.95-98]. Die Mechanik der Temperatur scheint für Proband B interessant zu sein [B, Z.98], wodurch eine analoge Mechanik eventuell für etwas mehr Komplexität sorgen könnte, ohne dabei mehr Erklärungsbedarf zu kreieren [H11]. Alle drei Probanden verstehen, dass Nahrung in irgendeiner Weise zubereitet werden sollte [A, Z.106-111][B, Z.107-111][C, Z.72-73], wodurch sinnvolle Converter-Mechaniken durchaus intuitiv sein können. Daher sollten Converter an einigen Stellen verwendet werden, um die Komplexität etwas zu erhöhen, ohne das Verständnis des Spielers zu fördern [H12]. Es zeigt sich im Verlauf, dass die Steuerung der Kolonisten sehr unintuitiv sein kann, falls nicht anders bereits bekannt [A, Z.72]. Die Prioritätenliste, mit welcher die indirekten Anweisungen der Kolonisten optimiert werden können, wird von dem Großteil der Probanden nicht verwendet und teilweise auch falsch verstanden [A, Z.85-90][B, Z.127-128]. Das System der Prioritäten scheint positiven Anklang zu finden [A, Z.135][B, Z.155][C, Z.130-132], wodurch ein analoges System unbedingt implementiert werden sollte [H13], allerdings sollten die Nummern gegebenenfalls durch Symbole oder leichter zu verstehende Darstellungsmöglichkeiten ersetzt werden, welche keine Einbuße in der Komplexität darstellen, aber das Verständnis für neuere Spieler durchaus fördern können [H14]. Auch das indirekte Anweisen der Kolonisten wird größtenteils als positiv empfunden, und für den Anwendungsfall von RimWorld dem direkten Anweisen, wie zu finden in beispielsweise *Age of Empires*, vorgezogen [A, Z.127][B, Z.155][C, Z.122-124]. Während dieses System bei neueren Spielern eher ein Gefühl von Machtlosigkeit auslöst [A, Z.129], schlägt diese Empfindung bei steigender Erfahrung in ein Gefühl von mehr Kontrolle und Vorausplanung [B, Z.160-161], und letztendlich in das positive Gefühl des Beobachtens eines eigenen „Ameisenstamms“[C, Z.123-124] um. Daher wird dieses System auf lange Sicht einen positiveren Effekt bringen, als die alternative des direkten Anweisens, und wird daher vorzugsweise implementiert [H15]. Allerdings sollte es Hinweise auf die Steuerung geben, um anfänglichen Frust zu vermeiden, und neuere Spieler direkt abzuholen [H16]. Es zeigt sich außerdem, dass die vielen, zufällig generierten Umstände, beginnend bei der Weltkugel, der Startkachel und den Kolonisten, tendenziell eher als positiv aufgefasst werden, auch wenn Anfangs etwas unübersichtlich [A, Z.139-134][B, Z.173-175][C, 133-136]. Laut Probanden erhöhe diese Zufälligkeit in vielerlei Hinsicht die replayability des Spiels. Es empfiehlt sich daher, eine ähnliche Struktur zu übernehmen, und eine zufällig generierte Karte, wie auch zufällig generierte Einheiten mit jeweils eigenen, distinkten Eigenschaften bereitzustellen [H17]. Für alle Proban-

den scheint das User-Interface in einigen Punkten undurchsichtig und unübersichtlich. Proband A empfindet die Menge an Text als sehr überwältigend und problematisch [A, Z.57], wodurch es sinnvoll sein könnte, einige Informationen eher als Bilder beziehungsweise Icons darzustellen, statt als Text [H18]. Proband B hat eher Probleme mit der Anordnung der dargestellten Informationen, darunter die bereits erwähnte Uhrzeit. Solche zentralen Elemente sollten daher sichtbarer dargestellt werden ([H10]). Eine frustrierende Eigenschaft des Spiels ist die teilweise zu stark anziehende Schwierigkeit, welche vermieden werden sollte bei der Implementierung des Prototypen. Die Schwierigkeit sollte schwer genug sein, damit das Gefühl einer Herausforderung entsteht, aber nicht zu schwer, sodass der Spieler sich teilweise auch zurücklehnen kann, und der Kolonie beim Arbeiten zuschauen kann, ohne stetig gezwungen zu sein, Input zu geben [H19][C, Z.114-119]. Letztendlich ist ein Tutorial für das Erlernen des Spiels unerlässlich aufgrund der vielen gegebenen Informationen [A, Z.121-124][B, Z.143-145]. Es gibt dabei verschiedene Möglichkeiten, ein Tutorial zu implementieren, aber dem Spieler sollte jederzeit eine Art Hilfestellung zur Verfügung stehen, damit kein Gefühl von Stagnation oder Frustration entsteht aufgrund fehlender Informationen [H20]. Zuletzt scheint die Kontrolle über die Geschwindigkeit des Spiels als essenzieller Bestandteil, sowohl Proband B [B, Z.78-81] als auch Proband C greifen darauf häufig zurück, lediglich Proband A sieht darin keinen Nutzen [A, Z.77-79], wobei dieser Nutzen mit der Erfahrung in Videospielen generell erkennbar wird. Demnach sollte eine solche Zeitkontrolle ebenfalls implementiert werden, um die Kontrolle des Nutzers zu erhöhen [H21].

Die hier aufgestellten Hypothesen sind zur besseren Übersicht und für spätere Referenzen in [Tabelle 5](#) aufgelistet.

### 5.3 Umfrage

Die zuvor erarbeiteten Hypothesen sollten ursprünglich im Folgenden über die empirische Methode der Online-Umfrage überprüft werden [23]. Die Online-Umfrage sollte auf *Discord*-Servern durchgeführt werden. Discord ist ein soziales Medium, welches dazu dient Videos oder Videos auszutauschen, oder sich über Sprach- oder Textkanäle miteinander zu vernetzen und zu kommunizieren [50]. Discord hat 140 Millionen aktive Nutzer pro Monat, Tendenz stark steigend (Stand 2021) [17]. Es gibt dabei mehrere *Discord-Server*, welchen man mit einer Einladung beitreten kann, auf welchem sich mehrere Personen befinden, mit welchen man kommunizieren kann. Es gibt für RimWorld einen dedizierten *Discord-Server*, auf welchem sich (Stand 15.08.2022) 37.634 Mitglieder befinden. Bei der Erarbeitung ist jedoch ein entscheidender Punkt aufgefallen, welcher die Umfrage überflüssig macht: *Discord-Server* dieser Art werden im Normalfall tendenziell eher von Leuten betreten, welche bereits viel Kontakt mit dem Spiel haben oder hatten, und Informationen austauschen wollen. Da die Hypothesen zu einem entscheidenden Teil aus Annahmen bestehen, welche das Spielgefühl für Anfänger optimieren sollen, wäre der Raum, in welchem die Umfrage stattfinden würde, unangemessen. Ein erfahrener Spieler ist bereits an das User Interface gewöhnt, weiß, wonach und wohin er schauen muss und hat eigene Strategien für das Spiel entwickelt, wie das Interview mit Proband C gezeigt hat, was die Hypothesen großteils unbrauchbar macht.

Tabelle 5: Übersicht aller aufgestellten Hypothesen der durchgeführten Interviews

H1	Die Auswahlmöglichkeiten im Startmenü auf das Nötigste reduzieren
H2	Steuerung und Optionen oberhalb der Weltkugel anzeigen
H3	Herausfordernde Umgebungseigenschaften an- und ausschalten können
H4	Mehrere Sprachen unterstützen für eine höhere Inklusion und weniger Frust
H5	Keine Hilfe bei der Auswahl der Startkachel
H6	Traits intuitiv und Anzahl gering halten
H7	Traits nicht eindeutig im Nutzen vergleichbar machen
H8	Kleine Hinweise auf Probleme geben
H9	Lagerplätze als baubare Strukturen erkenntlicher machen
H10	Zentrale Informationen (Uhrzeit) kenntlicher darstellen
H11	Temperatur als mögliche Erweiterung der Komplexität
H12	Intuitive Converter als Erweiterung der Komplexität
H13	Prioritätensystem schafft mehr Kontrolle
H14	Prioritäten in Form von Grafiken erhöhen das Verständnis
H15	Das indirekte Anwisen der Kolonisten verstärkt das Spielgefühl positiv
H16	Die Steuerung sollte ersichtlicher sein
H17	Zufällige Startbedingungen erhöhen die replayability
H18	Grafiken erhöhen die Übersicht im User-Interface
H19	Die Schwierigkeit muss sinnvoll zwischen herausfordernd und entspannt balanciert werden
H20	Es bedarf einem Tutorial und/oder Hilfestellungen auf Abruf
H21	Eine Zeitkontrolle ist eine sinnvolle Ergänzung

## 6 Prototyp

Für den praktischen Anteil dieser Arbeit wird auf Grundlage der gewonnenen Erkenntnisse aus vorheriger Sektionen ein Prototyp angefertigt. Es wird also auf die Hypothesen zurückgegriffen, welche durch die Interviews und Umfragen erarbeitet und überprüft wurden. Außerdem wird anhand eines geeigneten Beispiels der Natur eine Thematik Stück für Stück untersucht und in Mechaniken transformiert. Konkret werden in Folgendem die Vorgänge innerhalb einer Bienenkolonie genauer beleuchtet, und erörtert, welche Elemente sich für ein Resource Management Game eignen. Der Prototyp wird versioniert mittels GitHub und ist jederzeit in Form eines [Repositories](#) [53] aufrufbar.

### 6.1 Thematik

Für den Prototypen bedarf es einer Thematik beziehungsweise einer Idee für das Gesamtkonzept. Dafür wurde die Idee einer *Bienenkolonie* für interessant befunden. Bienen und deren Kolonien sind aus vielerlei Hinsicht geeignet als Thematik für den Prototyp, darunter der Fakt, dass Bienen bekannt dafür sind, Honig zu produzieren, wobei Honig eine konkrete Ressource darstellt. Im Kontext eines Resource Management Games und mit Rücksicht auf das zuvor untersuchte Spiel *RimWorld* eignet sich jedoch am besten eine ganz spezifische Art der Bienen, die *Honigbiene*, genauer gesagt die *Westliche Honigbiene* (lat. *apis mellifera*) [25]. Grund dafür ist die Verhaltensweise von Honigbienen, da diese, anders als nahe Verwandte, stärker dazu tendieren, in Kolonien zu leben.

Hummeln beispielsweise formen zwar auch Kolonien, aber deutlich kleinere und kürzer lebende. Die meisten Arten der Wildbienen jedoch sind tendenziell Einzelgänger. Eine weibliche Wildbiene legt ein Nest und versorgt ihren Nachwuchs mit Pollen und Nektar [77]. Somit bietet die Gattung der Honigbiene einige Vorgänge, welche sich in ein Colony Management Game umsetzen lassen. Dazu werden im Folgenden interessante Vorgänge und Eigenschaften der Gattung der Honigbiene genauer untersucht. Es ist zu erwähnen, dass der folgende Abschnitt nicht dazu dient, das gesamte Spektrum einer Bienenkolonie zu untersuchen, sondern lediglich Aspekte, die förderlich für die Entwicklung des Resource Management Games wären.

### 6.1.1 Nahrung

Eine Honigbiene ernährt sich primär von *Nektar*, welchen sie von Blumen sammelt. Dieser Nektar ist zuckerhaltiger Saft, welcher von den Pflanzen ausgeschieden wird, um damit Insekten verschiedener Arten anzulocken. Dabei sind Sonnenblumen, Obstblüten, Löwenzahn und Raps besonders gute Nektarquellen [22]. Eine weitere Möglichkeit um an energiereichen Zucker zu gelangen ist der sogenannte *Honigtau*, welcher von einigen Insekten, darunter Blatt- und Schildläusen, ausgeschieden wird. Aus diesen beiden zuckerhaltigen Säften können die Bienen den *Honig* produzieren, welcher ebenfalls als Nahrungsquelle dient, aber vor allem über die Wintermonate besonders wichtig ist, da Honig sehr lange haltbar ist [47]. Der durch Honigtau produzierte Honig wird von Imkern als *Waldhonig* gekennzeichnet [22]. Saugt eine Biene den Nektar einer Blüte auf, gelangt dieser in den *Honigmagen*. Dort kommt er in Kontakt mit verschiedenen Enzymen, welche den Nektar in *Glukose* (Traubenzucker) und *Fruktose* (Fruchtzucker) [48] aufspalten. Diesen biochemisch veränderten Nektar gibt die *Sammelbiene* im Bienenstock an die sich dort befindlichen *Arbeitsbienen* durch Auswürgen weiter, welche den Nektar erneut aufnehmen und auswürgen. Mit jedem dieser Prozesse wird der Nektar viskoser, wodurch allmählich Honig entsteht [76].

Neben dem Nektar, dem Honigtau und dem Honig, welche primär als Quellen für Kohlenhydrate dienen, benötigen Bienen außerdem noch Quellen für Fette und Proteine. Dafür werden *Pollen* verwendet, wobei Pollen die männlichen Geschlechtszellen einer Samenpflanze sind. Da die Samenpflanze mehr Pollen produziert, als für die Fortpflanzung nötig, können diese ohne Probleme von den Sammelbienen aufgenommen werden [47]. Die Pollen werden in *Pollenpaketen* aufbewahrt (vgl. Abbildung 14), wobei Teile der aufgesammelten Pollen nicht in diesen Pollenpaketen landen, sondern an den Hinterbeinen haften bleiben. Besucht diese Sammelbiene nun eine weitere Blüte, werden diese Pollen abgestreift und der Fortpflanzungsmechanismus der Samenpflanze profitiert. Besonders pollenreiche Quellen sind dabei Obstbäume, Mohn, Mais, Klee und Raps [47].

Die letzte wichtige Ressource, welche Bienen für ihr Überleben benötigen, ist Wasser. Dieses kann von Flüssen oder Seen bezogen werden, wobei diese Quellen nicht mehr als 500 Meter weit vom Stock entfernt sein sollten. Erhalten Bienen nicht genug Wasser, können diese unter Verstopfungen leiden, was für das Überleben gefährlich sein kann [47, 22].

Es lassen sich somit wichtige Ressourcen identifizieren: *Nektar*, *Honigtau*, *Honig*, *Pollen* und *Wasser*. Außerdem ist die Aufteilung zwischen *Sammel-* und *Arbeitsbiene* eine mögliche Mechanik. Die Umwandlung von Nektar und Honigtau zu Honig könnte



Abbildung 14: Pollenpakete einer Honigbiene ([25])

ebenfalls eine Mechanik darstellen, wie auch die naturgegebenen Quellen für Nektar, Honigtau und Pollen.

### 6.1.2 Kasten

Eine fundamentale Kategorisierung innerhalb einer Kolonie sind die *Kasten*. Es gibt drei verschiedene Kasten beziehungsweise Arten von Bienen innerhalb solch einer Kolonie. Die erste Art ist die *Bienenkönigin*. Diese Art der Biene ist genau einmal vertreten und *immer* ein Weibchen, zudem innerhalb einer Kolonie das einzige vollentwickelte. Die Hauptaufgabe der Königin ist das Brüten neuer Bienen und das Steuern des Schwärms mittels verschiedener Pheromone [20]. Die zweite Art sind die sogenannten *Drohnen*. Diese Art ist *ausschließlich männlich*. Diese Kaste ist lediglich zur Befruchtung der Königin da und ist nur von Frühling bis Sommer des Jahres in der Kolonie vertreten. Anschließend werden alle Drohnen gewaltsam aus der Kolonie entfernt, was als *Drohnenschlacht* betitelt wird [28]. Die dritte und letzte Kaste der Kolonie sind die *Arbeitsbienen*, welche, analog zur Königin, *ausschließlich weiblich* sind, jedoch den deutlich größten Teil einer Kolonie ausmachen. Trotzdem legen diese Bienen in der Regel keine Eier, sind im Gegensatz dazu aber deutlich fürsorglicher gegenüber der Brut im Vergleich zur Königin. Die Arbeitsbienen sind prinzipiell für sämtliches weiteres Geschehen verantwortlich, darunter die Entfernung von Leichen oder die Nahrungsbeschaffung [26, S.2-3]. In Abbildung 15 werden die verschiedenen Kasten morphologisch unterscheidbar dargestellt mit einer Markierung verschiedener Merkmale.

### 6.1.3 Fortpflanzung

Ein wichtiger Teil des Fortbestehens einer Kolonie ist die *Vermehrung* beziehungsweise *Fortpflanzung*. Wie bereits zuvor erwähnt gilt, dass sowohl Königinnen, als auch Arbeitsbienen jederzeit *weiblich*, und Drohnen stets *männlich* sind. Eine Königin ist in der Lage unbefruchtete Eier zu legen, oder diese Eier mittels Paarung mit Drohnen zu befruchten. Auch Arbeitsbienen sind in der Lage unbefruchtete Eier zu legen, was in einer Kolonie tendenziell selten passiert, da die Geschlechtsorgane der Arbeitsbienen zurück-

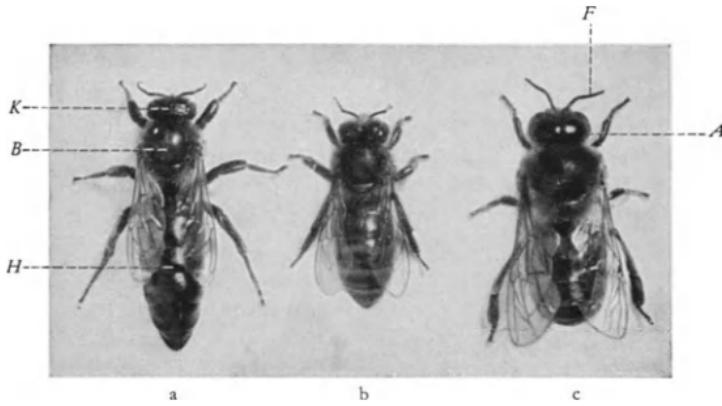


Abbildung 15: (a) Königin, (b) Arbeitsbiene, (c) Drohne. (K) Kopf, (B) Brust, (H) Hinterleib, (A) Auge, (F) Fühler ([26, S.2])

entwickelt und verkrümmt sind [28]. Wird allmählich eine neue Königin gebraucht, da die derzeitige Königin an das Ende ihres Lebens gelangt, sondert diese bestimmte Pheromone aus, wodurch dem Schwarm mitgeteilt wird, neue Königinnen heranzuziehen. Eine würde theoretisch ausreichen, aber um kein Risiko einzugehen werden mindestens *sechs* Königinnen, oder mehr, versucht heranzuziehen. Nachdem eine geeignete Königin geschlüpft ist, werden die restlichen gewaltsam aus dem Schwarm entfernt [26, S.33]. Spätestens zwei Wochen später fliegt die neu geschlüpfte Königin aus und versprüht Pheromone, auch *Königinnensubstanz* genannt, welche Drohnen eigener und fremder Völker anlocken, um sich mit der Königin zu paaren. Es wird mit maximal 12 Drohnen der Paarungsakt vollzogen, wobei die Königin bis zu *zehn Millionen* Spermien aufnimmt, welche für die restliche Lebenszeit reichen [20].

Einige Tage nachdem die Eier der zukünftigen Königinnen gelegt wurden, in der Regel neun Tage, beginnt der Akt des *Schwärmens*. Dabei verlässt ein Teil der Kolonie, tausende von Bienen, zusammen mit der noch regierenden Königin den Schwarm, um sich auf die Suche nach einem neuen Zuhause zu machen und ein neues Bienenvolk zu gründen. Dieser Akt passiert in der Regel nur ein Mal pro Jahr, gegen Mai. *Spurbienen* agieren als Späher und teilen Informationen über mögliche neue Heimatplätze. Laut Forschungen bedarf es 15 Spurbienen, welche dieselben Informationen über einen passenden Ort teilen, damit die Entscheidung gefällt wird [74]. Dank der Pheromone der Königin bleibt der Schwarm während des Schwärmens stets zusammen [20].

Eine Königin besitzt einen *diploiden* Chromosomensatz, dementsprechend  $2n = 32$  Chromosomen. Die *unbefruchteten* Eier werden von keinem zweiten Chromosomensatz ergänzt, wodurch die sich daraus entwickelnden Drohnen vorerst einen *haploiden* Chromosomensatz besitzen, dementsprechend  $n = 16$  Chromosomen in Summe. Diese werden jedoch nachträglich diploid, lediglich die Keimzellen bleiben haploid. Dadurch gilt, dass die anschließend diploiden Körperzellen der Drohne *homozygot* sind, da sie mittels *Autopolyploidisierung* aus einer haploiden Eizelle entstanden sind. Beide Gene eines Merkmals stimmen also exakt überein [33]. Aus unbefruchteten Eiern schlüpfen ausschließlich Drohnen. Gegensätzlich dazu schlüpfen aus den *befruchteten* Eiern Arbeitsbienen *oder* Königinnen. Diese Eizellen sind durch die Beteiligung zweier Partien, der Königin und einer Drohne, sowohl *heterozygot*, als auch diploid. Der Faktor, ob aus



Abbildung 16: Verschiedene Stadien einer Honigbiene, ausgewachsen (links), Puppe (mitte), Larve (rechts) ([25])

einem befruchteten Ei eine Arbeitsbiene oder eine Königin schlüpfen wird, ist modifikativ, also durch äußere Einflüsse, bestimmt. Der Unterschied liegt in der zugegebenen Nahrung der Maden in den ersten drei Lebenstagen. Während die zukünftigen Arbeitsbienen mit Pollen und Honig ernährt werden, erhalten die zukünftigen Königinnen ausschließlich *Gelée Royal* (engl. *Royal Jelly*) in ihren sogenannten *Weiselzellen*, welche ausschließlich einer zukünftigen Königin vorenthalten sind [74]. Somit gilt, dass Arbeitsbienen und Königinnen genetisch identisch sind, jedoch durch die zugegebene Nahrung modifikativ zu einer gewissen Kaste herangezogen werden können [28]. Es gilt also dass, auch wenn auf den ersten Blick paradox erscheinend, eine Drohne nie einen Vater, aber immer einen Großvater hat.

#### 6.1.4 Metamorphose

Der Vorgang der *Metamorphose* beschreibt das Anpassen der physischen Form oder ein plötzliches Ändern der äußeren Erscheinung [19]. Bienen durchlaufen vier verschiedene Zustände der Metamorphose, beginnend als *Ei* (engl. *egg*). Diese Eier werden von der Königin in eine leere Wabe gelegt, haben eine Größe von 1mm bis 1.5mm und ähneln einem Reiskorn. Nach circa *drei Tagen* brechen die Eier und eine *Larve* (engl. *larva*) kommt hervor. Diese Larve ist weiß und C-förmig (vgl. Abbildung 16). Die Dauer der kommenden Metamorphose hängt von der jeweiligen Kaste der zukünftigen Biene ab, Arbeitsbienen benötigen 6 Tage, Drohnen 6.5 Tage und Königinnen 5.5 Tage. Ist die Larve reif für die Metamorphose, richtet sie sich auf, sodass die Arbeitsbienen, welche Zuständig für die Brut sind, die Zelle mit *Bienenwachs* bedecken. Die Larve verhärtet und wird zu einer *Puppe* (engl. *pupa*). Analog zum vorherigen Stadium ist die Dauer bis zur kommenden Metamorphose abhängig von der Kaste der zukünftigen Biene. Eine Arbeitsbiene benötigt 12 Tage, eine Drohne 14.5 Tage und eine Königin 8 Tage. Ist die Zeit reif und die Metamorphose abgeschlossen beißt sich die Biene durch das Bienenwachs der Zelle und gesellt sich zu ihren Artgenossen [25].

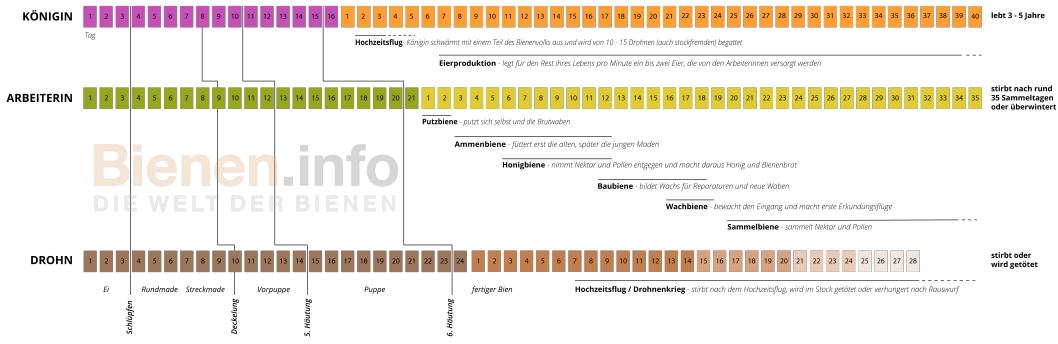


Abbildung 17: Überblick über den Lebensverlauf einer Biene jeder Kaste ([75])

### 6.1.5 Aufgabenverteilung

Die Aufgaben der Königin und der Drohnen wurde zuvor bereits ausgiebig erläutert. Eine Arbeitsbiene hat drei exakt eingeteilte Phasen ihres Lebens, in welchen verschiedene Tätigkeiten verrichtet werden. Die zu erledigenden Aufgaben sind also direkt gekoppelt an das derzeitige Alter einer Arbeitsbiene. Der erste Lebensabschnitt ist zwischen dem 1. und 10. Lebenstag der Biene. In diesem Stadium wird die Biene als *Hausbiene* bezeichnet, da sie sich ausschließlich innerhalb des Bienenstocks aufhalten. Dort reinigen sie die Zellen, kümmern sich um die Brut als sogenannte *Brutamme* und sind ansonsten meist untätig. Der zweite Lebensabschnitt findet zwischen dem 10. und 20. Lebenstag der Biene statt. In diesem Abschnitt wird die Biene als *Baubiene* eingestuft. Die Aufgaben sind nun primär das Entgegennehmen des von den Sammlern gebrachten Nektar oder der Pollen, das Bauen neuer Waben und das Herstellen von Wachs. Außerdem sind manche dieser Bienen im *Wächterdienst* und beschützen den Stock vor Eindringlingen wie Wespen, Pferden oder Menschen. Im dritten und letzten Abschnitt des Lebens sind die Bienen *Sammlerbienen*, welche Nahrung für den restlichen Stock von der Außenwelt suchen. Allerdings gibt es nur Ausflüge, wenn das Wetter und die Temperatur passend sind. Bei Regen, Schnee oder generell im Winter sitzen diese Bienen im Stock und warten auf Veränderung der Außenbedingungen. In der Regel widmen diese Bienen sich nicht den anderen Aufgaben und warten stattdessen [26, S.42-44]. Abbildung 17 zeigt einen Überblick der jeweiligen Aufgaben abhängig von dem Alter einer Biene jeglicher Kaste.

### 6.1.6 Lebenserwartungen

Die Lebenserwartung einer jeden Biene hängt mit ihrer jeweiligen Kaste zusammen. Die kürzeste Lebensdauer weisen die Drohnen auf, welche zwischen zwei bis vier Wochen leben, da diese anschließend durch die Drophenschlacht gewaltsam entfernt oder getötet werden. Am längsten hingegen leben Königinnen, mit drei bis fünf Jahren Lebensdauer. Bei Arbeitsbienen ist es entscheidend, ob diese während des Sommers oder gegen Anbruch des Winters geschlüpft sind. Durch die Wintereinnistung steigt die Lebenserwartung der Arbeitsbienen stark an, von gerade mal zwei bis vier Wochen auf sechs bis sieben Monate [75]. Eine Übersicht dieser Lebenserwartungen ist in Abbildung 18 vorzufinden.

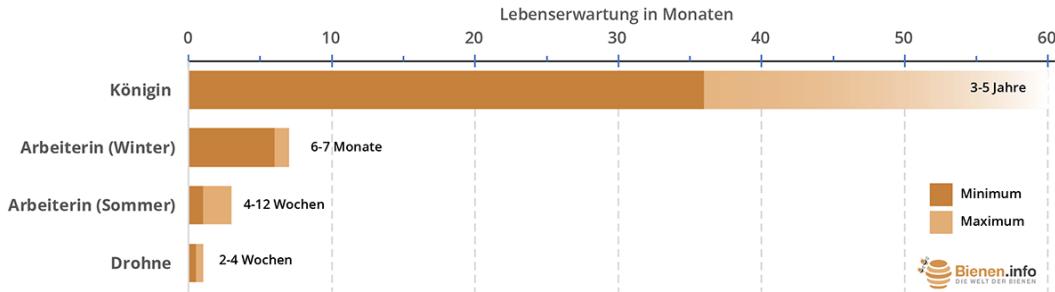


Abbildung 18: Die minimale und maximale Lebenserwartung einer Biene jeder Kaste ([75])

### 6.1.7 Winter

Die Vorbereitung auf den kommenden Winter beginnt bereits im Spätsommer, wobei neue Brut erzeugt wird für den Zweck der Überwinterung, sogenannte *Winterbienen*. Ab Oktober werden die *Sommerbienen* aus der Kolonie entfernt, sodass die Kolonie lediglich aus Winterbienen besteht. Dadurch, dass keine Energie für Brutpflege oder Sammelflüge aufgewendet werden muss, ist die Lebenserwartung der Winterbienen deutlich höher als die der Sommerbienen, wie bereits zuvor erläutert. Außerdem legen sich die Winterbienen ein Fettpolster zu, durch hohen Konsum von Pollen. Statt eines Winterschlafes formiert sich der Stock zu einer *Wintertraube*, bei welcher die Kolonie die Königin umschließt und durch Muskelvibration warm hält. Mit Anfang des Frühlings wird mit gleicher Technik der Stock auf 35°C erhitzt, wodurch der Nahrungsverbrauch stark ansteigt aber das erfolgreiche Brüten der neuen Generation sichert. Sobald die ersten Blumen wieder spritzen setzen wieder die Sammelflüge ein [72].

## 6.2 Spiel-Engine

Ein Videospiel wird heutzutage nicht mehr von Grund auf neu programmiert. Die Grundlage schafft in den meisten Fällen eine sogenannte *Game Engine* (Spiele-Engine). Diese Engine bietet wichtige Grundlagen, welche es dem Entwickler oder dem Entwicklungsteam deutlich vereinfachen loszulegen. Darunter fallen Grafik-, Physik- und Audiosysteme, welche je nach Engine variieren [30]. Aufgrund der sehr hohen Popularität [46] und der bereits bestehenden Erfahrung wird bei dieser Arbeit auf *Unity*[68] zurückgegriffen. Mögliche und erwogene Alternativen waren *Godot Engine* und *Unreal Engine*, welche sich rein theoretisch allesamt anbieten würden. Auch wenn jede dieser Game Engines für den Umfang und die Komplexität des Prototypen geeignet wäre, überwiegt die vorhandene Erfahrung in Unity für die letztendliche Entscheidung.

## 6.3 Spielwelt

Die grundlegende Idee der Spielwelt ist, dass diese in zwei Instanzen geteilt wird. Die erste Instanz stellt die Oberwelt dar, wo Pflanzen und Bäume wachsen, wobei die zweite Instanz den Innenraum des Bienenstocks darstellt. In dem Bienenstock können Strukturen gebaut werden, welche in der Außenwelt nicht verfügbar sind. In der Außenwelt hingegen finden sich die Ressourcen, welche für das Fortleben essenziell sind. Für die ge-

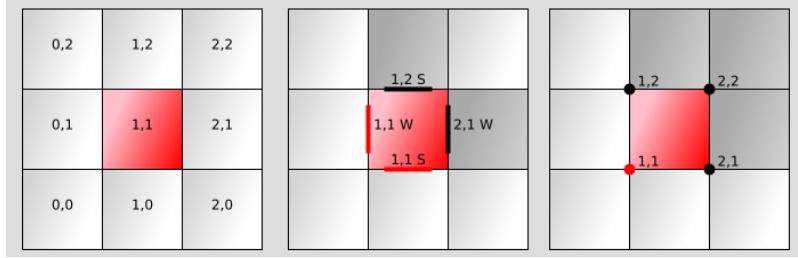


Abbildung 19: Aufbau eines Square Grids ([7])

samte Welt, in welcher das Spiel gespielt wird, stehen mehrere Optionen der Erstellung zur Verfügung.

**Gridless** Die erste Möglichkeit einer Spielwelt ist eine Karte ohne dargestelltes oder vorhandenes Grid. Solch ein System verwendet beispielsweise *Age of Mythology*, wobei die Karte nicht in ein Grid, sondern viele tausende Koordinaten eingeteilt ist. Diese Form der Positionierung ist daher gängig für *Real Time Strategy* Spiele.

**Square Grid** Eine weitere Möglichkeit zur Positionierung von Gebäuden oder Einheiten innerhalb einer Spielwelt ist ein *Square Grid* (vgl. Abbildung 19), also die Einteilung der Karte in eine gewisse Anzahl von Quadranten, welche als eigene Felder agieren und worauf Gebäude oder Einheiten zugreifen können. Dieses System wird unter anderem in *Starcraft II* oder *Age of Empires* verwendet, wodurch auch diese Form der Positionierung gängig für *Real Time Strategy* ist. Allerdings findet man diese Eigenschaft auch in *Turn Based Strategy*, darunter die älteren Teile der *Civilization*-Reihe. Dieses System wurde im Laufe der Jahre jedoch von einem Square Grid auf ein Hex Grid umgestellt.

**Hex Grid** Die letzte Möglichkeit ist ein *Hex Grid*, welches vor allem Anklang findet im Genre der *Turn Based Strategy* Games, darunter *Endless Legend*, *Civilization VI* und *Humankind*. Der klare Vorteil von einem Grid bestehend aus Hexagonen ist die *Distanzberechnung* und die Anzahl *direkter Nachbarn*. Im Vergleich zu einem Quadrat besitzt ein Hexagon in einem Grid 6 statt 4 direkter Nachbarn, wobei direkt bedeutet, dass die Kanten aneinander angrenzen. Um einen diagonalen Weg in einem Square Grid einzuschlagen, muss ein weiterer Weg aufgewendet werden, da nach dem *Satz des Pythagoras* gilt:

$$a^2 + b^2 = c^2 \quad (1)$$

Der diagonale Weg innerhalb eines Quadrates mit Länge 1 und Breite 1 wäre folglich

$$\sqrt{1^2 + 1^2} = \sqrt{2} \quad (2)$$

Wobei offensichtlich gilt,

$$\sqrt{2} > 1 \quad (3)$$

Möchte man also Bewegungen von Einheiten in 6 statt 4 Richtungen ermöglichen, empfiehlt sich ein Hex Grid statt einem Square Grid. Die Distanz zwischen allen gegenüberliegenden Kanten eines Hexagons ist stets gleichlang (vgl. Abbildung 20). Unter der Berücksichtigung, dass die Thematik der Bienen auch mit *Bienenwaben* und deren hexagonalen Form assoziiert wird, wird der Prototyp ein Hex Grid verwenden.

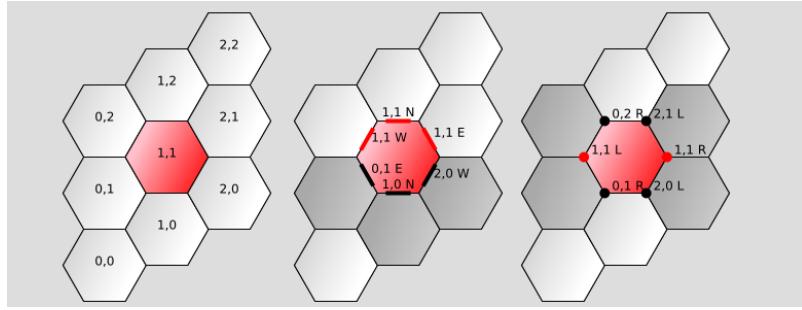


Abbildung 20: Aufbau eines Hex Grids ([7])

### 6.3.1 Generierung

Das Terrain beziehungsweise Grid wird über einen vielschichtigen Algorithmus generiert, welcher die gesamte Fläche in Columns und die Columns wiederum in Chunks unterteilt. Der Algorithmus wurde mittels eines im Internet gefundenen Tutorials [13] erarbeitet und anschließend angepasst. Mittels der Methode *GenerateMap* (vgl. [Quellcode-Ausschnitt 2](#)) kann von außerhalb dann der Algorithmus verwendet werden. Es gibt etliche Variationen zur Anpassung, darunter ein Seed, welcher die Generierung beeinflusst, Meeresspiegel, Erosion, Feuchtigkeit und Temperatur, es ist damit viel Diversität zwischen verschiedener Karten gegeben. In dieser Arbeit wird nicht weiter auf dieses Tutorial eingegangen, ist jedoch für weitere Informationen verlinkt. Der Bienenstock ist in dem Prototypen lediglich eine kleinere Version der generierten Außenwelt und noch nicht visuell als Bienenstock erkennbar. Dies wird in zukünftigen Iterationen angepasst und gelb und orange eingefärbt. Außerdem soll ein Ausgang für die Bienen in dem Bienenstock generiert werden, durch welchen die Einheiten sich von Bienenstock nach Außenwelt und umgekehrt bewegen können.

```

1  public void GenerateMap (int x, int z, bool wrapping) {
2      Random.State originalRandomState = Random.state;
3      if (!useFixedSeed) {
4          seed = Random.Range(0, int.MaxValue);
5          seed ^= (int)System.DateTime.Now.Ticks;
6          seed ^= (int)Time.unscaledTime;
7          seed &= int.MaxValue;
8      }
9      Random.initState(seed);
10
11     cellCount = x * z;
12     overworldGrid.CreateMap(x, z, wrapping);
13     if (searchFrontier == null) {
14         searchFrontier = new HexCellPriorityQueue();
15     }
16     for (int i = 0; i < cellCount; i++) {
17         overworldGrid.GetCell(i).WaterLevel = waterLevel;
18     }
19     CreateRegions();
20     CreateLand();
21     ErodeLand();
22     CreateClimate();
23     CreateRivers();
24     SetTerrainType();
25     for (int i = 0; i < cellCount; i++) {
26         overworldGrid.GetCell(i).SearchPhase = 0;
27         overworldGrid.GetCell(i).MapType = HexMapType.Overworld;
28     }
29
30     Random.state = originalRandomState;
31 }
```

*Quellcode-Ausschnitt 2: World Generation*

## 6.4 Spielbeginn

Das Spiel beginnt auf der nun zufällig generierten, hexagonalen Spielwelt. Es wird mit exakt zwei *Arbeitsbienen* und einer *Königin* gestartet, welche sich im mittleren Alter befinden. Diese Bienen starten auf drei zufälligen Kacheln nebeneinander auf einer zufällig generierten Karte.

## 6.5 Gameplay Loop

Der primäre *Gameplay Loop* besteht darin, die Bienenkolonie am Leben zu halten. Nach Spielbeginn wird der Spieler mit den gegebenen Arbeitsbienen und der Königin auf sich alleine gestellt. Der Spieler sollte damit anfangen, etwas Nektar und Pollen

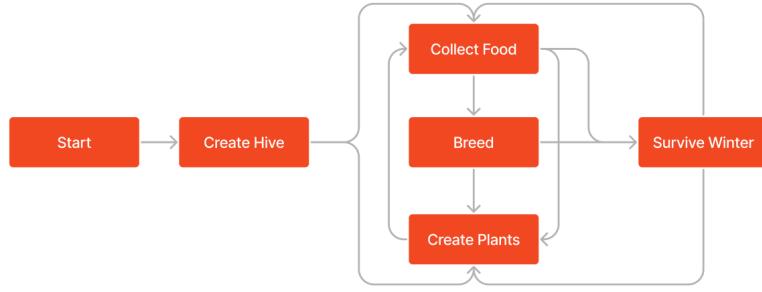


Abbildung 21: Grobe Skizze des Gameplay Loops

zu sammeln, um diese dann zu Bienenwachs zu verarbeiten und einen Stock zu bauen. Innerhalb dieses Stocks sollte die Königin beginnen, Drohnen heranzuziehen, mit welchen man wiederum neue Arbeiterinnen und/oder eine neue Königin heranziehen kann. Das Ziel besteht darin, das Überleben der Kolonie möglichst lange zu sichern. Die Herausforderung besteht darin, dass die Winter schwer sind und mit der Zeit auch immer schwerer werden. Es gilt, die Balance zu finden zwischen Nahrungsbeschaffung, Erweiterung des Stocks und Erzeugung neuer Brut. Zeugt der Spieler zu viel neue Bienen, kippt das Gleichgewicht und die Kolonie stirbt an einem Mangel an Nahrung. Das grobe Konzept des Gameplay Loops ist in Abbildung 21 veranschaulicht.

## 6.6 Spielende

Analog zu *SimCity 2000* gibt es kein direkt vordefiniertes Ende. Der Endzustand des Game Over wird jedoch erreicht, wenn keine Königin mehr verfügbar ist, um die Kolonie zu leiten. Das Spiel ist also theoretisch unendlich lange spielbar, sollte jedoch von der Schwierigkeit so gesteigert werden, dass ein natürliches Ende nach einiger Zeit eintritt. Die Endzustände des Spieles können später noch angepasst werden, eine Idee wäre es, dem Spieler nach Sterben der Königin noch etwas mehr Zeit zu geben um darauf reagieren zu können, sodass eine gerade herangezogene Königin noch die Chance hat zu schlüpfen. Es könnte auch die maximale Jahreszahl begrenzt werden und dem Spieler eine Siegbedingung bereitstellen, in Form eines Überlebens für eine gewisse Zeit.

## 6.7 Mechaniken

Es wurden in vorherigen Sektionen einige Mechaniken und Eigenheiten von älteren und neu erschienen Management Games untersucht und Hypothesen dazu angeführt. Diese Hypothesen und Mechaniken, welche sich als besonders interessant und nutzerfreundlich erweisen, werden nun unter Berücksichtigung der erörterten Vorgänge innerhalb einer Bienenkolonie konkretisiert und ausgefeilt. Außerdem werden die definierten Eigenschaften eines Resource Management Games berücksichtigt und versucht, die untersuchten Gegebenheiten (Ressourcen, Ökonomie, Informationsgehalt und Verwaltungsaspekte) möglichst sinnvoll mit den Vorgängen einer Bienenkolonie zu kombinieren, um daraus ein Spiel beziehungsweise Prototypen zu gestalten, welches sehr stark an die echten Vorgänge einer solchen angelehnt ist. Im folgende referenzierte Hypothesen aus Tabelle 5 werden mit [HX] abgekürzt, wobei X für die jeweilige Nummer der Hypothese steht.

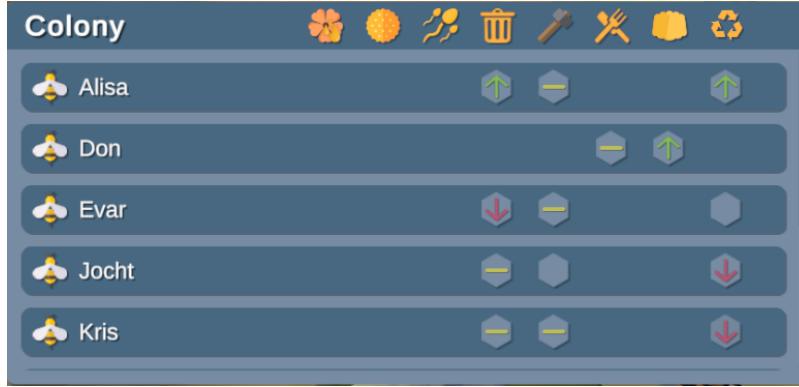


Abbildung 22: Fertig implementierte Prioritätenliste

### 6.7.1 Anweisungsstil

Wie sich in den Interviews und der Analyse von RimWorld gezeigt hat, lassen sich die Art, auf welche Anweisungen an die jeweiligen Einheiten erteilt werden, in *direktes* und *indirektes* Anweisen beziehungsweise Zuweisen. Während in den bereits vorgestellten Titeln meistens auf eine direkte Anweisung der Einheiten zurückgegriffen wird, stellt RimWorld eine Neuerung dar, welche laut [H15] für das Genre des Colony Managements als durchaus positiv von den Probanden aufgefasst wird. Daher wird dieser Anweisungsstil, mitsamt der Prioritätenliste [H13], welche mittels Grafiken verständlicher gemacht wird [H14]. Diese Grafiken finden sich in Abbildung 22 wieder. Es gibt fünf verschiedene Prioritätszustände pro Tätigkeitsbereich pro Biene. Falls die Biene aufgrund ihrer Kaste oder ihres Alters eine bestimmte Tätigkeit nicht machen kann, ist der Button nicht vorhanden. Ansonsten gilt, dass ein grüner Pfeil nach oben bedeutet *Hohe Priorität*, ein gelber Strich bedeutet *Mittlere Priorität*, ein roter Pfeil nach unten bedeutet *Niedrige Priorität*. Falls keine Grafik in dem Button angezeigt wird, steht dies für *Untätig*, womit der Spieler auch entscheiden kann, dass manche Bienen bestimmte Tätigkeiten nicht erledigen, selbst wenn möglich. Wird eine Aktion auf ein Feld angewiesen, wird eine *JobOrder* erstellt, welche alle wichtigen Informationen enthält (vgl. Quellcode-Ausschnitt 3). Diese JobOrder wird in eine Liste hinzugefügt, welche jedes Frame iteriert wird, und geschaut, ob noch offene Anfragen vorhanden sind. Für jede offene Anfrage wird anhand einer Liste vieler Kriterien eine Biene aus der Liste aller vorhandenen adulten Bienen gesucht (vgl. Quellcode-Ausschnitt 4). Zuerst werden alle Bienen gesucht, welche keine Anfrage zugewiesen haben, theoretisch die Anfrage ausführen können und ausgewachsen sind. In einer zweiten Iteration werden die nun ausgesuchten Bienen auf Prioritäten geprüft, die Bienen mit der jeweils höchsten Priorität werden in einer Liste gespeichert. Zuletzt werden spezifische Informationen gesucht, wie etwa ein passendes Inventar für die Anfrage, beispielsweise genug Pollen für die Anfrage Pollinate. Letztendlich sollte von dieser dritten Liste die Biene mit dem kürzesten Pfad ausgewählt werden, wobei dieser letzte Schritt aus zeitlichen Gründen noch nicht im Prototyp vorhanden, aber algorithmisch durchaus möglich ist.

```

1  public JobOrder AddJobOrder(Bee bee, BeeAction action, HexCell hexCell) {
2      JobOrder jobOrder = new JobOrder();
3      jobOrder.Action = action;
4      jobOrder.AssignedBee = bee;
5      jobOrder.Finished = false;
6      jobOrder.MapType = hexCell.MapType;
7      jobOrder.Cell = hexCell;
8
9      jobQueue.Add(jobOrder);
10     hexCell.AssignJob(jobOrder);
11     hexCell.ShowJobHighlight();
12     return jobOrder;
13 }

```

*Quellcode-Ausschnitt 3: Erstellung einer neuen Anweisung*

```

1  private void AssignJobs() {
2      for (int i = 0; i < jobQueue.Count; i++) {
3          JobOrder jobOrder = jobQueue[i];
4          Bee bee = jobOrder.AssignedBee == null ?
5              FindBee(jobOrder) : jobOrder.AssignedBee;
6          if (bee) {
7              jobOrder.AssignedBee = bee;
8              bee.AssignJob(jobOrder);
9              this.activeJobs.Add(jobOrder);
10             this.jobQueue.Remove(jobOrder);
11             bee.Travel(jobOrder.Cell);
12         }
13     }
14 }

```

*Quellcode-Ausschnitt 4: Zuweisung einer Biene von einer offenen Anweisung*

### 6.7.2 Ressourcen

Die im Spiel vorhandenen Ressourcen sind in [Tabelle 6](#) aufgelistet. Die Gründe für die Auswahl der Ressourcen sind *Sektion 6.1.1 Nahrung* zu finden. *Honigtau* wird nicht als Ressource implementiert, da diese von lebenden Tieren extrahiert wird, und aufgrund von weiteren benötigten Modellen und Animationen keine weiteren Tiere neben den eigentlichen Bienen implementiert werden. Um das Spiel spannender und etwas komplexer zu gestalten werden verschiedene *Sources*, *Converter* und *Drains* (vgl. *Sektion 1.1.2*) verwendet. Alle involvierten Ressourcen, wie auch baubare Waben, sind in [Abbildung 23](#) skizziert. Die Zahlen sind dabei rein experimentell und müssen im Verlauf der Tests gegebenenfalls angepasst werden.

Tabelle 6: Verfügbare konkrete Ressourcen

Nahrung	Nektar, Honig, Pollen, Royal Jelly, Wasser
Baumaterial	Bienenwachs
Einheiten	Königinnen, Drohnen, Arbeitsbienen

**Nektar** Nektar wird ausschließlich von Blumen extrahiert, welche eine *Source* darstellen. Jedoch ist an jede Blume eine Bedingung geknüpft. Blumen haben ein eigenes Inventar für Nektar und Pollen, welches bei Extraktion durch eine Biene geleert wird und neu regenerieren muss. Eine Biene sollte also frequentiv die Blumen wechseln, um möglichst viele Ressourcen zu sammeln. Nektar hat eine gewisse Lebensdauer und wird nach einiger Zeit schlecht, wodurch eine gewisse Anzahl von Nektar aus dem Spiel entfernt wird. Diese Lebensdauer agiert dadurch als *Drain*.

**Pollen** Die zweite Ressource sind die Pollen, welche ebenfalls von Blumen extrahiert werden. Pollen können unter anderem verwendet werden, um neue Blumen zu pflanzen und spielen daher gerade im Frühling eine große Rolle, aber auch für das Herstellen für Bienenwachs und Gelée Royal. Pollen besitzen eine etwas längere Lebensdauer und können daher über den Winter gelagert werden. Dadurch, dass Pollen mehrere Anwendungsfälle besitzen, muss der Spieler abwägen, für welchen er diese Ressource am ehesten verwendet beziehungsweise wie er die vorhandenen Ressourcen aufteilt.

**Honig** Honig ist eine manuell hergestellte Ressource, welche an einem *Evaporator*, also einer leicht umgebauten Wabe, von einer Biene hergestellt werden kann. Dieser Evaporator stellt, neben den anderen baubaren Waben, einen *Converter* dar. Ein wichtiger Aspekt des Honigs ist der Fakt, dass man mehr als ein Nektar pro Honig verwenden muss, damit der Spieler abwägen muss, ob es die Herstellung wert ist. Dadurch entsteht mehr Komplexität und der Spieler besitzt Entscheidungsfreiheit. So könnten manche Spieler versuchen, lediglich genug Honig für den Winter herzustellen, und andere wiederum jeglichen Nektar direkt in Honig umwandeln. Es entsteht somit eine mögliche *Strategie*.

**Bienenwachs** Entgegen der anderen Ressourcen ist Bienenwachs keine Nahrung. Diese Ressource wird einzig und allein für den Bau neuer Strukturen verwendet. Diese Strukturen werden im Folgenden näher erläutert. Bienenwachs muss an einem *Mixer* hergestellt werden, welcher sowohl Honig als auch Pollen verwendet, um neues Bienenwachs herzustellen. Aus diesem Bienenwachs können dann *Brutwaben*, *Lagerwaben* und *Refiner* hergestellt werden, wie auch ein neuer Stock, sollte man den gegebenen verlagern wollen.

**Gelée Royal** Eine weitere fundamentale Ressource ist das Gelée Royal oder auch *Royal Jelly*, welches an einem *Refiner* durch Vermischen von Pollen und Honig erzeugt werden kann. Diese Ressource ist wichtig für das Heranziehen einer neuen Königin und wird in die Brutwabe einer *Arbeitsbienenlarve* gegeben, damit diese sich zu einer Königin entwickelt.

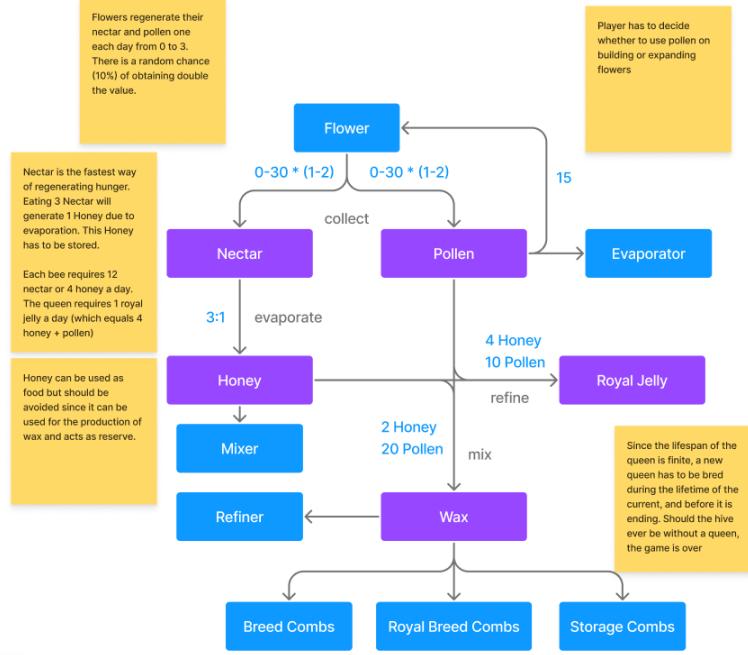


Abbildung 23: Ressourcenverlauf und mögliche Umwandlungen

**Wasser** Die einzige nicht vom Spieler sammelbare oder lagerbare Ressource ist Wasser. Dieses müssen die Bienen durch Nahrung aufnehmen (Nektar besitzt mehr Wasser als Honig), oder über das Trinken an einem Fluss. Da im Winter keine Ausflüge gestattet sind, ist es wichtig, dass Bienen vor dem Winter genug Wasser zu sich nehmen, um nicht zu verdursten.

**Königin, Drohne und Arbeitsbiene** Diese greifbaren Ressourcen sind das Fundament des Spiels, wobei neue Bienen, analog zu *Sektion 6.1.3 Fortpflanzung*, Kastenspezifisch herangezogen werden können, mehr dazu im folgenden Abschnitt.

### 6.7.3 Jahreszeiten

Es gibt vier verschiedene Jahreszeiten, *Frühling*, *Sommer*, *Herbst* und *Winter*. Alle 30 Tage wechselt die Jahreszeit zur jeweils nächsten. Jede Jahreszeit ist dabei anders und bietet andere, mögliche Events. Jegliche Angaben von Chancen sind dabei rein experimentell und müssen im späteren Verlauf getestet werden. Im Prototypen werden aufgrund der niedrigen Zeitspanne keine Events implementiert sein, jedoch bereits konzipiert für die spätere Implementation.

**Frühling** ist die Zeit, in welcher neue Blumen spritzen. Es gibt ein besonderes Event namens *Pollenflug*, wobei besonders viele Blumen spritzen, was es dem Spieler durchaus erleichtern kann, neue Nahrung zu beschaffen. Das Event hat eine Chance von  $\frac{1}{30}$  pro Tag zu passieren, und kann pro Jahr maximal ein Mal geschehen.

**Sommer** ist die Zeit der Hitze und des *Schwärms*. Deshalb gibt es zwei Events welche passieren können, das erste ist die *Dürre*, wobei einige Blumen sterben und manche

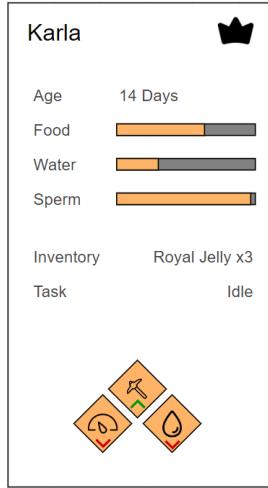


Abbildung 24: Skizze des Informationsfensters einer ausgewählten Biene

Flüsse austrocknen. Die Chance dafür ist pro Tag  $\frac{1}{60}$ . Das zweite ist das Schwärmen, welches jedes Jahr erneut passiert, insofern eine Königin vorhanden ist. Dabei wird angegeben, dass die Königin den Stock verlassen wird. Deshalb ist der Spieler dazu gezwungen, eine neue Königin heranzuziehen, da ansonsten keine Königin mehr vorhanden sein wird. Die Königin wird dabei einen kleinen Anteil der Kolonie mitnehmen (mindestens eine *Arbeitsbiene* und 20% der Arbeitsbienen). Diese Zahl ist variabel und wird gegebenenfalls angepasst. Der Spieler wird also etwas zurückgesetzt und gezwungen, zu handeln, was die Herausforderung durchaus erschwert.

**Herbst** bringt zwei negative Events mit sich. Es kann jeden Tag mit einer  $\frac{1}{60}$  Chance passieren, dass jegliche Blumen weniger Ertrag geben in entweder Pollen oder Nektar. Auch hier wird der Spieler dazu gezwungen, sich anzupassen und sinnvoll darauf zu reagieren. Zudem ist die Chance auf Regen durchaus etwas höher, als sonst in den Jahreszeiten. Als wiederkehrendes Event, analog zum Schwärmen, steht jedes Jahr der *Drohnenkrieg* beziehungsweise die *Drohnenschlacht* an, wobei sämtliche Drohnen aus dem Stock

**Winter** ist die Zeit des Notstands. Die Bienen sind in dieser Zeit nicht ausflugfähig und müssen in ihrem Bienenstock warten, bis der Winter endet. Daher muss darauf geachtet werden, dass genug Honig gelagert ist, damit die Kolonie überleben kann. Sollte Regen während dieser Zeit fallen, ist dieser stattdessen Schnee. Es sterben zu dieser Jahreszeit sämtliche Blumen ab, wodurch, selbst wenn eine Biene rausfliegen würde, es keine Möglichkeit gibt, Nahrung zu beschaffen.

#### 6.7.4 Bienen

Die Bienen sind das Herzstück des Spiels und werden analog zu *RimWorld* indirekt gesteuert. Eine Biene hat mehrere Eigenschaften. Sämtliche aufgelisteten Eigenschaften sind visuell dargestellt als Skizze beziehungsweise Mockup in Abbildung 24. Die Klasse der *Bee* kann über das leicht vereinfachte UML-Klassendiagramm in Abbildung 25 ein-

Bee
<ul style="list-style-type: none"> <li>- hexUnit : HexUnit</li> <li>...</li> <li>- maxFoodLevel : int</li> <li>- currentFoodLevel : int</li> <li>- maxWaterLevel : int</li> <li>- currentWaterLevel : int</li> <li>- maxSemenLevel : int</li> <li>- currentSemenLevel : int</li> <li>- ageHours : int</li> <li>- ageDays : int</li> <li>- _hourBorn : int</li> <li>- caste : Caste</li> <li>- job : Job</li> <li>- metamorphosis : Metamorphosis</li> <li>- _assignedJob : JobOrder</li> <li>- _priorities : Dictionary&lt;BeeAction, PriorityValue&gt;</li> <li>- _inventory : Dictionary&lt;Item, int&gt;</li> <li>- doWork : bool</li> <li>- canWork : bool</li> <li>...</li> <li>- lifespan : int</li> </ul>
<ul style="list-style-type: none"> <li>- GenerateStats() : void</li> <li>- GenerateCaste() : void</li> <li>...</li> </ul>

Abbildung 25: Simplifiziertes UML-Klassendiagramm der Bee.cs

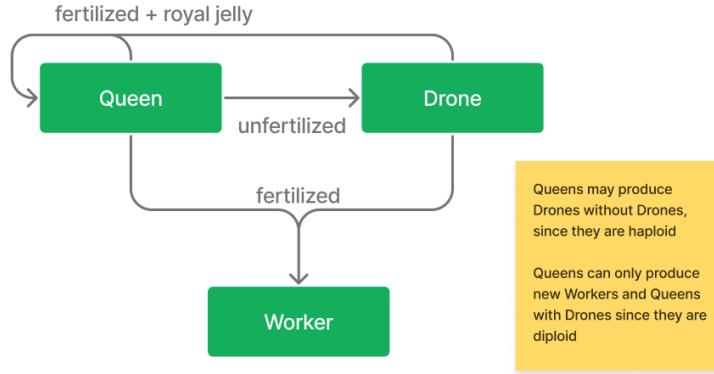


Abbildung 26: Reproduktionsverhalten der Bienen

gesehen werden. In dem dargestellten UML-Diagramm sind einige Felder nicht gezeigt, da diese für äußere Zugriffe unwichtig sind. Außerdem sind die meisten vorhandenen Methoden nicht aufgelistet, da die Anzahl dieser das Diagramm aufblähen würden. Die gesamte Klasse umfasst einen Zeilenumfang von 673 Zeilen.

**Kasten** Es gibt analog zu einer realen Bienenkolonie drei verschiedene Kästen. Jede Biene besitzt eine Variable in Form einer Enum, welche angibt, ob es sich um eine Königin, eine Drohne oder eine Arbeitsbiene handelt. Aufgrund dieser Information werden etliche Entscheidungen getroffen, darunter das gezeigte Modell, wovon es drei verschiedene für jede Kaste gibt, die möglichen ausführbaren Aktionen und die Lebensdauer. Es können kastenspezifisch neue Bienen herangezogen werden. Wie auch in der realen Welt werden für die Arbeitsbienen befruchtete Eier benötigt, für die Königinnen befruchtete Eier und Royal Jelly, und für die Dronen lediglich unbefruchtete Eier. Diese Mechanik ist in Abbildung 26 schematisch dargestellt.

**Name** Jede Biene hat zur besseren Unterscheidung einen zufällig generierten Namen aus einer großen Variation verschiedenster Ethnien. Der Generator entstammt dem Unity Asset Store [52].

**Alter** Das Alter der Biene ist von zentraler Bedeutung. Für die Königin ist es entscheidend einzuschätzen, wie lange diese noch lebt, für die Arbeitsbienen ist es wichtig zu erkennen, welche Arbeiten und Aufgaben diese übernehmen kann. Eine Biene am Ende ihres Lebens ist biologisch nicht mehr in der Lage, die neue Brut zu füttern, und dient lediglich dem Sammeln von Ressourcen.

**Hunger** Eine Biene hat das Bedürfnis nach Nahrung. Der Nahrungs Balken wird mit der Zeit weniger, weshalb die Biene Nahrung beschaffen muss, jedoch nicht aktiv. Ist Nahrung vorhanden und die Biene erreicht einen Schwellenwert, wird sich diese Biene die Nahrung eigenständig aus dem Lager entnehmen. Ist der Wert der Nahrung zu lange auf 0, stirbt die Biene. Die voreingestellte Reihenfolge der verspeisten Nahrung lautet:

Nektar > Honig > Pollen

Royal Jelly wird nicht von den Arbeitern angerührt und wird lediglich von der Königin verspeist, genau so wie für das Heranziehen neuer Königinnen zur Larve gegeben. Stetige Bewegung sorgt für schnelleren Hunger. Die Reihenfolge der Nahrung kann vom Spieler angepasst werden, wodurch mehr Entscheidungsfreiheit gegeben ist und mehr Taktik angewandt werden kann. So könnte ein Spieler möglichst viele Pollen sammeln und diese als primäre Nahrungsquelle nutzen. Natürlich müssen diese Mechaniken ausgiebig getestet werden, sodass verschiedene Strategien nicht zu unausgeglichen sind. Diese Änderung, wie auch das Füttern oder Füttern, ist im gegebenen Prototypen nicht implementiert.

**Durst** Bienen benötigen Flüssigkeit, genauer gesagt Wasser. Dieses wird nicht im Stock gelagert, sondern an Flüssen getrunken. Je mehr sich eine Biene bewegt, desto mehr Flüssigkeit benötigt diese. Sinkt dieser Wert auf 0, wird die Biene nach kurzer Zeit sterben. Die Funktionalität des Trinkens ist im Prototypen noch nicht implementiert, jedoch stirbt eine Biene an Durst, sobald der gegebene Wert auf 0 sinkt.

**Samen** Diese Eigenschaft hat lediglich eine Königin, und gibt an, wie viel Samen der Drohnen noch vorhanden sind um neue und befruchtete Eier zu legen. Es ist wichtig darauf ein Auge zu haben, da ohne verfügbare Samen weder Arbeitsbienen, noch Königinnen herangezogen werden können. Die Aktion zieht aus dem Pool der vorhandenen in der Königin gespeicherten Samen einen Wert ab, sodass die Anzahl der Aktion durch diese nicht greifbare Ressource limitiert ist.

**Inventar** Das Inventar besitzt jede Biene und zeigt an, was diese gerade trägt. Geträgen werden können Nektar, Pollen, Honig, Royal Jelly und Bienenwachs. Die Menge richtet sich nach der Art des Getragenen. Das Inventar ist ein zentraler Punkt der Ressourcen, da Bienen für Aktionen in den meisten Fällen die benötigte Ressource erst aus dem Lager holen müssen, falls nicht bereits im Inventar vorhanden, und dann diese nicht greifbaren Ressourcen weiterverwenden mittels Converter.

**Auftrag** Der momentane Auftrag ist ebenfalls Teil der Biene. Dieser zeigt dem Nutzer, was diese Biene gerade vorhat und wieso sie sich zu der bestimmten Stelle bewegt.

**Genetische Eigenschaften / Traits** Die genetischen Eigenschaften beziehungsweise *Traits* sind immer exakt drei Stück und werden nach [H6] in überschaubarer Anzahl gehalten und so benannt, dass aus dem Namen der Effekt grundlegend ersichtlich ist. Diese können sowohl *positive* als auch *negative* Auswirkungen auf die Effizienz der einzelnen Biene haben, wobei nicht eindeutig erkennbar sein soll, welcher der effektiv beste oder schlechteste Trait ist, der zur Auswahl steht, um [H7] nachzugehen, und Nutzer-eigene Strategien zu fördern. Die möglichen Traits und ihre Wahrscheinlichkeiten sind in [Tabelle 7](#) zu finden. Die Traits sind vererbbar, wobei bei der Vererbung eine Chance besteht, dass ein Trait *mutiert* und deshalb in einen anderen, zufälligen, geändert wird. Wird eine Königin von mehreren verschiedenen Drohnen besamt, werden die Erbinformationen gespeichert. Ob der jeweilige Trait nun von der Seite des Vaters oder der Seite der Mutter übernommen wird, ist gleichverteilt 50:50. Seien die besamenden Drohnen folgende:

Tabelle 7: Mögliche Traits einer Biene

Arbeitsfreudig / Arbeits scheu (Alle)	Erledigt alle Aufgaben 10% schneller / langsamer	12%
Sammelfreudig / Sammelscheu (Arbeitsbiene)	Sammelt Ressourcen 15% schneller / langsamer	12%
Fingerfertig / Grobmotorisch (Arbeitsbiene)	Stellt neue Ressourcen 10% schneller / langsamer her.	12%
Sparsam / Freigiebig (Arbeitsbiene)	Verbraucht beim Füttern etwas weniger / mehr Nahrung	12%
Empfänglich / Unempfänglich (Königin)	Benötigt weniger / mehr Drohnen um die Samen zu füllen und hat mehr / weniger Kapazität.	12%
Gesegnet / Verflucht (Alle)	Lebt ein längeres / kürzeres Leben.	12%
Wasserspeicher / Verdampfer (Alle)	Benötigt weniger / mehr Wasser und verbraucht weniger / mehr.	12%
Kleiner / Großer Magen (Alle)	Benötigt weniger / mehr Nahrung und verbraucht weniger / mehr.	12%
Königliche Immunität (Drone)	Die Drohne ist ausgenommen von der Drohenschlacht.	2%
Schillernd (Alle)	Die Biene sieht anders aus als gewöhnliche Bienen.	2%

- Drohne A: Schillernd, Arbeitsfreudig, Gesegnet
- Drohne B: Schillernd, Arbeitsfreudig, Verflucht
- Drohne C: Schillernd, Großer Magen, Verdampfer

Und die Königin besäße folgende Eigenschaften:

- Königin: Schillernd, Großer Magen, Empfänglich

Dann gelten folgende Chancen für die Traits der Brut:

- Schillernd: 100%
- Empfänglich:  $\frac{1}{2}$
- Großer Magen:  $\frac{1}{3} + \frac{1}{2} = \frac{5}{6}$
- Arbeitsfreudig:  $\frac{2}{3} \times \frac{1}{2} = \frac{1}{3}$
- Gesegnet:  $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$
- Verflucht:  $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$
- Verdampfer:  $\frac{1}{3} \times \frac{1}{2} = \frac{1}{6}$

Die Traits, welche speziell auf eine gewisse Art von Kaste gelten, werden im Hintergrund gespeichert und es wird ein neuer Trait gezogen. Dieser wird sich vorgemerkt. Sollte die Arbeitsbienenlarve also zur Königin herangezogen werden, wird der vorgemerkte Trait durch den Erstgezogenen ersetzt. Die Traits sind bereits namentlich implementiert, es werden drei bei der Geburt der Biene zugewiesen. Allerdings haben diese im Prototypen noch keine weiteren Effekte und keine passenden Grafiken. Auch werden die Traits noch nicht genetisch vererbt, sondern zufällig gezogen, was sich im Laufe der Entwicklung noch ändern wird.

**Mutation** Damit eine Diversität entstehen kann, und nicht stetig die gleichen Traits in der Kolonie weitervererbt werden, bedarf es der Möglichkeit einer Mutation. Es besteht eine geringe Chance, dass *nach dem Ziehen* eines vererbten Traits, dieser Trait durch einen völlig anderen ersetzt wird. Diese Mechanik ist in dem Prototypen noch nicht gegeben.

### 6.7.5 Aktionen

Es gibt 6 verschiedene, vom Spieler ausführbare, Anordnungen. Diese werden verwendet, um die Bienen indirekt zu steuern und den Stock somit möglichst am Leben zu erhalten. Von den Aktionen sind im Prototyp das Sammeln, das Bestäuben, das Eier legen und das Abbrechen funktional.

**Sammeln** wird verwendet, um von den in der Spielwelt vorhandenen Blumen den Nektar und die Pollen zu extrahieren. Diese Aktion ist lediglich von Arbeitsbienen im späten Alter ausführbar.

**Bestäuben** verwendet Pollen, um neue Blumen zu erstellen, welche als weitere Quelle von Nektar und Pollen agieren. Dies kann sinnvoll sein, sollten nicht genug Blumen vorhanden sein, etwa im Frühling nach einem harten Winter. Diese Aktion ist ebenfalls nur von älteren Bienen ausführbar.

**Besamen** ist eine Aktion, welche lediglich den Drohnen vorbehalten ist. Mit Auswahl der Königin finden sich nacheinander Drohnen, bis die Königin keine weiteren Samen speichern kann.

**Eier legen** wird lediglich von der Königin ausgeführt. Nach Auswahl einer passenden Brutwabe (normale Brutwabe oder königliche Brutwabe), fliegt die Königin zu der jeweiligen Stelle und legt ein Ei ab. Ob dieses befruchtet ist oder nicht hängt davon ab, ob die Königin Samen gespeichert hat.

**Abbrechen** wird verwendet, um eine bereits ausgeführte Anordnung zurückzuziehen. Anders als die anderen Aktionen ist hierfür keine Biene nötig.

**Zerstören** kann verwendet werden, um vom Spieler gebaute Strukturen wieder zu vernichten. Diese Aktion ist lediglich von mittelalten Arbeitsbienen ausführbar und ist auf den Bienenstock beschränkt.

### 6.7.6 Strukturen

Es gibt, analog zu den Aktionen, 6 verschiedene, baubare Strukturen. Diese dienen in den meisten Fällen als *Converter* und Arbeitsstätte der Bienen, aber auch als Lagerplatz. Sämtliche baubare Strukturen sind [Abbildung 23](#) zu entnehmen.

**Evaporator** ist ein Converter, um aus Nektar Honig herzustellen, und dient der Erfüllung von [H12]. Dieser wird, wie in [Abbildung 23](#) erkennbar, aus Pollen hergestellt. Mit Klick auf den gebauten Evaporator können Aufträge erstellt werden, wodurch festgelegt werden kann, wie viel Honig hergestellt werden soll. Für diese Aufträge müssen Bienen am Evaporator arbeiten.



*Abbildung 27: Vier verschiedene Stadien des Bienenmodells von links nach rechts: Ei, Adult, Puppe und Larve*

**Mixer** ist ebenfalls ein Converter, erfüllt ebenfalls [H12] und wird verwendet, um aus Honig und Pollen Bienenwachs herzustellen. Zum Bau dieser Struktur wird Honig verwendet, wodurch ein zuvor gebauter Evaporator sinnvoll ist. Das Auftragssystem ist analog zum Evaporator.

**Refiner** wird verwendet, um aus Honig und Pollen Royal Jelly herzustellen, wobei für den Bau dieser Struktur Bienenwachs verwendet wird. Dadurch kann es sinnvoll sein, einen Mixer gebaut zu haben. Das Auftragssystem ist analog zu dem Evaporator und dem Mixer und dient ebenfalls der Erfüllung von [H12].

**Lagerwaben** sind essenziell zum Speichern und Lagern bestimmter Ressourcen. In diesen Lagerwaben können Nektar, Pollen, Honig, Royal Jelly und Bienenwachs gespeichert werden, wobei eine Lagerwabe nur eine Art von Ressource halten kann, und ebenfalls ein quantitatives Limit für diese Ressource besitzt. Für den Bau einer Lagerwabe wird Bienenwachs verwendet. Diese Lagerstruktur dient als klare, baubare Struktur der Erfüllung von [H9].

**Brutwaben** werden für die Fortpflanzung beziehungsweise Eierhaltung der Königin verwendet. Junge Arbeitsbienen werden lediglich angeordnet Nektar oder Honig zu liefern. Wird die Larve zu einer Puppe, muss außerdem von einer Arbeitsbiene die Brutwabe durch Bienenwachs bedeckt werden.

**Königliche Brutwaben** sind ähnlich zu den normalen Brutwaben, mit der Ausnahme, dass lediglich befruchtete Eier dort gelegt werden können, und die jungen Arbeitsbienen zu der Larve Royal Jelly geben, falls vorhanden.

#### 6.7.7 Brutvorgang

Um neue Bienen herzustellen werden, wie bereits zuvor erläutert und in [Abbildung 26](#) erkennbar, verschiedene Bienen für verschiedene Resultate benötigt. Wird ein Ei in eine Brutkammer gelegt, beginnt ab diesem Zeitpunkt die Metamorphose, analog zu den in [Sektion 6.1.4 Metamorphose](#) erläuterten Vorgängen. Das gelegte Ei wird nach einer gegebenen Zeit zu einer Larve, danach zu einer Puppe und anschließend zu einer voll ausgewachsenen Biene. Die Modelle einer einzelnen Einheit sind in [Abbildung 27](#) zu erkennen.

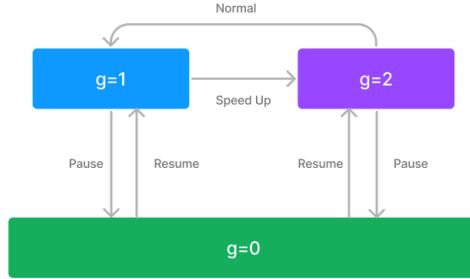


Abbildung 28: Zeitsteuerung Zustandsdiagramm

**Stadium 1: Ei** Das Ei liegt nach dem Legen in der Brutwabe. Zu diesem Stadium ist lediglich erkennbar, ob befruchtet oder unbefruchtet. Die Wabe muss ein Mal mit Nahrung, vorzugsweise Nektar oder Honig, gefüllt werden, damit das Ei überlebt. Nach einigen Tagen schlüpft daraus eine Larve und die Nahrung in der Brutwabe ist verbraucht.

**Stadium 2: Larve** Die Larve ist das Stadium, in welchem Royal Jelly dazugegeben werden kann. Liegt das Ei in einer Königlichen Brutkammer, wird einmalig Royal Jelly statt gewöhnlicher Nahrung dazugegeben. Damit entwickelt sich eine königliche Puppe statt einer herkömmlichen und die Nahrung wird verbraucht.

**Stadium 3: Puppe** Die Puppe benötigt, anders als die anderen Stadien, zusätzlich zur Nahrung noch Bienenwachs, welches die Brutwabe bedeckt. Zuerst muss Nahrung dazugegeben, danach die Wabe bedeckt werden. Sobald die Biene alt genug ist, schlüpft sie aus der Puppe und bricht durch die Wachsschicht. Eine erwachsene Biene ist somit herangereift.

**Stadium 4: Ausgewachsen** Das ausgewachsene Stadium ermöglicht sämtliche Aktionen einer Biene. In diesem Stadium befinden sich die meisten Bienen.

### 6.7.8 Zeitsteuerung

Analog zu RimWorld wird nach [H21] eine Zeitsteuerung zur besseren Kontrolle eingeführt. Zur Auswahl stehen dabei *Pause* beziehungsweise *Fortführen* (engl. *Resume*) als sich abwechselnde Zustände, und weiterhin *Normal* und *Erhöht*, wobei *Pause* die Geschwindigkeit des Spiels auf  $g = 0$  setzt, *Normal* die Geschwindigkeit auf  $g = 1$  setzt, und *Erhöht* die Geschwindigkeit auf  $g = 2$  setzt. Allerdings wird eine intelligenter Logik verwendet, um das Nutzererlebnis zu erhöhen, welche in Abbildung 28 veranschaulicht wird. Pausiert der Nutzer das Spiel, während die Geschwindigkeit auf  $g=2$  steht, wird sich dieser Wert gemerkt, und bei Klick auf den nun vorhandenen *Resume*-Button wiederhergestellt. Analog funktioniert dieser Prozess auch bei  $g=1$ .

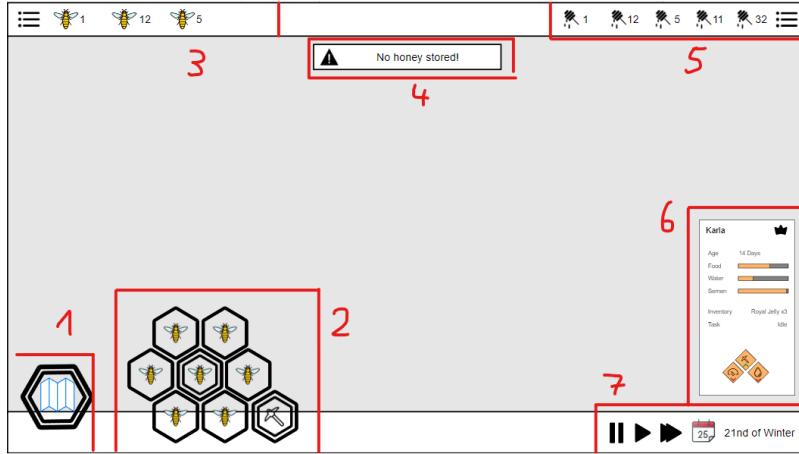


Abbildung 29: Erster Prototyp des Spielinterfaces

## 6.8 Interface

Für alle genannten Mechaniken und Informationen muss ein passendes Interface erstellt werden. Der erste Prototyp ist erkennbar in Form eines Mockups in Abbildung 29. Das Interface wird aus einzelnen Komponenten bestehen, welche in der Grafik durch die rot markierten Bereiche erkennbar sind und in folgenden Paragraphen mit (X) referenziert werden, wobei X dem roten Bereich mit der Zahl X in Abbildung 29 zugeordnet wird.

### 6.8.1 Karte wechseln

Um zwischen der Außenwelt und dem inneren des Bienenstocks hin- und herzuwechseln gibt es den Button, welcher in Bereich (1) von Abbildung 29 dargestellt wird. Befindet sich die Kamera in der Außenwelt, ist auf diesem Knopf ein Bienenstock abgebildet, befindet sich die Kamera im Bienenstock ist dort ein Baum abgebildet, welcher symbolisch für die Außenwelt steht.

### 6.8.2 Action Buttons

Zur Steuerung der Bienen bedarf es verschiedener auswählbaren Aktionen, welche bereits zuvor erläutert wurden. Es gibt insgesamt 12 verschiedene Aktionen, welche in zwei hexagonale Menüs aufgeteilt sind, zum einen die Tätigkeiten und zum anderen die Strukturen. Diese Menu Buttons sind mit einem zweiten, inneren Hexagon markiert und in Bereich (2) auffindbar. Klickt man auf den jeweiligen Menü-Knopf öffnen sich die jeweils 6 untergeordneten Buttons. Ist währenddessen das andere Menü geöffnet, wird es geschlossen. Es kann somit maximal nur ein Menü gleichzeitig aktiv sein, aber es können beide zeitgleich geschlossen werden. Für den in der Arbeit angefertigten Prototypen werden lediglich die Aktionen *Gather*, *Pollinate*, *Build Storage*, *Lay Eggs* und *Cancel* funktional implementiert. Die ursprüngliche Idee war es, sämtliche Aktionen in ein Menü zu verpacken, wurde dann jedoch aufgrund der schlechteren Übersicht verworfen und aufgeteilt.

### **6.8.3 Bee List**

Für einen besseren Überblick der Anzahl verschiedener in der Kolonie vorhandener Kästen wird eine kleine Übersicht dargestellt (3). Zum jetzigen Zeitpunkt werden dort 6 verschiedene Werte angezeigt, je ein Element pro Kaste und je ein Element pro prä-adulter Biene je Kaste. Der Button an der linken Seite öffnet die Liste der Prioritäten, welche bereits zuvor erläutert wurde (vgl. [Abbildung 22](#)). Die Grafiken der jeweiligen Kaste oder des Jobs sind noch nicht unterschiedlich dargestellt.

### **6.8.4 Alerts**

Damit der Spieler leichter Überblick über kritische Situationen, wie auch stattfindende Events, haben kann, wird in der Mitte des Bildschirms eine kleine Benachrichtigung visualisiert (4). Ein Alert bleibt für einen kurzen Moment und verschwindet dann wieder, der Spieler wird damit angeregt, aufmerksamer zu spielen und bedient sich der Hypothese [H8]. In dem Prototyp kann mit Drücken der Taste 9 auf der Tastatur für Testzwecke ein kleiner Alert aufgerufen werden. In der späteren Implementierung können sehr einfach neue Alerts ergänzt werden.

### **6.8.5 Resource List**

Analog zu RimWorld wird dem Spieler der globale Speicherzustand aller Ressourcen angezeigt (5). Es werden also die Summen aller verschiedenen sammel- und tragbaren Ressourcen angezeigt: *Nectar*, *Pollen*, *Wax*, *Honey* und *Royal Jelly*. Der Button auf der rechten Seite ist im Prototypen noch nicht funktional, soll aber eine Statistik offenbaren, um den Verlauf der Ressourcenstände zu visualisieren. Dies soll dem Spieler etwas mehr Überblick über den Werteverlauf der Kolonie bieten. Da das Lager noch nicht funktional, sondern lediglich dekorativ ist, wird jederzeit bei jeder Ressource eine 0 angezeigt.

### **6.8.6 Infafenster**

An der rechten Seite des Bildschirms ist ein Infafenster auffindbar, welches Informationen zum ausgewählten Objekt darstellt. In dem Mockup des Prototypen ist das angezeigte Infafenster jenes, welches bei Auswahl einer Biene gezeigt wird (6). Es werden unter anderem Hunger, Durst, Traits und momentane Aufgabe dargestellt.

### **6.8.7 Time Control**

Ebenfalls analog zu RimWorld ist eine Zeitsteuerung implementiert, welche bereits zuvor erläutert wurde. Auf der rechten Seite dieser Steuerung wird das derzeitige Datum angezeigt in dem Format *Year A, Day B of C, Dh*, wobei A das derzeitige Jahr, B der momentane Tag, C die derzeitige Jahreszeit und D die momentane Stunde ist (7). Wie bereits zuvor aufgezeigt gibt es drei verschiedene Grade der Zeit, g=0, g=1 und g=2.

### **6.8.8 Start Screen**

Das Interface, welches dem Spieler als allererstes gezeigt wird, ist der Start Screen, welcher für den Anfang sehr simpel gehalten ist nach Hypothese [H1]. Es werden lediglich zwei Buttons zum Erstellen und Laden eines Spielstands angezeigt, ein Hintergrund



Abbildung 30: Start Screen des Spiels

des Spiels und eine Überschrift mit dem Titel des Spiels [Abbildung 30](#). Das Laden und Speichern eines Spielstands sind im Prototyp noch nicht implementiert, die Fundamente dafür sind jedoch bereitgestellt.

#### 6.8.9 Spielwelt

Der hellgraue Bereich in [Abbildung 29](#) stellt die interaktive Spielwelt dar, in welcher die Aktionen angegeben werden können und die Einheiten sich dementsprechend verhalten.

### 6.9 Unit Tests

Um die Stabilität der Anwendung im Allgemeinen zu gewährleisten, und die Zuverlässigkeit der einzelnen Funktionen sicherzustellen, werden, auch im Bereich des Game Development, sogenannte *Unit Tests* verwendet. Es gibt dabei mehrere Vorteile dieser Tests. Man kann beispielsweise mit diesen Tests Szenarien abdecken, welche nur sehr selten im produktiven Anwendungsfall auftreten würden, sogenannte *Corner Cases*. Außerdem ist die Chance deutlich geringer, dass bereits aufgetretene Bugs, welche durch Unit Tests bereits behoben wurden, erneut auftreten. Der größte Nachteil dieser Tests ist der Fakt, dass deutlich mehr Code geschrieben werden muss, was sehr viel Zeit einfordert [29]. In einem industriellen Umfeld und bei der Entwicklung einer marktreifen Anwendung ist dieser Schritt allerdings unerlässlich. Es ist daher sinnvoll, auch im weiteren Verlauf, Unit Tests für die gesamte Anwendung nachträglich hinzuzufügen. Aufgrund der kurzen Zeitspanne und dem bereits großen Umfang des Quellcodes wird in dieser Arbeit bewusst auf diese Art des Testings verzichtet.

## 7 Fazit

# Literatur

- [1] Ernest W. Adams und Joris Dormans. *Game mechanics: Advanced game design.* New Riders, 2012.
- [2] Ernest W. Adams und Joris Dormans. *Game mechanics: Advanced game design.* New Riders, 2012, S. 61–62.
- [3] *Age of Empires Official Page*. URL: <https://www.ageofempires.com/games/aoe/> (besucht am 03.08.2022).
- [4] *Age of Empires Ressourcen*. URL: <https://ageofempires.fandom.com/wiki/Resource> (besucht am 04.08.2022).
- [5] *Age of Empires Technologien*. URL: [https://ageofempires.fandom.com/wiki/Technology\\_\(Age\\_of\\_Empires\)](https://ageofempires.fandom.com/wiki/Technology_(Age_of_Empires)) (besucht am 05.08.2022).
- [6] *AllGame Rezension zu Sim City 2000*. URL: <https://web.archive.org/web/20141115005407/http://www.allgame.com/game.php?id=11771&tab=review> (besucht am 05.08.2022).
- [7] *Amits Thoughts on Grids*. URL: <http://www-cs-students.stanford.edu/~amitp/game-programming/grids/> (besucht am 21.08.2022).
- [8] *Arcology in Sim City 2000*. URL: <https://simcity.fandom.com/wiki/Arcology> (besucht am 05.08.2022).
- [9] *Backstories*. URL: <https://rimworldwiki.com/wiki/Backstories> (besucht am 09.08.2022).
- [10] Nina Baur und Blasius Jörg. *Handbuch Methoden der Empirischen Sozialforschung*. Springer Fachmedien Wiesbaden, 2014.
- [11] Thomas Borovskis. *PC Games Test, Spiel des Monats*. URL: <https://www.pcgames.de/15-Jahre-PC-Games-Thema-206368/Tests/Age-of-Empires-612590/> (besucht am 04.08.2022).
- [12] Josh Bycer. *How Event-Driven Game Design Keeps The Player Guessing*. Dezember 2016. URL: <https://game-wisdom.com/critical/event-driven-game-design> (besucht am 08.08.2022).
- [13] *Catlike Coding Hex Map Tutorial*. URL: <https://catlikecoding.com/unity/tutorials/hex-map/> (besucht am 14.09.2022).
- [14] *Chess Piece Value*. URL: <https://www.chess.com/terms/chess-piece-value> (besucht am 07.08.2022).
- [15] *Civilization 1*. URL: <https://web.archive.org/web/20150918012513/https://www.civilization.com/en/games/civilization-i/> (besucht am 04.08.2022).
- [16] *Colonist*. URL: <https://rimworldwiki.com/wiki/Colonist> (besucht am 09.08.2022).
- [17] David Curry. *Discord Revenue and Usage Statistics (2022)*. URL: <https://www.businessofapps.com/data/discord-statistics/> (besucht am 15.08.2022).
- [18] *Definition einer Wirtschaft*. URL: <https://wirtschaftslexikon.gabler.de/definition/wirtschaft-54080> (besucht am 08.08.2022).

- [19] *Definition Metamorphose* (engl. *metamorphosis*). URL: <https://www.merriam-webster.com/dictionary/metamorphosis> (besucht am 17.08.2022).
- [20] *Die Bienenkönigin*. URL: <https://www.bee-careful.com/de/initiative/die-bienenkoenigin/#:~:text=Neben%20dem%20Ablegen%20der%20Eier,ausch%20w%C3%A4hrend%20des%20Schw%C3%A4rmens%20%E2%80%93%20zusammen>. (besucht am 17.08.2022).
- [21] *Entstehungsgeschichte von Civilization*. URL: [https://web.archive.org/web/20140222210101/http://www.gamasutra.com/view/feature/1523/the\\_history\\_of\\_civilization.php?print=1](https://web.archive.org/web/20140222210101/http://www.gamasutra.com/view/feature/1523/the_history_of_civilization.php?print=1) (besucht am 04.08.2022).
- [22] *Ernährung von Bienen*. URL: <https://honig-und-bienen.de/ernaehrung-der-biene/> (besucht am 15.08.2022).
- [23] Joel R. Evans und Anil Mathur. „The value of online surveys“. In: *Internet Res.* 15 (2005), S. 195–219.
- [24] *Events*. URL: <https://rimworldwiki.com/wiki/Events> (besucht am 09.08.2022).
- [25] *Featured Creature: European Honey Bee*. URL: [https://entnemdept.ufl.edu/creatures/misc/BEES/euro\\_honey\\_bee.htm](https://entnemdept.ufl.edu/creatures/misc/BEES/euro_honey_bee.htm) (besucht am 15.08.2022).
- [26] Karl von Frisch. *Aus dem Leben der Bienen*. Springer, 1977.
- [27] *GameRankings Metacritic*. URL: <https://web.archive.org/web/20160305152046/http://www.gamerankings.com/pc/90380-age-of-empires/index.html> (besucht am 04.08.2022).
- [28] *Geschlechtsbestimmung bei Honigbienen*. URL: <https://www.lernhelfer.de/schuelerlexikon/biologie-abitur/artikel/geschlechtsbestimmung-bei-honigbienen#> (besucht am 17.08.2022).
- [29] Francois Guibert. *Unit testing in video games*. Juni 2017. URL: <https://www.gamedeveloper.com/programming/unit-testing-in-video-games> (besucht am 21.09.2022).
- [30] Jared Halpern. *The What and Why of Game Engines*. Dez. 2018. URL: <https://medium.com/@jareddehalpern/the-what-and-why-of-game-engines-f2b89a46d01f#:~:text=Game%20engines%20provide%20tremendous%20efficiency,entirely%20on%20writing%20gameplay%20code.> (besucht am 21.09.2022).
- [31] Dan Hastings. *What Is Fog of War?* Oktober 2019. URL: <https://nerdburglars.net/what-is-fog-of-war/> (besucht am 08.08.2022).
- [32] Karen Holtzblatt und Hugh Beyer. *Contextual design: A customer-center approach to software design*. Morgan Kaufmann, 1997, S. 64–66.
- [33] *Homozygot und Heterozygot*. URL: <https://studyflix.de/biologie/homozygot-und-heterozygot-2664> (besucht am 17.08.2022).
- [34] *IGDB Eintrag von Age of Empires*. URL: <https://www.igdb.com/games/age-of-empires> (besucht am 02.08.2022).
- [35] *IGDB Eintrag von Civilization*. URL: <https://www.igdb.com/games/sid-meiers-civilization> (besucht am 04.08.2022).

- [36] *IGDB Eintrag von Sid Meier's Civilization: Beyond Earth*. URL: <https://www.igdb.com/games/sid-meiers-civilization-beyond-earth> (besucht am 04.08.2022).
- [37] *IGDB Eintrag von Sim City 2000*. URL: <https://www.igdb.com/games/simcity-2000> (besucht am 05.08.2022).
- [38] Petri Lankoski und Staffan Bjork. *Game research methods: An overview*. ETC Press, 2015.
- [39] *Liste aller Sim City 2000 Gebäude*. URL: [https://simcity.fandom.com/wiki/List\\_of\\_SimCity\\_2000\\_buildings](https://simcity.fandom.com/wiki/List_of_SimCity_2000_buildings) (besucht am 05.08.2022).
- [40] *Liste aller verzeichneten Civilization Spiele*. URL: <https://www.igdb.com/collections/civilization> (besucht am 04.08.2022).
- [41] *Liste an Ressourcen in Civilization*. URL: [https://civilization.fandom.com/wiki/List\\_of\\_resources\\_in\\_Civ1](https://civilization.fandom.com/wiki/List_of_resources_in_Civ1) (besucht am 04.08.2022).
- [42] Ed Lomas. *Review: Sim City 2000*. SSega Saturn Magazin", Ausgabe Nummer 1, November 1995. 1995.
- [43] *MacUser Rezension zu Sim City 2000*. 1995. URL: [https://web.archive.org/web/20000122070956/http://macuser.zdnet.com/mu\\_1295/personal/gameroom.html](https://web.archive.org/web/20000122070956/http://macuser.zdnet.com/mu_1295/personal/gameroom.html) (besucht am 05.08.2022).
- [44] Laurent Mathevet, Jacopo Perego und Ina Taneva. *On Information Design in Games*. Feb. 2017. DOI: [10.13140/RG.2.2.35431.34724](https://doi.org/10.13140/RG.2.2.35431.34724).
- [45] *Mental Break*. URL: [https://rimworldwiki.com/wiki/Mental\\_break](https://rimworldwiki.com/wiki/Mental_break) (besucht am 09.08.2022).
- [46] *Most used Engines*. URL: <https://itch.io/game-development/engines/most-projects> (besucht am 21.09.2022).
- [47] *Nahrung der Bienen*. URL: <https://www.die-honigbiene.de/nahrung-der-bienen.html> (besucht am 15.08.2022).
- [48] *Nicht alles, was nach Frucht klingt, ist automatisch auch gesund. Anders als häufig vermutet, ist Fructose weder "besser" noch "gesünder als weißer Industriezucker*. URL: <https://www.minimed.at/medizinische-themen/stoffwechsel-verdauung/fructose/#:~:text=Gleich%20wie%20Glucose%2C%20der%20Traubenzucker,so%20hohe%20S%C3%BC%C3%9Fkraft%20wie%20Traubenzucker.> (besucht am 15.08.2022).
- [49] *PC Player Magazin*. 1997. URL: [https://archive.org/details/PC-Player-German-Magazine-1997-12/page/n77\(mode/1up](https://archive.org/details/PC-Player-German-Magazine-1997-12/page/n77(mode/1up) (besucht am 04.08.2022).
- [50] Craig Pearson. *Discord*. URL: <https://store.epicgames.com/en-US/news/what-is-discord-and-what-is-it-used-for#:~:text=Discord%20is%20a%20free%20communications,to%20connect%20with%20people%20online.> (besucht am 15.08.2022).
- [51] *Power Play*. 1997. URL: [https://archive.org/details/powerplaymagazine-1997-11/page/n121\(mode/1up](https://archive.org/details/powerplaymagazine-1997-11/page/n121(mode/1up) (besucht am 04.08.2022).

- [52] *Random Generator* - Itay Sagui. URL: <https://assetstore.unity.com/packages/tools/input-management/random-generator-118768> (besucht am 15.09.2022).
- [53] *Repository des Prototyp*. URL: <https://github.com/ProgFroz/hexabees> (besucht am 18.09.2022).
- [54] *Resource Management Games – Steam Marketing Tool*. URL: <https://games-stats.com/steam/?tag=resource-management> (besucht am 21.09.2022).
- [55] *Resource Management, Game Mechanics Fandom*. URL: [https://gamenmechanics.fandom.com/wiki/Resource\\_Management](https://gamenmechanics.fandom.com/wiki/Resource_Management) (besucht am 07.08.2022).
- [56] *Resources*. URL: <https://rimworldwiki.com/wiki/Resources> (besucht am 10.08.2022).
- [57] *Rezension Civilization AllGame*. URL: <https://web.archive.org/web/20141115032525/http://www.allgame.com/game.php?id=20216&tab=review> (besucht am 04.08.2022).
- [58] *Rezension Civilization Game Informer*. URL: <https://web.archive.org/web/19971120012730/http://www.gameinformer.com/oct95/civ.htm> (besucht am 04.08.2022).
- [59] *Rezension Civilization Next Generation*. "Civilization", Next Generation Ausgabe Nummer 10, Oktober 1995. 1995.
- [60] Cory Roberts. *Timeline of SimCity*. Jan. 2018. URL: <https://medium.com/shinkansen-retrogaming/timeline-of-simcity-5dbc81e2b764> (besucht am 05.08.2022).
- [61] *Screenshot Game Over in Sim City 2000*. URL: <https://imgur.com/a/fuf0cf2> (besucht am 05.08.2022).
- [62] SergiuHellDragoonHQ. *SimCity 2000 - Gameplay (PC/HD)*. Zeitstempel 9:34. URL: <https://www.youtube.com/watch?v=jk7BkwsgX8> (besucht am 05.08.2022).
- [63] Yvens Serpa. *The Pillars of Internal Economy — An Introduction to Game Economics*. Oktober 2020. URL: <https://yvensserpa.medium.com/the-pillars-of-internal-economy-an-introduction-to-game-economics-38d8e73d5afa> (besucht am 08.08.2022).
- [64] *Siegbedingungen Civilization*. URL: [https://civilization.fandom.com/wiki/Victory\\_\(Civil1\)](https://civilization.fandom.com/wiki/Victory_(Civil1)) (besucht am 04.08.2022).
- [65] *Sim City 2000 Tipps und Tricks*. 1996. URL: <https://www.somacon.com/p606.php> (besucht am 05.08.2022).
- [66] *Sim City 2000, EA*. URL: <https://www.ea.com/de-de/games/simcity/simcity-2000#description> (besucht am 05.08.2022).
- [67] *Traits*. URL: <https://rimworldwiki.com/wiki/Traits> (besucht am 09.08.2022).
- [68] *Unity*. URL: <https://unity.com/> (besucht am 21.09.2022).
- [69] *User Manual für Sim City 2000, digitalisiert*. URL: <https://classicreload.com/sites/default/files/sim-city-2000-manual.pdf> (besucht am 05.08.2022).

- [70] *Vergleich Civilization Spiele*. URL: <https://www.pcgamesn.com/best-civilization-games> (besucht am 04.08.2022).
- [71] *Was ist eine Arkologie*. URL: <https://spiegato.de/was-ist-eine-arkologie> (besucht am 05.08.2022).
- [72] *Was machen Honigbienen im Winter?* URL: <https://www.landwirtschaft.de/landwirtschaft-verstehen/haetten-sies-gewusst/tierhaltung/was-machen-honigbienen-im-winter#:~:text=Honigbienen%20machen%20keinen%20Winterschlaf.,der%20Traube%20sitzt%20die%20K%C3%B6nigin.> (besucht am 17.08.2022).
- [73] *What is Resource Management and Why Is It Important?* URL: <https://www.planview.com/resources/guide/resource-management-software/resource-management-leverage-people-budgets/> (besucht am 07.08.2022).
- [74] *Wie entsteht ein Bienenschwarm?* URL: <https://www.bee-careful.com/de/initiative/bienenschwarm/> (besucht am 17.08.2022).
- [75] *Wie lange leben Bienen?* URL: <https://bienen.info/wie-lange-leben-bienen/> (besucht am 17.08.2022).
- [76] *Wie machen Bienen Honig aus Nektar?* URL: [https://www.t-online.de/leben/familie/id\\_69540058/wie-machen-bienen-honig-aus-nektar-.html](https://www.t-online.de/leben/familie/id_69540058/wie-machen-bienen-honig-aus-nektar-.html) (besucht am 15.08.2022).
- [77] *Wildbienen und Honigbienen, Unterschiede und Gemeinsamkeiten*. URL: <https://www.deutschewildtierstiftung.de/aktuelles/wildbienen-und-die-honigbiene-unterschiede-und-gemeinsamkeiten> (besucht am 15.08.2022).
- [78] Critical Kate Willaert. *THE SUMERIAN GAME: THE MOST IMPORTANT VIDEO GAME YOU'VE NEVER HEARD OF*. URL: <https://www.acriticalhit.com/sumerian-game-most-important-video-game-youve-never-heard/> (besucht am 21.09.2022).

# Notizen

## Offene Fragen

- Age of Empires und Civ noch ausbauen?
- Genauer auf alle Nachfolger und Vorgänger eingehen, mit Publishern, Developern und Änderungen?
- Wirklich alle verfügbaren Rezensionen suchen und reinpacken?
- Umfrage zu gespielten Klassikern über Discord?
- Noch Anno 1602 als Klassiker hinzufügen?
- Notable Mentions? Crusader Kings, Anno 1602,
- Für Verwaltung noch Quelle suchen? Schwer
- Indirketes vs Direktes steuern von Einheiten (AoE vs RimWorld)
- Wie kann man Hexabees steigend schwerer machen?
- Welche Belohnung kann den Spieler zufrieden stellen?
- Noch den Bienenstockbau unter 4.1. erläutern? Waben, Wachs?

Wo unterscheiden sich Resource Management Games von anderen spielen - Main Gameplay Loop

## Ideen

- HexaBees turn based, oder RTS?
- Falls RTS dann
- Für Neuerscheinungen:
- RimWorld
- (Stellaris)
- Factorio
- Frostpunk (Moral als abstrakte Ressource / Ansteigende Gefahr)
- Age of Darkness (Ansteigende Gefahr)
- Northgard
- Potioncraft (maybe spielen)
- Magic: The Gathering

- Siedler von Catan
- Alle Klassiker mit den neusten Version vergleichen
- **PriorittenfrHypothesen**
- Bei Sterben der Königin kein direktes GameOver sondern weitere Chancen
- Mehr auf Entwicklungsprozess eingehen, ursprüngliche Idee und wieso überarbeitet.

## Fremdwörter

- Game engine
- Real Time Strategy
- launch arcos
- Singleplayer (Game)
- Isometrisch
- Domination
- Space Race
- tangible
- intangible
- concrete
- abstract
- source
- drain
- converter
- trader
- fog of war
- event
- replayability

## **Abkürzungen**

- SAT - Sega Saturn, Konsole
- RTS
- R - Residents
- C - Commercial
- I - Industries
- KI
- 

## **Interview für RimWorld**

-