Hochschule Worms

Fachbereich Informatik

Studiengang M.Sc. Mobile Computing

Parametrisierung adaptiver Lernspiele am Beispiel eines adaptiven Wimmelbildspiels

Thesis zur Erlangung des akademischen Grades

Master of Science

vorgelegt von

Jonas Steinbach

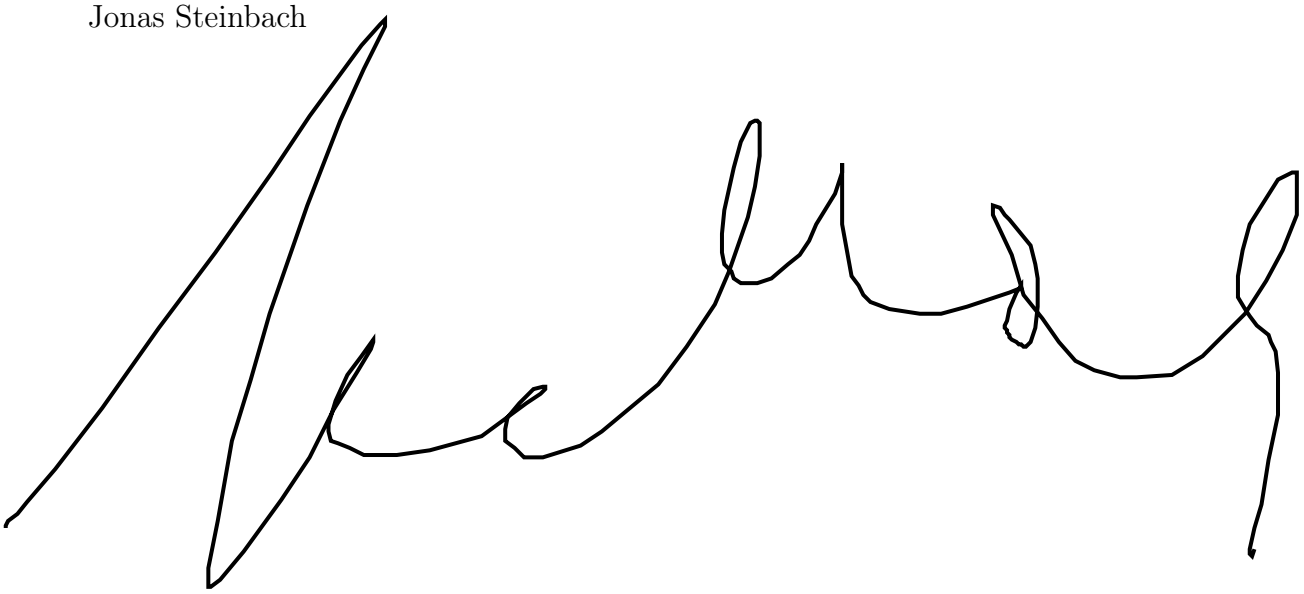
Gutachter:	Prof. Dr. Heinemann
Zweitgutachter:	Prof. Dr. König/Wiebel
	M. Sc. Daniel Atorf
Beabeitungszeitraum:	WS 20/SS 21
Abgabedatum:	30.4.2021
Sperrvermerk:	Nein

Selbstständigkeitserklärung

Entsprechend §23 Abs.6 der Rahmenprüfungsordnung der Hochschule Worms versichere ich hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und ausschließlich die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 30. April 2021

Jonas Steinbach



Die Gedanken sind frei,
Wer kann sie errathen?
Sie rauschen vorbei
Wie nächtliche Schatten.
Kein Mensch kann sie wißen,
Kein Jäger sie schießen.
Es bleibet dabei:
Die Gedanken sind frei.

Danksagung

Ich bedanke mich herzlich bei Frau Heinemann, die mir (metaphorisch!) das Fliegen beigebracht hat; bei Daniel Atorf, der sich regelmäßig mit mir zur frühen Morgenstunde mittags um Eins zusammengesetzt hat; bei meinen unfreiwilligen freiwilligen Testern Akin und Moritz; bei den TypeScript und Phaser Discord-Communties, die mich auch durch kryptischen Fehlermeldungen geführt haben; und natürlich bei meinen Freunden und meiner Familie, und Allen, die im Großen und Kleinen mitgewirkt haben — weil Dank leider keine Selbstverständlichkeit ist. Zu guter Letzt bedanke ich mich bei der Ente, die am Tag der Abgabe die ganze Nacht hindurch gequakt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungen

AR Adaptivity Response

AZaaLw Ausbildungszentrum für abbildende Aufklärung der Luftwaffe

DDA dynamische Schwierigkeitsanpassung

DGBL/GBL Digital Game-Based Learning

GBL Game-Based Learning

IOSB Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung

JS JavaScript

JSON JavaScript Object Notation

MUDs Multi User Dungeons

NPC Non-Player Character

PCG Prozedurale Generierung

PRNG Pseudo-Random Number Generators

SG Serious Game

TS TypeScript

UCD User-Centered Design

UGC User-generated Content

UI/Benutzerschnittstelle User Interface

UX User Experience

Kurze Zusammenfassung — Die dynamische Schwierigkeitsanpassung ist eine beliebete Methode, um Spiele adaptiv zu gestalten. Dabei wird eine für den Spieler relevante Schwierigkeit ermittelt und im Spiel abgebildet. Damit die Schwierigkeit im adaptiven Spiel abgebildet werden kann, muss das Spiel parametrisiert werden. Dabei bedeutet Parametrisierung das Finden und Einbinden von Mechanismen, mit denen das Spiel adaptiv beeinflußt werden kann. In dieser Arbeit wird ein Verfahren zur Parametrisierung adaptiver Spiele vorgestellt und am Beispiel eines adaptiven Wimmelbildspiels deomstriert.

Abstract — dynamische Schwierigkeitsanpassung is a popular method to adapt games. This requires the finding of a relevant difficulty level and applying it in the game. The process of mapping and applying the difficulty in the game is called parametrization. This work shows how to paramartize adaptive games by an example implementation of an adaptive *hidden object* game.

Schlüsselwörter/Keywords. Adaptive Systeme, DDA, Parametrisierung, DGBL, Game Design, Gamification Design, Adapitve Game Design, Bildauswertung, SG

Kapitel 1

Einleitung

Lernen ist vielleicht die einfachste und schwierigste Tätigkeit des Menschen zugleich. Viele Dinge können einfach, fast wie im Vorbeigehen gelernt werden — Herdplatten sind heiß, Wasser macht naß, Sandkuchen kann man nicht essen. Und dann wiederum gibt es Dinge, deren Erlernen Zeit und Mühe kosten— Algebra, Fremdsprachen oder Teilchenphysik.

Aktuell, zu Zeiten des globalen Corona-Lockdowns, wird wieder viel vom Lernen und Lehren gesprochen. Dabei geht es um das Öffnen und Schließen von Kitas und Schulen, die mangelnde und stockende Digitalisierung, die Überforderung von Schülern und Lehrern im Heimunterricht, und um das digitale Lernen.

Während Schulen Schüler schon seit mehereren tausend Jahren[1] „quälen"', ist digitales Lernen erst seit den letzten Dekaden ein Begriff; als Edutainment, Exergames, E-Learning oder als Serious Game (SG), Digital Game-Based Learning (DGBL oder GBL) oder Applied Gaming.

Dabei wird versucht Lernen und Software zu vereinen, um Schüler und Lerner jeglicher Couleur im neuen medialen Jahrtausend anzusprechen und zu begeistern. Besonderer Computerspiele gelten als modernes und zeitgenössisches Mittel, um Motivation, Engagement, und —im besten Fall— Lernerfolg zu steigern (Unterabschnitt 3.1.1).

Auch wird die Schar der Lernenden nicht als graue, frontal zu bespaßende Masse, sondern als Individuen mit eigenen Stärken, Schwächen, Bedürfnissen und Charakterzügen betrachtet[2], die entsprechend individuell gefördert werden müssen : Das Lernangebot soll sich dem Lerner anpassen, nicht umgekehrt!

Diese Art der Anpassung kann man auch als adaptives Lernen bezeichnen [83].

Grundvoraussetzung für adaptives Lernen ist Wissen über den Lerner — denn wie soll auf individuelle Eigenschaften eingegangen werden können, wenn diese unbekannt sind? Fehlt dieses Wissen, so muss es gesammelt werden. Auch ist Wissen nie fest, sondern verändert sich, besonders wenn es um die Eigenschaften

[1] Die ersten Schulen sollen von den Sumerern ca. 3–4.000 v.Chr. genutzt worden sein. [130]
[2] Eine Ansicht, die schon Quintilian im alten Rom teilte [121, S. 49f]

und Charakterzüge eines Menschen geht. Es muss also kontinuierlich überprüft und erweitert werden. Schließlich muss das Wissen angewandt werden: Wie geht es dem Lerner? Was ist sein Zustand vor dem Bildschirm? Braucht er Unterstützung? Hilfe? Wie soll geholfen werden? Soll das System eingreifen? Angepasst werden? — dem Lerner soll immer die richtige Hilfe zur richtigen Zeit zuteil werden.

Dieses Ökosystem aus Tracking, Profiling, Hilfe, Unterstützung und Anpassung bezeichnet man als adaptives System (Adaptivitätszyklus nach Shute und Zapata-Rivera, siehe Unterabschnitt 3.2.1).

Nun darf man sich ein solches adaptives System nicht als Monolith, als großes Ganzes vorstellen. Viel mehr ist es ein ineinander verschachteltes Zusammenspiel von Einzelteilen, in interoperable Module unterteilt — Einige Module beobachten den Lerner und sammeln Wissen, andere verwalten das Wissen, prüfen und verbessern es, und wieder andere wenden das Wissen an und interagieren mit dem Lerner.

Ein Beispiel für mit dem Lerner interagierende Module sind Lernspiele.

1.1 Problemstellung

Im Bereich der Lernspiele (DGBL, siehe Unterabschnitt 3.1.1 und Unterabschnitt 3.2.3) hat sich die DDA[3] als beliebte Form der Anpassung hervorgetan. Dabei wird versucht den Spieler möglichst lange in einem Vertiefungszustand (Unterabschnitt 3.1.2) zu halten, indem die Schwierigkeit des Spiels gemäß den Fertigkeiten des Lerners zwischen Langeweile und Überforderung balanciert wird. Sprich: Das Spiel darf nicht zu leicht oder zu schwierig sein, sonst gibt der Lerner gelangweilt oder frustriert auf.

Dazu muss zum einen die Leistung des Lerners abgefragt und eine Schwierigkeit ermittelt (systemseitig), und zum anderen die ermittelte Schwierigkeit tatsächlich im Spiel abgebildet werden (lokal). Streicher et al. [115, S. 23f] bezeichnen dies als *When, what and how to adapt.*

Aber wie bildet man Schwierigkeit ab, wenn sie nicht statisch vordefiniert ist, sondern sich fließend an den Bedürfnissen des Lerners orientiert? Wie muß das Spiel aufgebaut sein, damit es tatsächlich an den Lerner angepasst werden kann? Es folgt: Nicht nur die Schwierigkeit, sondern auch das Lernspiel muss anpassbar sein! Somit ergibt sich ein Bedürfnis nach anpassbaren Lernspielen — und die Frage:

> *Wie parametrisiert man ein adaptives Lernspiel?*

Dabei entspricht die „Parametrisierung“ dem *what* — wie muss das Spiel erstellt werden, so dass es angepasst werden kann, welche Mechaniken eignen sich für eine adaptive Anpassung, wie läßt sich Schwierigkeit parametrisieren? Dies kann man als die Errichtung von *Hebeln* für die Schwierigkeitsanpassung zu sehen [134].

[3] DDA = Dynamic Difficulty Adjustment (eng.)

Die Beantwortung dieser Frage ist auch im Zusammenhang mit *„[the lack of] real application examples, [...] high costs for design, authoring, and the technical implementation of both the games (including the serious game elements) and the adaptation system (the adaptivity) [and] the necessity for reliable and easily transferable concepts to effectively and efficiently create serious games with adaptive components“* [115, S. 39] bedeutsam — die Parametrisierung verbindet adaptives System und Spiel, und ist somit als eine Schlüsselkomponente bei der Erstellung von adaptiven Spielen zu betrachten.

Als Mittel zum Zweck kommt die Prozedurale Generierung (PCG)[4] in Frage. Mit ihr können Spielinhalte zur Laufzeit algorithmisch generiert werden, und müssen nicht statisch vordefiniert sein. Interessant ist dabei die Tatsache, dass sich die algorithmische Generierung parametrisieren läßt [34, S. 2] — es können also nach Bedarf und ohne Mehraufwand oder Mehrkosten[5] beliebig viele Variationen der benötigten Spielinhalte erstellt werden. Ändert sich die Spielschwierigkeit, wird eine entsprechende neue Spielvariante erzeugt. Die Herausforderung ist dabei die Steuerbarkeit des PCG-Prozesses [67, S. 8].

Zur Beantwortung der Frage wird prototypisch ein adaptives/adaptierbares Wimmelbildspiel[6] erstellt. Dieses soll als *Testbed* eingesetzt werden, um die DDA zu erforschen [109], da das Anpassen der Spielparameter mühselig und zeitaufwändig ist [76, S. 1].

Die Spielidee dazu stammt von einer Forschungsgruppe[7] am Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung (IOSB), die sich (u.a.) mit E-Learning, DGBL, adaptiven Systemen und DDA beschäftigt.

Anwendungskontext des Spiels ist –der Name des Instituts verrät es– die Bildauswertung, genauer gesagt die Typenkunde (Abschnitt 3.5).

1.2 Vorgehen & Aufbau

Zuerst wird ein Überblick über das Themengebiet gegeben, verwandte Arbeiten vorgestellt Kapitel 2 und notwendiges Grundlagenwissen – *Was ist ein Spiel? Was ist ein adaptives System? Was ist DDA? Was hat Gamification mit adaptiven Systemen und DGBL zu tun? Was ist PCG Was sind Bildauswertung und Wimmelbildspiele?*– definiert (Kapitel 3).

Anschließend –im Hauptteil der Arbeit– wird die Methodik zur Beantwortung der Forschungsfrage vorgestellt (Kapitel 4). Dazu wird der Begriff „adaptives Spiel“ in den Kontext der Parametrisierung gestellt und ein Verfahren zur Erstellung und Parametrisierung von adaptiven Spielen gezeigt. Das gezeigte Verfahren wird in den nachfolgenden Kapiteln umgesetzt: Dazu wird ein adaptives Spiel konzipiert (Kapitel 5), implementiert (Kapitel 6) und evaluiert (Kapitel 7).

[4] PCG = Procedural Content Generation (eng.)
[5] Computer bekommen kein Gehalt.
[6] „Hidden Object“-Spiel
[7] Forschungsgruppe „WiMo“ (Wissensmodelle)

Schließlich wird das gezeigte Verfahren diskutiert (Kapitel 8) und ein Fazit gebildet (Kapitel 9).

1.3 Abgrenzung & Einschränkungen

Diese Arbeit beschäftigt sich mit der Parametrisierung eines adaptiven Lernspiels. Dabei liegt der Fokus auf der Parametrisierung, und nicht auf dem Lernspiel.

Es findet keine Evaluierung hinsichtlich Lernerfolg und Didaktik statt. Auch die Ausbildung angehender Bildauswerter und die Typenkunde sind nur im Rahmen der Konzeption interessant.

Das Spiel ist kein Simulator für reale Luft- und Satellitenbilder, die im Spiel verwendteten Inhalte und Modelle sind prototypischer Natur[8].

Die Erforschung von DDA und Adaptivität ist nicht Teil der Arbeit — die Arbeit beschäftigt sich mit der Frage welche „Stellschrauben“ ein adaptives Spiel besitzen sollte, und wie diese zu Erstellen sind (*what to adapt*), nicht aber, wann und wie diese „Stellschrauben“ eingestellt werden müssen (*When and how to adapt*).

[8] „Dummybilder“

Kapitel 2

Verwandte Arbeiten

Dieses Kapitel zeigt verwandte Arbeiten, um einen Überblick über das Anwendungsgebiet zu geben, und mögliche Lösungen und Lösungsansätze im Themenkomplex aufzuzeigen.

Streicher et al. [115] benennen das Problem der Parametrisierung von adaptiven und adaptierbaren Spielen, verweisen aber nur auf das *Game Design*. Auch stellen sie ein adaptives Lernspiel für die Bildauswertung vor (Lost Earth 2307), geben aber keinen Einblick in die Implementierung und Parametrisierung des Spiels. Yannanakis et al. [136][9] nennen *game parameterization* ebenfalls als wichtigen Bestandteil ihres Konzept für ein Lernspiel zur Konfliktresolution, gehen aber auch nicht auf die Parametrisierung ein.

Lopes und Bidarra [67] geben einen Überblick über *targets of adaption*, also Spielelemente, die parametrisierbar sind.

Die Erstellung eines adaptiven Spiels kann man auch als *Adaptive Game Design* bezeichnen. In diesem Zusammenhang betrachten Charles et al. [24] die Verwendung von Spielermodellen und -Typen, während Lindely und Sennerstein [66] die Verwendung von Schema- und Spieleranalyse zur Anpassung von adaptiven Spielen und Spielmechaniken empfehlen. Wu und Lin [133] zeigen einen Einfluss der menschlichen Physiologie auf das *adaptive game design* auf, und Tijs et al. [119] stellen vor, wie ein *emotionally adapitve game* erstellt werden kann. Schließlich wird von Gilleade und Dix [45] eine Eignung von *Frustration* im Kontext von *adaptive game design* gesehen.

Im Kontext des Wimmelbildspiels und der Bildauswertung zeigt sich, dass es schon seit längerer Zeit im Fokus des IOSB liegt. Beispielsweise finden sich Erwähnungen in [117], [114] oder in [113]. In letzterem läßt sich auch die Schwierigkeit manuell einstellen (Adaptivity Response (AR)). [112] beschäftigt sich mit der Frage *„when an adaptive serious game needs to adap“* im Kontext eines 2.5D *seek-and-find-games* für die Bildauswertung.

Abseits der adaptiven Spiele beschäftigen sich Streicher und Lehmann [111] und [63] mit der Erstellung eines kartenbasierten Lernspiels für die Bildauswertung

und sprechen dabei auch die Erzeugung von Schwierigkeit im Spiel an.

Im Kontext adaptiver Spiele erstellt Biegemeier [10] eine Schnittstelle, mit der Nutzungsdaten analysiert und Einfluß auf das Spiel genommen werden kann.

Die Parametrisierung adaptiver Spiele wird meist in Verbindung der PCG erwähnt. [74] implementieren und parametrisieren einen prozeduralen Platformer/Sidescroller auf Basis von *Rhythm-Group Therapy* und *Brain Computer Interfaces*. Die prozedruale Generierung eines adaptiven Sidescrollers/Platformers ist ebenfalls für Jennings-Teats et al. [54] interessant.

Jenseits der DDA beschäftigen sich Van Lankfeld et al. [124] mit der *Incongruity Theory* zur Adjustierung eines Spiel, während Pfau et al. [87] Schwierigkeit durch Verwendung von *Deep Player Behavior Models* parametrisieren.

Zook et al. [138] beschäftigen sich mit der prozeduralen Generierung eines adaptiven Spiels unter Verwendung von *Player Models*. Lopes, Eisemann und Bidarra [68, S. 7] zeigen *„how game designers can control adaptive game world generation“*. Dies basiert auf einem Mix aus *adaption rules* und *retrieval algorithms*. Frommel et al. [41] passen die Schwierigkeit eines *2D vertical scrolling platform browser game* durch *Emotion-based Difficulty Adjustment* (EDDA) an. Sie basieren ihre Spielekonzeption auf Rational Level Design (RLD).

Kapitel 3

Grundlagen

Um alle Leser auf den selben Wissenstand zu heben, und um Missverständnisse zu vermeiden, wird in diesem Kapitel eine kurze Vorstellung notwendiger Grundlagen gegeben.

3.1 Spiele

Diese Arbeit beschäftigt sich mit Spielen, genauer gesagt adaptiven Lernspielen. Da überrascht es, dass es keine eindeutige Defintion von Spielen gibt, obwohl Spiele schon seit mehreren tausend Jahre gespielt werden [99, S. 35][98, Kap. 7].

Roger Caillois [22, S. 9ff] definiert *Play* als

- *Free.* Nicht verpflichtend.
- *Separate.* Zeitlich und räumlich begrenzt.
- *Uncertain.* Ohne sicheres Ergebnis, ungewiss.
- *Unproductive.* Die Situation des Spielers nach dem Spiel entspricht der Situation des Spielers vor dem Spiel.
- *Governed by Rules.* An Regeln gebunden.
- *Make-believe.* Findet fiktiv in einer „Spiel-Welt“ statt.

und unterscheidet zwischen verschiedenen Arten von Spielen (*Agon*: Wettkampf, Herausforderung. *Alea*: Glücksspiel, Zufall. *Mimicry*: Rollenspiel, Verkleidung. *Ilinx*: Schwindel, Rausch, Unordnung.) Diese unterteilt er weiter in *paidia* und *ludus*: *Paidia* ist unbeschwert und *capricious*, während *ludus* das Gegenteil und mit *tedious convention* überladen ist.

Salen und Zichermann bezeichnen ein Spiel als: *„a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.“* [98, Kap. 7, S. 11]

Jane McGonigal ([72, S. 21]) fasst die vorherigen Definitionen zusammen: *„When you strip away the genre differences and the technological complexities, all games share four defining traits: a goal, rules, a feedback system, and voluntary participation.“*:

- *Goal.* Das, was der Spieler im Spiel erreichen möchte, und worauf er dementsprechend hinarbeitet.
- *Rules.* Die Limitierungen, die den Spieler vom einfachen Erreichen des Ziels abhalten.
- *Feedback system.* Rückmeldung wie nah der Spieler am Erreichen des Ziels ist.
- *Voluntary participation.* Der/die Spieler können sich jederzeit aus dem Spiel zurückziehen.

Abseits der „akademischen“ Definitionen bezeichnet Schell ([99, S. 47]) ein Spiel als *„A game is a problem-solving activity, approached with a playful attitude.“* Um aber auf dem wissenschaftlichem Pfad zu bleiben, folgt diese Arbeit der Definition von McGonigal.

Hierbei sei angemerkt, dass Spiele in dieser Arbeit immer Computerspiele bezeichnen, auch wenn es auch Nicht-Computerspiele gibt.

3.1.1 Spiele und Lernen

Im Kontext dieser Arbeit sind Spiele interessant, da sie sich zum Lernen eignen [26][32][103][81][80]. Sie sind „arrangierte Lernumgebungen“, obwohl sie meist nicht als klassische „Lern-Lehr-Umgebung“ aufgefast werden [13, S. 1]. Koster bezeichnet Spiele auch als *teachers* [61, S. 46]. Sie haben die Eigenschaft –ähnlich guter Bücher [33]– den Spieler in ihren Bann ziehen, ein Zustand, der als Immersion bezeichnet wird [99, S. 138f][13, S. 1].

In diesem Zusammenhang spricht Bopp ([13, S. 1]) von einer *Immersiver Didaktik*: *„Computerspiele sind als arrangierte Lernumgebungen darauf angelegt, eine Bewusstwerdung ihres didaktischen Designs [...] zu vermeiden, um das Eintauchen ins Spielgeschehen, das zeitweise Vergessen des Selbst und der der Spielumgebung, die sogenannte Immersion bzw. den Spiel-Flow nicht zu gefährden.“*

Zu einem ähnlichen Schluss kommt auch Marc Prensky [93, S. 2]. Er beschäftigt sich mit Motivation und Lernen, und kommt zu dem Schluss, dass Spiele das ideale Vehikel für Motivation sind: *„People play games because the process of game playing is engaging. In fact, the top two reasons people say they play interactive games [...] is because they are challenging and relaxing. This formulation seems very close to that magical state of motivation some refer to as “flow.“*. Dabei dient der Flow-Zustand als Vehikel, um *„to facilitate positive user experience in order to maximize the impact of educational games“* [58].

Aus diesem Grund schlägt Prensky vor, Spiele und Lernen zu verbinden — eine Verbindung, die er DGBL nennt [92][101].

Plass et al. [90, S. 260] nennen vier Gründe für die Verbindung von Spielen und Lernen:

- *Motivation.* Spiele motivieren und sorgen dafür, dass „Lernen Spaß macht“.
- *Player Engagement.* Spiele können auf viele verschiedene Weisen gespielt werden und bringen den Spieler dazu, sich mit dem Lernmaterial in unbekannter Form ausienanderzusetzen.
- *Adaptivity.* Spiele können an den Spieler angepasst werden und so auf verschiedenste Einschränkungen Rücksich nehmen.
- *Graceful Failure.* Der Spieler kann scheitern, ohne, dass es Auswirkungen auf die Realität hat.

Das Phänomen von DGBL tritt unter vielen Namen auf. So werden Spiele, die sich zum Lernen eignen, auch als Lernspiele oder SG bezeichnet. In dieser Arbeit werden diese Begriff aber kritisch gesehen, da es keinen Grund gibt, zwischen Spielen und Lernspielen zu unterscheiden: Unabhängig der transportierten Lerninhalte ist ein Spiel ein Spiel. Dementsprechend ist ein Lernspiel „nur“ ein Spiel, dass in einem Lernkontext eingesetzt wird.

Die von Bopp und Prensky genannten Immersion und Flow werden in dieser Arbeit als Vertiefungszustände zusammengefasst.

3.1.2 Vertiefungszustände

Cairns et al. [23, S. 7f] unterscheiden zwischen Flow und Immersion, sehen aber auch eine große Ähnlichkeit.

Immersion

Immersion in Spielen meint *„the engagement or involvement a person feels as a result of playing a digital game“* [23, S. 2].

Es gibt drei Stufen von Immersion [23, S. 2]:

- *Engagement.* Der Spieler investiert Zeit und Anstrengung um ein Spiel zu spielen.
- *Engrossment.* Der Spieler schenkt dem Spiel viel Aufmerksamkeit und ist auch emotional im Spiel involviert.
- *Total immersion.* Der Spieler denkt an nichts anderes mehr und „feels in the game“.

Die dritte Stufen *Total Immersion* kann mit *Flow* gleichgesetzt werden, wobei *Total Immersion* aber als kurzweilig beschrieben wird, während *Flow* über einen längeren Zeitraum andauern kann [23, S. 8].

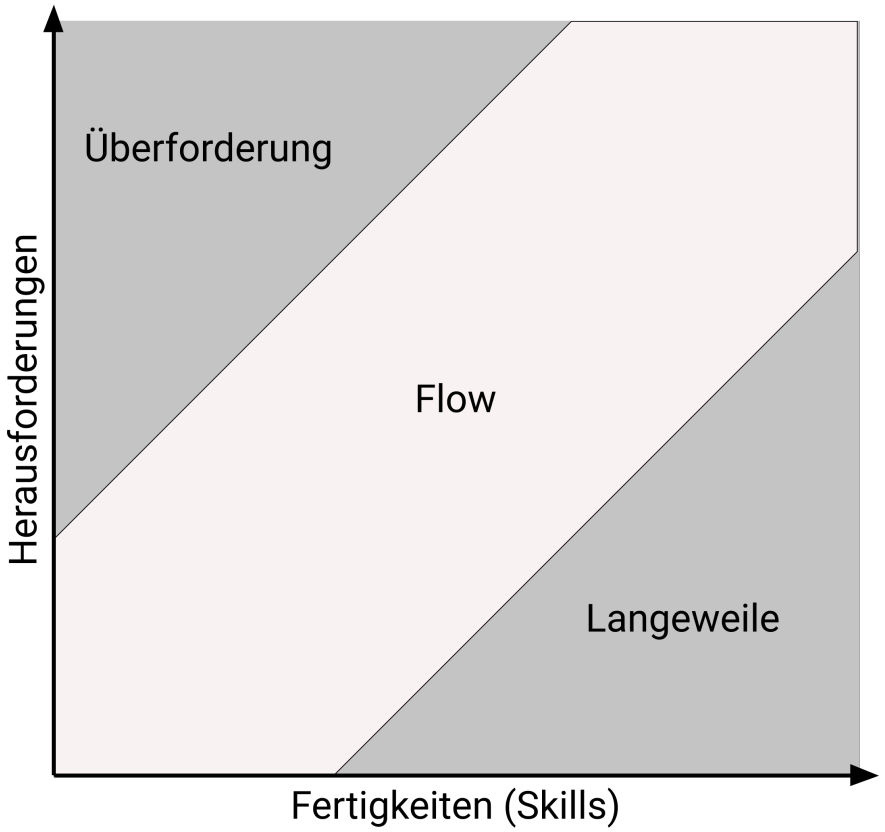


Abbildung 3.1: Flow-Diagramm

Flow

Flow beschreibt einen *„ automatic, effortless, yet highly focused state of consciousness“* der von Mihalyi Csikszentmihalyi [27, S. 111ff] während der Beobachtung von Schachspielern, Felskletterern, Tänzern und Komponisten beobachtet wurde. Das Flow-Erlebnis besteht aus neun Elementen:

1. *There are clear goals every step of the way.* Der Mensch weiß genau, was er als nächstes tun muss.
2. *There is immediate feedback to one's actions.* Der Mensch weiß genau, ob er richtig oder falsch mit seiner Tätigkeit liegt.
3. *There is a balance between challenges and skills.* Die Tätigkeit ist nicht zu leicht oder zu schwer. Sie ist genau angemessen.
4. *Action and awareness are merged.* Der Mensch ist auf die Tätigkeit fokussiert.
5. *Distractions are excluded from consciousness.* Der Mensch geht keinen falschen Gedanken nach.
6. *There is no worry of failure.* Der Mensch sorgt sich nicht um die Zukunft und das Ergebnis.
7. *Self-consciousness disappears.* Der Mensch vergisst sich selbst. Nach Beendi-

gung des Flow-Erlebnisses kann ein „merkwürdiges“ Gefühl der Selbstwahrnehmung eintreten.

8. *The sense of time becomes distorted.* Der Mensch vergisst die Zeit. Er kümmert sich nicht darum, wie lange seine Tätigkeit schon andauert oder andauern wird.

9. *The activity becomes autotelic.* Die Tätigkeit wird zum Selbstzweck.

Flow tritt nur ein, wenn die Tätigkeit die richtige Balance zwischen Langeweile und Überforderung trifft, also im *Flow-channel* ist (Abbildung 3.1). Wird diese Balance gebrochen, so endet das Flow-Erlebnis.

Eine Art, wie Flow im Spiel erzeugt und gehalten werden kann, sind adaptive Systeme.

3.2 Adaptive Systeme

Alle Menschen sind unterschiedlich, und Lernen dementsprechend auch verschieden [104][2]. Diese Beobachtung ist nicht neu ([91, S. 275]), wird aber durch die digitale Technologie gefördert. Besonders adaptive Systeme sind eine Möglichkeit, wie individuelles Lernen umgesetzt werden kann, und können als *user-centered* oder *player-centered* Ansatz gesehen werden. Dabei überwacht das adaptive System den Benutzer und passt es seinen Bedürfnissen an [105, S. 1]:

> Adaptive educational systems monitor important learner characteristics and make appropriate adjustments to the instructional milieu to support and enhance learning. The goal of adaptive educational systems [...] is to create an instructionally sound and flexible environment that supports learning for students with a range of abilities, disabilities, interests, backgrounds, and other characteristics.

In dieser Arbeit werden adaptive Systeme entsprechend der oben gegebenen Definition im Nutzungskontext zur Unterstützung von Lernprozessen gesehen. In diesem Kontext beschreibt Adaptivität *„changes to the environment to accommodate the needs of the learner“* [91, S. 276]

Adaptive System sind vielseitig einsetzbar und eignen sich zur Verbesserung von *„motivated usage, user acceptance, and user identification within and outside of the games.“*[115, S. 1]. Sie verbessern *„efficiency, effectiveness, and enjoyment of the learning experience.“* [104, S. 113].

Die Erstellung adaptiver Systeme ist multidisziplinär, sie berührt beispielsweise *„game design, software engineering, cognitive science, pedagogy, evaluation methodologies, etc.“* [115, S. 23]

Die Verwirklichungsmöglichkeiten adaptive Systeme beruhen auf der Macht modernern Technologie [104, S. 108], beispielsweise künstlicher Intelligenz, *Machine Learning*, User-Tracking und *Big Data* [115, S. 2] oder *cognitive modelling* [60].

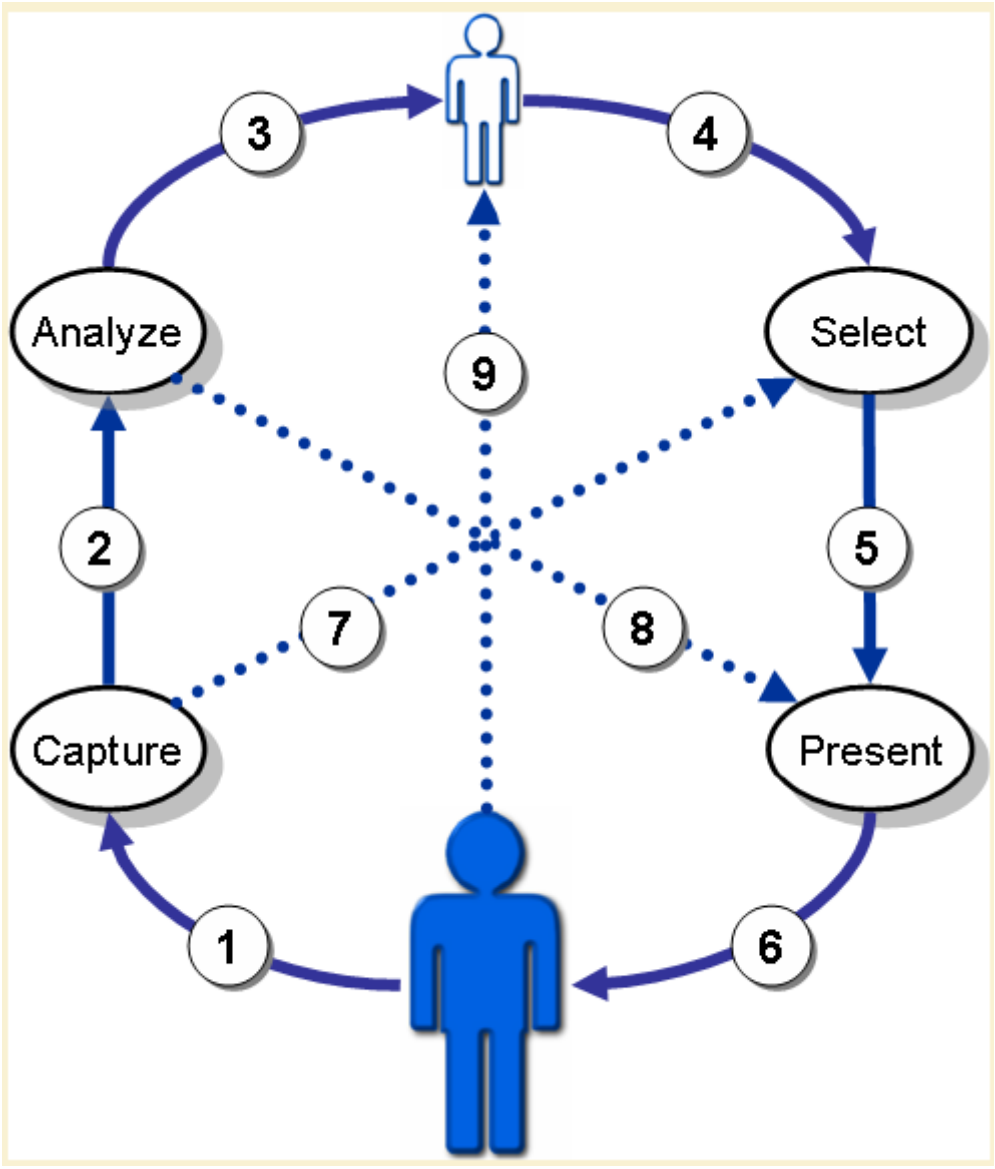


Abbildung 3.2: Adaptivitätszyklus von Shute und Zapata-Rivera [105]

Das adaptive System kann durch den Adaptivitätszyklus von Shute und Zapata-Rivera [105] beschrieben werden.

3.2.1 Adaptivitätszyklus

Der Adaptivitätszyklus von Shute und Zapata-Rivera [105] (Abbildung 3.2) besteht aus vier Phasen bzw. Prozessen:

- *Capture*. Ein Lerner (blaue Figur) interagiert mit dem System (1). Alle relevanten kognitive (Eingaben, Fehler, Lösungen, ...) und nicht-kognitive (Engagement, Flow, ...) Daten der Interaktion werden erfasst und dem System weitergegeben (2).
- *Analyze*. Die Daten der Interaktion werden analysiert und benutzt, um ein

Lernermodell (kleine weiße Figur) zu erzeugen oder zu verbessern (3). Das Lernermodell beschreibt das Wissen des Lerners zur jeweiligen Lerndomäne.

- *Select.* Basierend auf Lernermodell, Lerndomäne und Lernziel werden neue für den Lerner relevante Informationen und Lernstrategien ausgewählt (4).
- *Present.* Die gewählten Informationen werden aufbereitet, in eine lernbare Form gebracht (5) und dem Lerner vorgelegt (6), auf dass der Zyklus von neuem beginnen kann.

Das System ist flexibel, verschiedene Formen der Adaption können angewandt werden (7, 8, 9). Diese sind für diese Arbeit aber ohne Belang.

3.2.2 Die drei W-Fragen

Es gibt drei Schwierigkeiten beim Erstellen von adaptiven Sytemen. Streicher et al. [115, S. 23f] bezeichnen sie als *When, what and how to adapt*, während Eleven et al.[2, S. 8] und Lopes und Bidarra [67] nur zwischen *How and when* unterscheiden:

- *When.* Wann soll das System eingreifen? Wie muss sich das Verhalten des Benutzers geändert haben, damit das System ein Eingreifen authorisiert? [91, S. 277f] nennen verschiedene kognitive, motivierende, affektive und soziokulturelle Variablen, an denen sich das System orientieren kann. Hierbei sollte auch auf Spielertypen achtgegeben werden [8].
- *What.* Was soll das System verändern? Welche Medien, Informationen oder Spielinhalte sollen angepasst werden?
- *How.* In welchem Ausmaß soll die Veränderung durchgeführt werden?

Auf den Adaptivitätszyklus angewandt, beschäftigt sich *When* mit *Analyze*, *What* mit *Select* und *Present*, und *How* mit der *Present*-Phase.

3.2.3 Adaptive Systeme und Spiele

Besonders interessant sind Adaptive Systeme in Verbindung von Spielen, da Spiele sich für das Lernen eignen (DGBL) und intrinsisch motivierend sind. Auch können sie *„[help] players avoid getting stuck, [adapt] the gameplay more to the player's preference/taste, or [detect] deviant player behaviour“* [91, S. 1] und *„make the game experience more unique and personal“* [67, S. 1].

Spiele können adaptiv und/oder adaptierbar sein. Adaptierbare Spiele werden vom Benutzer angepasst. Adaptive Spiele können automatisch angepasst werden [115, S. 13ff]. Es ist möglich, dass ein Spiel sowohl adaptiv als auch adaptierbar ist.

Hier fällt eine Doppelung im Adaptivitätszyklus (Abbildung 3.2) auf: In diesem wurden während der *Select*-Phase (4) für den Lerner relevante Informationen und Lernstrategien gewählt. Diese Informationen können in Form von Text und Video, aber auch als Spiel vermittelt werden. Das Spiel wiederum kann ebenfalls adaptiv sein, und einen eigenen Adaptivitätszyklus bilden.

Dabei gilt: Ein adaptives Spiel ist immer ein adaptives System, ein adaptives System jedoch nicht immer ein Spiel.

In Kontext von Spielen beschreibt Adaptivität *„the automatic adaptation of game elements, i.e., of content, user interfaces, game mechanics, game difficulty, etc., to customize or personalize the interactive experience.“* [115, S. 1]

Die dynamische Anpassung der Spielschwierigkeit bezeichnet man als DDA. Diese Anpassung ist aus der Not geboren, dass die in Spielen angebotenen Schwierigkeiten (leicht, mittel, schwer) oft nicht ausreichen, um die Anforderungen der Spieler an das Spiel abzudecken[70][67, S. 1].

Ein bekanntes Beispiel für DDA ist das Spiel *Mario Kart*, bei dem ein weit abgeschlagener Spieler einen Geschwindigkeitsbonus bekommt und bessere Gegenstände findet, um den Abstand zu den anderen Spielern zu verringern.

Bei der DDA wird die Schwierigkeit, genauer gesagt die *task difficulty* angepasst. Sie bezeichnet *„the degree to which the activity represents a personally demanding situation requiring a considerable amount of cognitive or physical effort in order to develop the learner's knowledge and skill levels“* [82, S. 2417]

Im adaptiven System wird die angepasste Schwierigkeit mittels einer AR mitgeteilt.

3.2.4 Adaptivity Response

Die *AR* drückt die Schwierigkeit in wenigen skalaren, normativen Parametern aus [108, S.2f]. *Response* bezeichnet hierbei die Antwort des Systems an das Spiel, also den Pfeil „zurück“ — im Adaptivitätszyklus (Unterabschnitt 3.2.1) der Übergang (5) von „Select“ zu „Present“.

Die AR kann als Formel ausgedrückt werden:

$$AR(Context\ C, Time\ t) = (P^C(t), A^C(t), S^C(t))$$

Die AR zum Zeitpunk t im Anwendungskontext C besteht aus drei Werten: P, A und S.

- *Performance P.* Bewertet die aktuelle Performance des Nutzers. Beispiele: Der Fortschritt im Level, die aktuelle Schwierigkeit, Punktestand, ...
- *Assistance A.* Ob und wann dem Nutzer geholfen werden soll, bzw. ob ihm gerade Unterstützung zuteil wird. Beispiele: Lösungsvorschläge, „Joker“, Hinweise, ...
- *Skill S.* Die Langzeitperformance des Nutzers. Wie gut er (wiederholt) in Übungsaufgaben, Quizzen, Testaten, Prüfungen,... abgeschlossen hat.

Dabei ordnen P^C, A^C und S^C jedem Zeitpunkt t Werte in [0, 1] zu.

Vorteil der AR ist, dass sie durch die generische Parameter eine hohe Interoperabilität erreicht, und gleichzeitig durch Zusatzinformationen kontextsensitiv gestaltet werden kann [108, S.2]. Auch eignet sich die AR zum simulieren und testen von

adaptiven Systemen [109] — ein großer Vorteil, da adaptive Systeme sehr komplex sind, und AI-getroffene Entscheidungen für den Menschen oft nur schwierig nachvollziehbar sind.

3.3 Gamification

So wie eine genaue Einordnung des Begriffs „Spiel“ schwierig ist (Abschnitt 3.1), gibt es auch eine Vielzahl an unterschiedlichen Definitionen von Gamification, über die kein Konsens herrscht [6, S. 1][4, S. 4].

Die erste (und bekannteste) Definition von Gamification –*the use of game design elements in non-game contexts*– erfolgte durch Deterding et al. 2011 [30], um das neue Phänomen von „mass-market consumer software that takes inspiration from video games“ einzuordnen, dass sich zum damaligen Stand auf dem Höhepunkt[1] des „Gartner Hype Cycles“ befand [38]. Dabei verrät die Verwendung des Wortes „consumer software“, dass die Ursprünge von Gamification im *Business*-Bereich einzuordnen sind.

Ebenfalls aus dem Bereich *Business* (Service Marketing) kommend, geben Huotari & Hamari 2012 eine Definition von Gamification als *„a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation“* [52][51].

Gamification läßt sich aber auch jenseits des *Business*-Kontexts einsetzen, weshalb neuere Definitionen auf eine explizite Nennung verzichten — auch fällt *Business* unter den Bereich *non-game context* und ist somit bereits abgedeckt [4, S. 4].

Weiterhin lassen sich die beiden Defintionen zusammenfassen: Baptista & Oliveira (2019) *„acknowledge gamification as the use of game-design elements in non-gaming contexts ([30]), in a process of enhancing a service with game-related features that support users' overall value creation ([51])“* [6, S. 1].

Morschheuser et al (2017, 2018) hinwiederum schränken Gamification als *„enhancement of information technology via design features borrowed from (video) games“* [78, S. 1298], beziehungsweise als *„enrichment of software with design features known from games in order to invoke similarly engaging experiences as games do“* [77, S. 220] auf einen digitalen Rahmen ein.

Dabei fällt auf, dass der Anwendungskontext von Gamification zuvor frei und nicht auf den digitalen Rahmen begrenzt wurde — auch eine Rabatt-Aktionen im Supermarkt (z.B. *Treuepunkte* beim Einkauf) ist angewandte Gamification.

Ebenso ist das Streichen des Zauns durch Tom Sawyer [123, Kap. 2] ein Beispiel für Gamificaton: Ein *non-gaming context* (Der Zaun) wird mit *game-design elements and related features* (Belohnungen, Ränge, ein Tauschsystem, Zuschauer- und Kommentarfunktionen) erweitert, um das Erstellen von *value* (Das Auftragen von Farbe) zu unterstützen.

[1] *peak of inflated expectations*

Dabei läßt sich auch Verwandschaft von Gamification und Game-Based Learning (GBL) feststellen [4, S. 4] — oder um es mit den Worten von Mark Twain auszudrücken [123, Kap. 2]:

> If he had been a great and wise philosopher, like the writer of this book, he would now have comprehended that Work consists of whatever a body is obliged to do, and that Play consists of whatever a body is not obliged to do.

„The key is to figure out how to turn Work into Play.“ [11, S. 1], eine Forderung die auch Marc Prensky teilt: *„The huge wall which has separated learning and fun, work and play for the last few hundred years is finally beginning to tremble and will soon come tumbling down, to everyone's benefit.“*[92, S. 4]

Der Unterschied zwischen Gamification und GBL liegt im Anwendungskontext: GBL umfasst Spiele, während Gamification „Nicht-Spiele“ umfasst. Da Spiele aber schwierig zu definieren sind (Abschnitt 3.1), läßt Gamification die Grenze zwischen Spielen und Nicht-Spielen verschwimmen [30, S. 3]: *„A gamified intervention may not operate as a full game experience but contains gaming elements, such as the scoring of points, in-game rewards, or engaging in quests.“* [40, S. 2]

Auf den Anwendungskontext der Arbeit bezogen, bedeutet das: *„Gamification is not when learning is changed into a computer game but rather when adding a design layer of game elements to enhance learning, increase engagement, and encourage positive behavior“* [4, S. 7]. In diesem Sinne ist Gamification parasitär; jeder Nicht-Spiel Kontxt kann um eine Gamification-Ebene erweitert werden.

Daraus ergibt sich eine eine „Schiff des Theseus“-Frage: Wann ist ein Spiel ein Spiel? Kann ein Nicht-Spiel durch Zufügen von Spielelementen in ein Spiel bzw. einen „Spiel-ähnlichen Zustand“ verwandelt werden? Und umgekehrt: Welche Spieleigenschaften müssen von einem Spiel entfernen werden, so dass es zu einem Nicht-Spiel wird? Und kann man ein Spiel gamifizieren [48]?

Diese Fragen liegen allerdings außerhalb des Rahmens dieser Arbeit. Auf den Anwendungskontext dieser Arbeit bedacht bezeichnet Gamification *„the application of game features, mainly video game elements, into non-game context for the purpose of promoting motivation and engagement in learning“* [4, S. 1].

Eine besondere Bedeutung wird der Gamification im Zusammenspiel mit adaptiven Systemen verliehen: Zum einen versucht Gamification –als Gegenstück zu DGBL– ebenfalls Motivation, Flow und Engagement zu beeinflussen, und ist somit für adaptive Systeme relevant. Zum anderen kann ein adaptives Spiel (Unterabschnitt 3.2.3) als Anhängsel an ein größeres adaptives System gamifiziert sein — dies bedeutet, dass ein Spiel nicht nur unter Gesichtspunkten des *Game Design*, sondern auch unter Gesichtspunkten des *Gamification Design* gesehen werden sollte.

3.4 Spielertypen

Alle Menschen sind unterschiedlich. Sie haben verschiedene Charakterzüge, Vorlieben, Stärken, Schwächen, Abneigungen, uvm.

Oftmals ist es notwendig, sie in Gruppen einzuteilen, z.B. um eine Marktsegmentierung zu erforschen [122], geeignete Werbeträger zu finden, Covid-Risikogruppen präferiert zu schützen, oder leistungsschwache Schüler zu fördern (Lerntypen).

Es gibt viele Möglichkeiten der Unterteilung, beispielsweise demographisch auf die Bevölkerung bezogen (Alter, Geschlecht, Herkunft, Einkommen), oder psychographisch (Persönlichkeitspsychologisch) [99, Kap. 9].

Besonders im Bereich der Psychographie sind viele Möglichkeiten zur Einordnung auf Grund der verschiedenen psychologischen Tests wie Myers-Briggs (MBTI) [14], Gardner's *Theory of Multiple Inzelligences* [44], Keirsey Temperament Sorter (KTS) [55], ... gegeben. So verwenden Nacke et al. [79] den MBTI als Grundlage für ihr BrainHex-Modell.

Im Bereich der Spielentwicklung (Game Design [99][20], Gamification [120]) wird oft auf Spielertypologien zurückgegriffen, um Spiele spielerzentriert und zielgruppengerecht zu entwickeln (User-Centered Design (UCD)), d.h. Bausteine zu finden, die die gewählte Zielgruppe besonders ansprechen, oder um neue Spieler anzuwerben. Auch eignen sie sich für das In-Game Marketing, können also helfen, im Spiel Spielinhalte zu bewerben [47][2]. Weiterhin kann man sie für das *Player-Tracking* im Sinne des adaptiven Systems sehen — sind Lerntyp und Spielertyp bekannt, ist es einfacher, ein Lernspiel mit für den Lerner motivierenden Inhalten und Mechaniken zu finden. Auch Alsawaier (2018) [4, S. 22] sieht einen Nutzen von Spielertypen für DGBL: „*The research about players' types inform us about a proper gamified design for delivering pedagogical content*"

Die bekannteste (weil erste) Spielertypologie stammt von Richard Bartle [7]. Er beschäftigte sich mit Multi User Dungeons (MUDs) und beobachtete, dass es vier Arten von Spielern gibt, namentlich *Killers*, *Socialisers*, *Explorers* und *Achievers*. Diese vier Spielertypen lassen sich auf einem zweiachsigen Graphen einordnen (Abbildung 3.3). Eine Achse unterscheidet zwischen den Spieler-zentrierten und den Welt-zentrierten Spielern, und die andere zwischen *acting* und *interaction*.

Faszinierend an Bartle's Spielertypen ist weniger der Bezug zu MUDs, als viel mehr die Tatsache, dass diese Spielertypen immer noch Bestand haben [137]. Während Bartle's Spielertypen gerne kritisiert oder erweitert werden, bleiben sie im Kern gleich[122, S. 7].

Ein oft missverstandener Punkt der Kritik ist die Tatsache, dass Bartle's Spielertypen nicht singulär sind, sondern „*players will often drift between all four, depending on their mood or current playing style*" [7, S. 3]. Ein Spieler bewegt sich also auf den gezeigten Achsen fort, sie sind linear.

Eine Neuaflage von Bartle's Spielertypen stammt von Amy Jo Kim [59], die diese

[2] Hierbei wird eine Verbindung zwischen Gamification und Marketing gut ersichtlich.

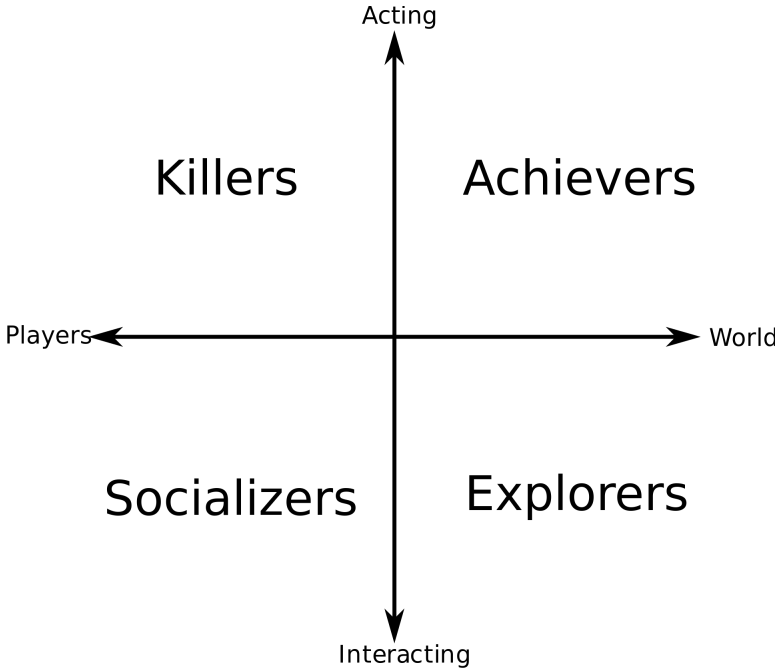


Abbildung 3.3: Spielertypen. Bartle [7]

als Grundlage für eine *Social Action Matrix* genommen hat (Abbildung 3.4), die jedem Spielertypus Verben zuordnet. Dabei fällt auf, dass aus der Welt-Achse eine *content*-Achse wird, um eine größere Verallgemeinerung zu ermöglichen. Auch verschieben sich einige Konzepte. Ist bei Bartle der Achiever oben rechts auf das Sammeln von Punkten fixiert, findet man diese Aktivität bei Kim eher im Bereich Compete (Compare, Showoff) oder Explore (Collect).

Bei der Verwendung von Spielertypen für das Game Design gibt es jedoch die Gefahr, dass sie zu *„self-fulfilling and self-validating“* Prophezeihungen werden — am Ende stimmt das Spiel exakt mit der verwendeten Typographie überein [122, S. 12].

Auch wurden bei der Untersuchung von Spielertypographien einige Genres (MMO, RPG, FPS) öfter und extensiver untersucht als andere [122, S. 12] — diese Genres sind auch die kommerziell erfolgreichen und verfügen über eine große Spieleranzahl. Es bleibt dementsprechend die Frage, wie gut sich die Spielertypographien auf kleinere „Nischen“-Genres (beispielsweise das Wimmelbildspiel (Abschnitt 3.7)) anwenden lassen.

3.5 Bildauswertung

Die Bildauswertung beschäftigt sich mit der Interpretation und Auswertung Bildern: *„When we can identify what we see on [. . .] images and communicate this information to others, we are practicing image interpretation. [. . .] Images contain raw image data. These data, when processed by a human interpreter's brain, become usable information“* [64, S. 193].

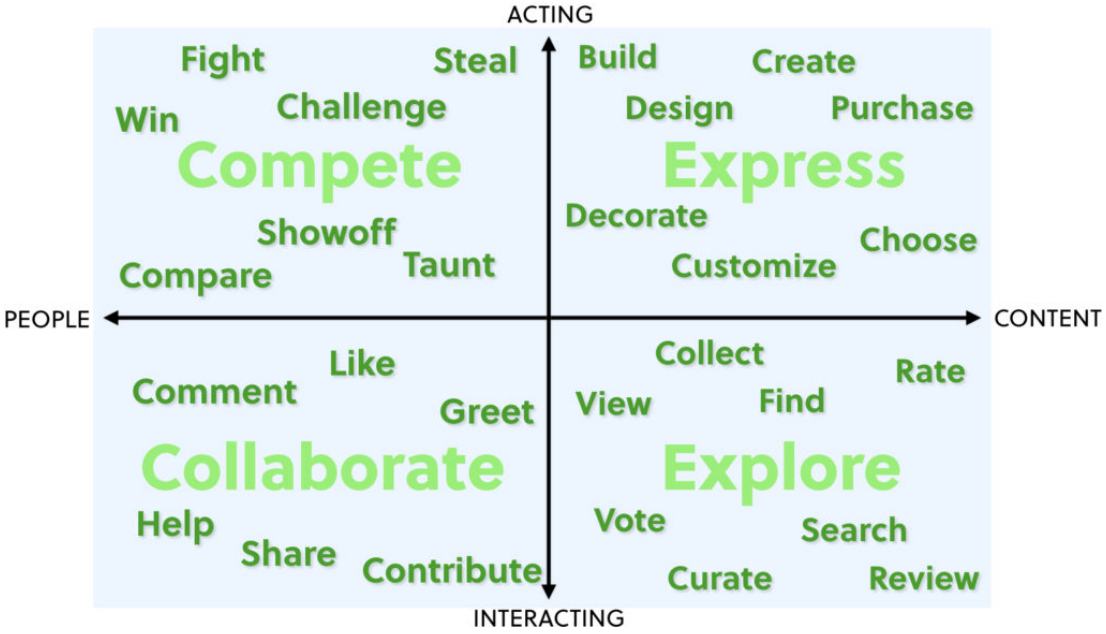


Abbildung 3.4: *Social Action Matrix.* Kim [59]

Anwendungskontext dieser Arbeit ist die Luftbildauswertung; die Interpretation und Auswertung von Luft-, Satelliten- und Radarbildern [110, S. 267].

Die Luftbildauswertung (im weiteren nur Bildauswertung genannt) ist eine *„seit langem bekannte und mittlerweile ausgereifte Disziplin der Geowissenschaften“* [100, S. 2]. Sie ist komplex, vielseitig und mit zivilen und militärischen Nutzen, beispielsweise für agrikulturelle Gutachten und Mineralerkundungen [64, S. 194], Vogelbeobachtung und Naturschutz [100], dem Monitoring von Wildtieren in der Savanne [97] oder der Suche nach Fliegerbomben aus dem zweiten Weltkrieg (Kampfmittelvorerkundung [69]).

Ein in der Geschichte bedeutender Einsatz der Bildauswertung ist die Kubakrise (1962): Durch von Aufklärungsflugzeugen angefertigten Luftbildern (Abbildung 3.5) konnte der vermutete Bau sowjetischer Raktenstellungen auf Kuba nachgewiesen werden.

In dieser Arbeit liegt der Fokus auf einem Teilaspekt der Bildauswertung — der Typenkunde. Dabei geht es um das Unterscheiden von Typen: Ein Bildauswerter muss nicht nur erkennen, *ob* ein interessantes Objekt zu sehen ist, sondern auch *was* das für ein Objekt ist, bzw. *welchen* Typen es besitzt, und wie es sich von vergleichbaren Objekten unterscheidet[113, S. 4]. *„In a sense, the image interpretation process is like the work of a detective trying to put all the pieces of evidence together to solve a mystery.“* [64, S. 201]

Auf das Beispiel (Abbildung 3.5) angewandt bedeutet das, dass nicht nur erkannt werden muss *ob* militärische Strutkuren erkannt wurden (Ja.), sondern auch, *welche* Sturkturen das sind (Raktenstellungen, Ausrüstung, Zelte, . . .). Denkt man sich die Annotationen im Bild weg, wird die Herausforderung im Bildauswerteralltag deutlich. Es ist anzunehmen, dass kaum ein Leser die Besonderheiten im

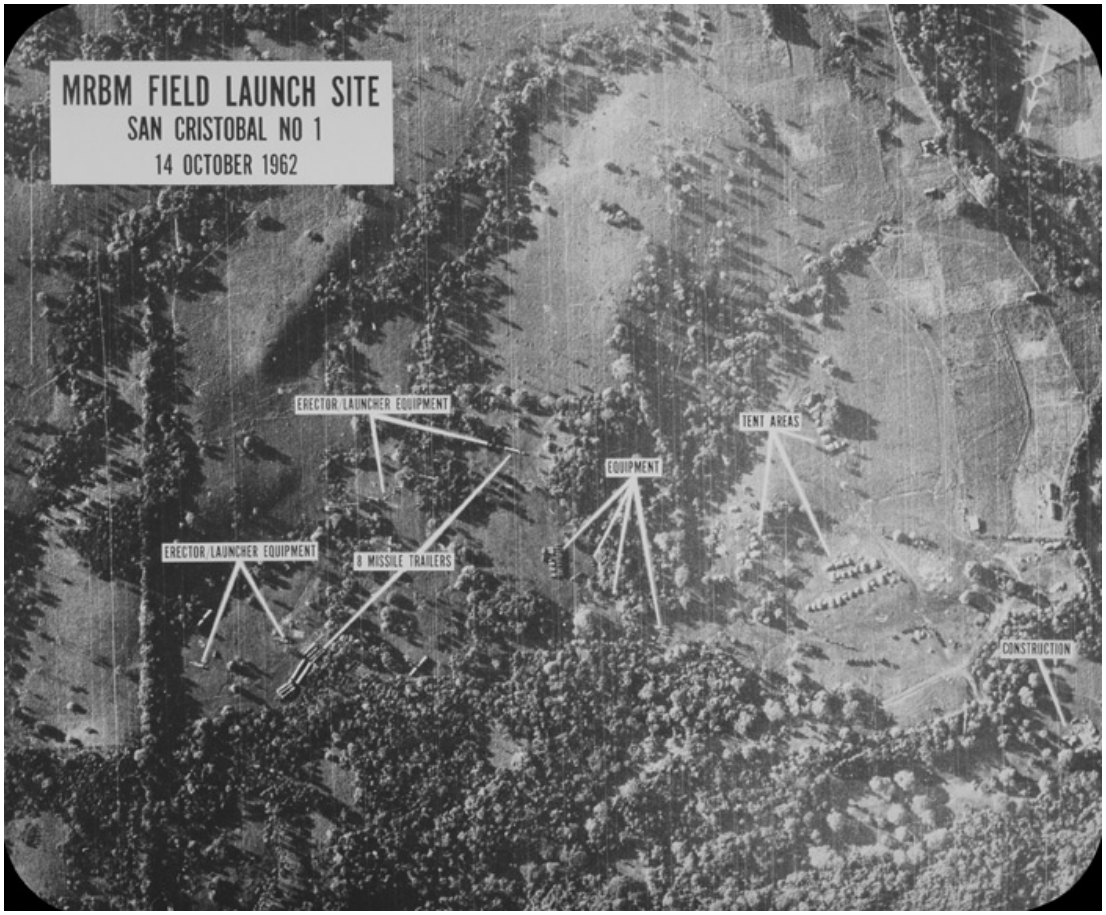


Abbildung 3.5: Sowjetische Raketenstellung auf Kuba, 1962

gezeigten Bild erkennen würde.

Gründe für Schwierigkeiten in der Interpretation sind vielfältig. Lillesand et al. [64, S. 195ff] nenen wichtige Merkmale, die bei der Interpretation eines Bildes berücksichtigt werden müssen:

- *Shape.* Die Form eines Objekts
- *Size.* Die Größe eines Objekts
- *Pattern.* Die räumliche Anordnung von Objekten
- *Tone/Hue.* Die Farbe/Helligkeit eines Objekts
- *Texture.* Die Häufigkeit eines Farbewertes. Blätter einer Baumart erscheinen von weit oben betrachtet in der selben Farbe, obwohl sie bei näherer Betrachtung alle unterschiedlich gefärbt sind.
- *Shadow.* Schatten erlauben Rückschlüsse auf die Form von Objekten und die Tageszeit (gut), blockieren aber andere Objekte (schlecht).
- *Site.* Topographische und geographische Gegebenheiten und Klima.
- *Association.* Ein Flugzeug auf einem Flugplatz ist einfacher zu erkennen als

ein Flugzeug auf einem Acker.

- *Resolution.* Bei geringer Auflösung ist es schwierig Details zu erkennen/unterscheiden.
- *Other.* Bildskalierung, Farbbalance, Qualität der Bilder, ...

3.6 Prozedurale Generierung

PCG ist eine in der Spiele-Entwicklung gebräuchliche Methodik um billig, schnell und effizient Spielinhalte (*Content*) zu erzeugen[56]. Sie bezeichnet *„the creation of game content [...] automatically (or semi-automatically), through algorithmic means“* [135, S. 1].

Grundgedanke dabei ist, dass ein Computer effizienter als ein Mensch arbeitet: Er kann schneller bis zur millionsten Stelle von π zählen, effizienter ein Bild tausend mal kopieren und besser hunderttausend Varianten eines Baumes generieren. Dementsprechend ist das Grundziel von PCG *„[enchancement of] replayability without designer effort“ [138, S.6].*

Während ein Computerspiel komplex ist [80], so besteht es im wesentlichen doch „nur“ aus einer geringen Menge an immer wiederkehrenden elementaren visuellen und auditiven Bausteinen, von Hendrikx et al. *Game Bits* genannt[50], beispielsweise

- Textures
- Sound
- Vegetation
- Buildings
- Fire, Water, Stone, Clouds

Diese *Game Bits* lassen sich *prozedural*, also als computer-lesbare Reihenfolge von Anweisungen, beschreiben und generieren[56, S. 89]: *„The key property of procedural generation is that it describes the entity, be it geometry, texture or effect, in terms of a sequence of generation instructions rather than as a static block of data. The instructions can then be called on when required to create instances of the asset and the description can be parametrised to allow the generation of instances with varying characteristics.“*

Während ein Mensch manuell, teuer und zeit-intensiv eine Menge an Bäumen erstellen muss, kann ein Computer –sofern er die Beschreibung eines Baums kennt– auf Abruf eine beliebige Anzahl an Bäumen erstellen. Dies befreit den Menschen von einer repetetiven Arbeit, senkt die Entwicklungskosten[3], erlaubt ein Skalieren der Spielinhalte[4] und verringert auch die Speichergröße des fertigen

[3]Es werden weniger Menschen für das Erstellen des Spiels gebraucht

[4]Dem Computer ist es egal, ob er einen oder einen Million Bäume generiert

Spiels[5].

Nach Ebert et al. [34, S. 2] verfügen prozedurale Techniken über drei bedeutende Eigenschaften:

- *Abstraction.* Inhalte und Objekte werden nicht als konkrete Form sondern als abstrakte, computer-lesbare Beschreibung (als Funktion oder Algorithmus) gespeichert. Diese Beschreibung kann dann nach Bedarf wieder verwendet werden, um die beschriebene Form zu erzeugen. Dadurch verschiebt sich der Erstellungsvorgang vom Menschen zum Computer. item *Parametric control.* Die abstrakte Form läßt sich parametrisieren. Einzelne Eigenschaften des zu erzeugenden Objekts können also gezielt bestimmt und gesteuert werden; mit wenigen Parametern kommt man trotzdem zu hohem Detail.
- *Flexibility.* Das zu erzeugende Objekt läßt sich beliebig erweitern und verändern. Es unterliegt nicht mehr den originalen Einschränkungen und Limitierungen und kann für jeden Anwendungskontext angepasst werden.

Besonders der Aspekt *Parametric control* ist für den Kontext dieser Arbeit (Parametrisierung) interessant — „it allows the generation to be easily managed and enables vastly detailed scenes to be defined in the terms of a few parameters“[56].

Es gibt eine Vielzahl an prozeduralen Techniken. Für die in dieser Arbeit beschriebene Spiele-Entwicklung sind vor allem *Perlin-Noise*, *L-Systeme*, *Fraktale*, *Voronoi-Diagramme* und *Tiling* interessant.

Weitere prozedurale Techniken finden sich z.B. bei Hendrikx et al. [50] oder Smelik et al. [106].

3.6.1 Fraktale

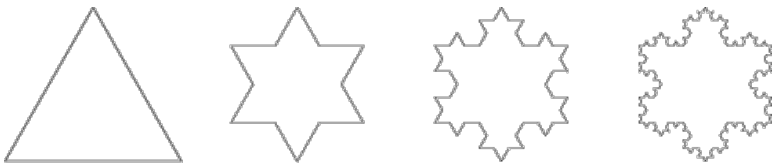


Abbildung 3.6: Die ersten Iterationen der Koch-Flocke. Grafik von [128]

Fraktale (von lat. *fraktus* = gebrochen) sind eine von Benoit Mandelbrot entdeckte[6] „nicht-euklidische“ Geometrie, mit der sich viele in der freien Natur vorkommende Strukturen beschreiben lassen.

Interessant sind Fraktale für die PCG, weil sie *self-similar* sind und über *infinite detail* verfügen[71] — dies erlaubt komplexe Objekte durch einer einfachen rekursiven Funktion auszudrücken ([50], S.24).

[5]Besonders diese letzte Tugend war lange Zeit das ausschlaggebende Argument für den Einsatz von PCG — und wird auch heute noch in der *demo*-Szene gepflegt. Dabei geht es um das Erstellen von *Demos* –„Ausführbare Programme, die in Echtzeit Computergrafiken und Musik darstellen[31]“– mit möglichst geringer Speichergröße

[6]Eigentlich eher ein Zusammenfassen von bisher unbeachteten und ignorierten Aspekten der euklidischen Geometrie

Beispiel: Die Koch-Flocke (Abbildung 3.6). Bei jeder Iteration werden die selben Merkmale rekursiv fortgesetzt (*self-similarity*). Die Iterationen können bis in die Unendlichkeit fortgesetzt werden (*infinite detail*).

3.6.2 L-Systeme

L-Systeme wurden vom Biologen Aristrid Lindenmayer 1968 als formale Sprache entworfen, um das Wachstum/die Entwicklung von einfachen Organismen (Algen) zu beschreiben [65]. Schnell hat man aber festgestellt, dass sich L-Systeme auch zum Beschreiben von komplexeren Strukturen, z.B. „höheren“ Pflanzen eignen [95].

Im wesentlichen basieren L-Systeme auf *rewriting*: „a technique for defining complex objects by successively replacing parts of a simple initial object using a set of rewriting rules or productions“ [95].

Beispiel: Ein Ausgangswort *ab*, *axiom* genannt, zusammen mit einer Regel $b \rightarrow ab$ führt nach einem Ersetzungsschritt zu einem Wort *aab*. Das *b* wurde ersetzt. Nach einem zweiten Ersetzungsschritt lautet sich das Wort *aaab*.

Richtig interessant werden L-Systeme erst in Verbindung mit *Schildkrötengrafiken*[94]. Dazu denke man sich eine Schildkröte, die über ein Stück Papier krabbelt und einen Stift hinter sich her zieht. Dieser Schildkröte können Befehle erteilt werden [1]:

- Start — Bewege dich vorwärts
- Stop — Höre auf dich zu bewegen
- Drehung — Drehe dich um x° Grad
- Zeichnen — Setze den Stift auf/ab

Auf das Wort *aaab* angewandt könnte man so das Wachstum eines Regenwurms zeigen — Der Rumpf (a) wächst, während der Kopf (b) immer gleich bleibt[7].

In der PCG werden L-Systeme für komplexe, regel-basierte (meist organische) Strukturen verwendet, z.B. Bäume [107] oder Städte [84].

Ähnlich wie Fraktale (Unterabschnitt 3.6.1) eignen sich auch L-Systeme gut für ein rekursives Vereinfachen von komplexen Strukturen in einfache Anweisungen.

3.6.3 Rauschen (Noise)

Die vielleicht wichtigste prozedurale Technik ist das Rauschen. Besonders das von Perlin geprägte Verfahren (Perlin-Noise[85]) findet große Verwendung. Dabei wird mittels einer pseudo-zufälligen Noise-Funktion Pseudo-Random Number Generators (PRNG) erzeugt ein Mix aus Daten (Zahlen).

Diese Daten können dann interpoliert und zu einem vielseitig-verwendbaren Rauschen-Bild zusammengefügt werden (3.7). So kann man das Bild verwenden

[7] Ohne Anspruch auf biologische Richtigkeit

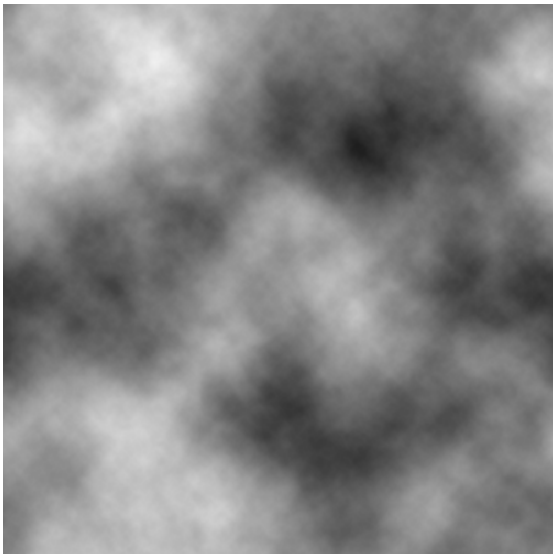


Abbildung 3.7: Zweidimensionales Perlin-Rauschen. Bild von Wikipedia, Public Domain

um das Höhenrelief eines Gebirges abzubilden, oder es mit weiteren Rauschen-Bildern zusammenfügen, um eine detaillierte Textur zu erstellen [56, S. 96ff].

Die Interpolation verleiht dem Rauschen-Bild ähnlich einem Fraktal „infinite detail".

Das wichtigste Stichwort aber ist die *Pseudo-Zufälligkeit*: *„Noise appears random, but isn't really. If it were really random, then you'd get a different result every time you call it. Instead, it's* pseudo-random *— it gives the appearance of randomness"*[86, s. 2-1]. Das Rauschen sieht nur zufällig aus — wird die Rauschen-Funktion aber wiederholt unter gleichen Bedingungen (gleicher Startwert, genannt *seed*) aufgerufen, so liefert sie immer das selbe Ergebnis! Das erzeugte Ergebnis ist also nicht zufällig, sondern reproduzierbar [56, s. 97].

3.6.4 Tiling

Das *Tiling* greift die o.g. Analogie der *Game Bits* auf: Ein großer, komplexer Raum (meist eine Spielkarte) wird aus verschiedenen kleinen Kacheln zusammengesetzt. Schon eine geringe Anzahl an Kacheln erlaubt es anspruchsvolle Landschaften zu erzeugen. Auch läßt sich so die Spielwelt *just in time* generieren, und muss nicht von vornherein vordefiniert werden.

Beispielsweise kann man das im vorigen Abschnitt genannte Rauschen-Bild als Grundlage nehmen, und bestimmten Zahlenwerten einen bestimmten Kacheltypen zuordnen. Aus dieser Kombination läßt sich eine undendlich-große Spielwelt erzeugen — eine Technik, die das Spiel *Minecraft* popularisiert hat [75].

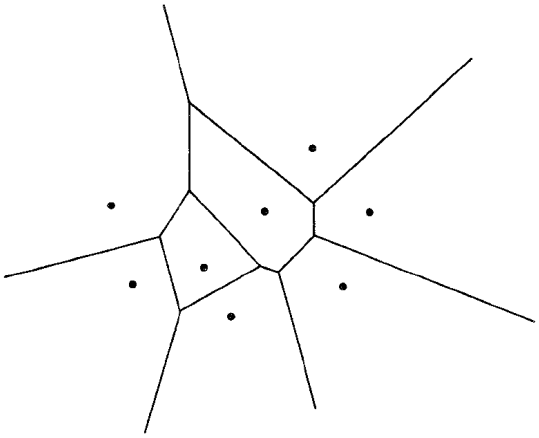


Abbildung 3.8: Voronoi-Diagramm. Grafik von [5, s. 347]

3.6.5 Voronoi-Diagramm

Während das *Tiling* eine geordnete, lineare, zweidimensionale Grundstruktur voraussetzt[8], agiert ein Voronoi-Diagramm auf einer Menge an Punkten. Diese Punkte werden benutzt, um den Raum in dem sie sich befinden zu zerteilen: *„Each point is associated with the region of the plane closest to it“* [5, s. 346].

Wie auch die vorherigen prozeduralen Techniken bildet ein Voronoi-Diagramm (3.8) Aspekte der Natur ab — z.B. Zellen oder Baumrinde.

3.7 Wimmelbildspiele

Der Begriff „Wimmelbild“ stammt aus der Kunstgeschichte und bezeichnet *„das kaum in Worte fassbare Mit- und Durcheinander unterschiedlicher Figuren und Handlungen“* [21, S. 102]. Viele Werke des Malers Hieronymus Bosch können als Wimmelbild bezeichnet werden. Beispielsweise „Der Garten der Lüste“ (Abbildung 3.9), auf dem viele verschiedene Figuren in unterschiedlichen Beschäftigungen abgebildet sind. Fordert man einen Betrachter auf, ein bestimmtes Objekt[9] auf dem Bild zu suchen, wäre dieser eine Weile beschäftigt, da das Bild von Objekten „wimmelt“. Diese Aufforderung kann man als Spiel sehen (Abschnitt 3.1).

Ein moderneres Beispiel für Wimmelbilder, und eine Form von Wimmelbildspielen[10], ist die Buchreihe „Wo ist Walter?“[49], in der auf jeder Seite der gleichnamige Hauptcharackter (Walter) gesucht werden muss.

Im Bereich der Computerspiele gibt es das kommerziell sehr erfolgreiche Genre der „hidden object games“ bzw. *Wimmelspiele*[11] — beispielsweise nennt der auf „hidden object games“ spezialisierte schwedische Spieleentwickler und Publisher G5 einen Jahresumsatz von SEK 335M für 2020 [43]. Interessant ist dabei auch,

[8] Die Kacheln müssen aneinanderpassen

[9] Beispielsweise ein Stachelschwein

[10] Explizite Handlungsaufforderung zum Suchen eines Objekts

[11] Laut Wikipedia [129] zählt es zu den Bild- und Puzzlespielen.

Abbildung 3.9: Wimmelbilder in der Kunst: Der Garten der Lüste, Mitteltafel. Hieronymus Bosch, ca. 1500

dass die Zielgruppe von G5 (und von „hidden object games“ allgemein) (ältere) Frauen sind [42][127][99, S. 123] — dies zeigt, dass das Vorurteil von Spielern als jung und männlich nicht zeitgemäß ist.

„Hidden object games“ sind immersiv und erzählen eine Geschichte [53, S. 161ff]: Oft wird der Spieler mit dem Lösen einer Aufgabe beauftragt. Dazu muss er von einem Schauplatz zum nächsten (jeweils als Wimmelbild abgebildet) und dort verschiedene Objekte finden (daher der Name „hidden objects“), die ihm wiederum Zugang zu anderen Schauplätzen liefern. Beispielsweise muss ein Schlüssel gefunden, und eine Tür geöffnet werden. Diese Suchen-und-Finden-Kette kann beliebig lang werden.

Im Bereich der Wissenschaft sind „hidden object games“ bis jetzt nicht im Fokus. Alghamadi (2014) [3] und Rangkuti (2019) [96] haben sie zum Lernen von

englischen Vokabeln verwendet. Feng et al. (2012) [37] schlagen den Einsatz von Purposive Hidden Object Games (P-HOG) als Crowdsourcing-Ansatz vor, um schnell und effizient Bilder mit korrekten Annotationen zu versehen (Immage Annotation). Dieser Ansatz wurde schon mit anderen Spieltypen versucht [125]. Auch erinnert er an das Lösen von CAPTCHAs [62].[12] Oei und Patterson [81, S.2f, S.13f] haben gezeigt, dass „hidden object games“ Kognition und visuelle Wahrnehmung ansprechen und sich dementsprechend zum Trainieren derselben eignen.

Wimmelbildspiele zeigen in ihrer Spielart eine Nähe zu den Anforderungen der Bildasuwertung (Abschnitt 3.5).

[12] Überlegung: Eine Kombination aus CAPTCHA und Minispiel wäre interessant.

Kapitel 4

Methodik

Die in dieser Arbeit zu untersuchende Frage lautet:

> *Wie parametrisiert man ein adaptives Lernspiel?*

Sie kann in drei Teilfragen unterteilt werden:

- Was ist ein adaptives Lernspiel?
- Wie erstellt man ein adaptives Lernspiel?
- Wie parametrisiert man ein adaptives Lernspiel?

Zur Beantwortung der Fragen wird im ersten Schritt das adpative Lernspiel definiert und in den im Grundlagenkapitel gezeigten Gesamtkontext adaptiver Systeme (Abschnitt 3.2) eingeordnet. Anschließend wird ein Verfahren gezeigt, wie man ein solches adaptives Spiel erstellen kann. Schließlich wird das Verfahren angewendet, um einen Prototypen eines adaptiven Spiels zu erzeugen und die Frage der Parametrisierung zu beantworten.

4.1 Adaptive Spiele

Was ist ein adaptives Lernspiel?

Gemäß dem Grundlagenkapitel (Abschnitt 3.1) ist ein adaptives Lernspiel ein Spiel, dass in einem Lernkontext eingesetzt wird, mit der Besonderheit, dass es adaptiv ist. Es kann demnach durch den Adaptivitätszyklus von Shute und Zapata-Rivera (Siehe Grundlagen Unterabschnitt 3.2.3) beschrieben werden.

Dabei kann das adaptive Spiel Teil eines größeren Systems (möglicherweise gamifiziert), oder eigenständig sein. Auch kann es in Spiel und System aufgeteilt werden. Dabei bildet das Spiel die mit dem Spieler/Lerner interagierende Komponente — also die *Capture*- und *Present*-Ebene.

Vor allem letztere ist mit Blick auf die Parametrisierung interessant (Abbildung 4.1): Für den Lerner relevante Informationen und Lernstrategien wurden bereits ausgewählt (4), und müssen nun in eine lernbare Form gebracht werden

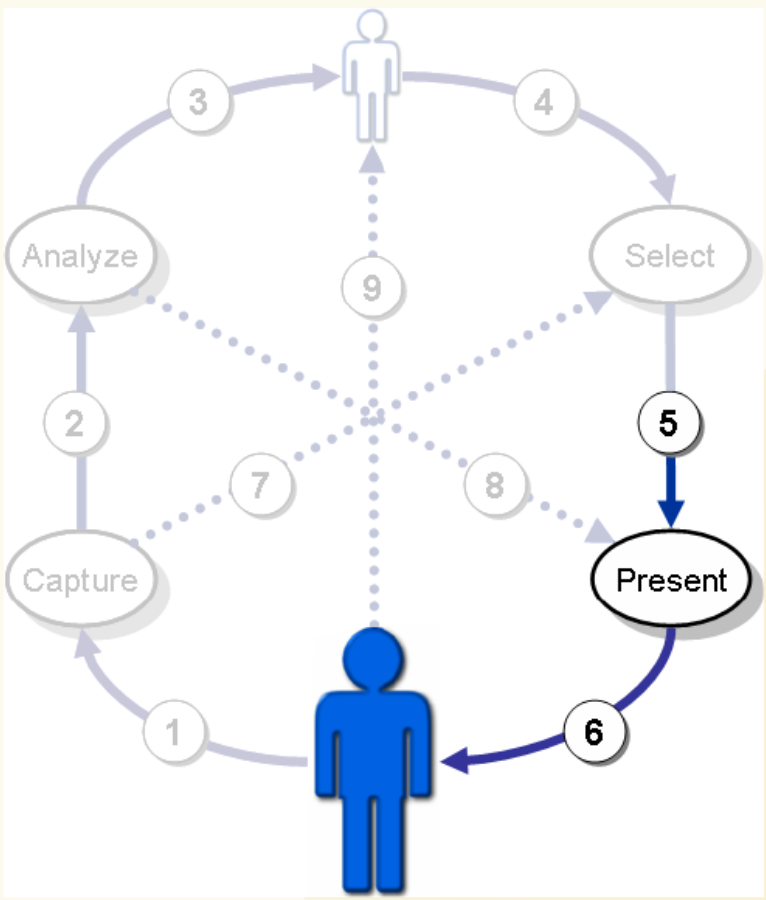


Abbildung 4.1: Das adaptive Spiel im Fokus der Parametrisierung. Originalschaubild aus [105].

(5), so dass sie dem Lerner vorgelegt werden können (6). Dabei umfasst die in (5) gewählte Präsentationsform die Art und den Aufbau des Spieles in Bezug auf das Lernermodell. Dazu gehört auch eine auf den Lerner angepasste Schwierigkeit, und demgemäß die Parametrisierung.

4.2 Erstellung adaptiver Spiele

Wie erstellt man ein adaptives Spiel?

Selbst unter Aufspaltung des adaptiven Spiels in Spiel und System bleibt die Erstellung komplex. So erfordert sie *„adequate game user experience design which must match the player type, play style, and preferences of as many users as possible“* [115, S. 3]. Auch wird großer Fokus auf *„good playability and player experience“* [115, S. 14] gelegt.

Mit Blick auf die Parametrisierung wird der Fokus auf die Erstellung der Spiel-Seite des adaptiven Spiels gelegt. Diese umfasst *Game Design*[99][98][61], aber auch *Gamification Design* [77][29], um den Bereich der Nicht-Spiele abzudecken. Wie im Grundlagenkapitel gezeigt existiert ein fließender Übergang zwischen Spiel und Nicht-Spiel und eine Verwandschaft zwischen Gamification und DGBL

(Abschnitt 3.3).

Dabei sei dazugesagt, dass Gamification -wie Game Design– ein kreativer und interaktiver Designprozess ist, für den die Verwendung strikter Design-Regeln eher schädlich ist, und nicht durch eine „Anleitung“ abgebildet werden kann [78, S. 1304ff]. Auch gleicht sich kein Spiel und kein Anwendungskontext. Die in dieser Arbeit gewählten Schritte zum Erstellen eines adaptiven Spiels sind auf die Bedürfnisse dieser Arbeit zugeschnitten und sollten nicht blind kopiert werden. Das Spiel wird prototypisch erstellt, mit Fokus auf die Parametrisierung. Die verwendeten Spielinhalte sind provisorischer Natur und nicht unter Aspekten des UCD gewählt.

Die Erstellung des Prototypen erfolgt (angelehnt an Morschheuser et al. [77] und *The Formal Loop* [99, S. 108]) in sechs Phasen:

- Projektvorbereitung
- Kontext- und Zielgruppenanalyse
- Ideenfindung
- Konzeption
- Implementierung
- Evaluation

Im ersten Schritt erfolgt die Projektvorbereitung, in der der Rahmen des Projekts vorgegeben und die Problemstellung formuiert wird [99, S. 73f]. Was ist der Anwendungskontext? Was sind die Anforderungen? Was ist das Problem? Was ist das Ziel? Was soll erreicht werden (*Mission statement*)? Welche Mittel stehen zur Erreichung des Ziels zur Verfügung? Unterliegt das Projekt technischen oder legalen Einschränkungen?

Anschließend –und ausgehend aus der Projektvorbereitung– wird der Anwendungskontext und die Zielgruppe analysiert, um *user-centered* vorzugehen: Wer solldas Spiel spielen? Was sind die Bedürfnisse der Zielgruppe? Welche Gemeinsamkeiten gibt es? Gibt es kulturelle Besonderheiten, die angesprochen werden müssen? Müssen auf körperliche Einschränkungen Rücksicht genommen werden?

Diese Fragen sind wichtig vorab zu klären, da sie eine realistische Einschätzung des Projekts ermöglichen, die Richtung für das Vorhaben vorgeben und den Rahmen für die Kreativphase bilden. Wichtig ist auch, dass das angegebene Ziel überprüfbar ist, damit das Vorhaben später überprüft werden kann.

In einer Kreativphase (der Ideenfindung) werden Lösungen für das gestellte Problem gesucht. Dabei werden in einem mehrstufigen iterativen Verfahren mittels verschiedener Kreativtechniken (Brainstorming, Mindmaps, Mockups, Moodmaps) Ideen gesammelt, diskutiert und verworfen, bis schließlich ein Konzept (*Game Design*) für das Spiel entsteht.

Schließlich wird das Konzept in einer Implementierung umgesetzt, die dann wiederum durch einen Benutzertest evalziert wird.

Das gewählte Verfahren läßt sich an einer Parabel veranschaulichen. Aufgabe ist es, von einem Punkt A zu einem Punkt B zu gelangen. Zwischen den beiden Punkten klafft ein Abgrund. Die Projektvorbereitung umreißt das Vorhaben (Von A nach B zu gelangen) und gibt einen Überblick über die vorhandenen Kapazitäten. In der Analyse wird die Beschaffenheit von Punkt A, die Größe und Weite der Klippe und das Vorhandensein eventueller Erdbeben oder Windboöen begutachtet, und auch versucht auszuspähen, wie Punkt B aussieht. Die Ideenfindung sucht nach einem Weg den Abgrund zu überqueren, und zeigt eine Lösung auf (eine Brücke). Diese wird schließlich gebaut und von einem glücklichen Freiwilligen getestet.

4.3 Parametrisierung

Wie parametrisiert man ein adaptives Spiel?

Mit Blick auf die drei W-Fragen bei der Erstellung eines adaptiven Systems (Unterabschnitt 3.2.2), beantwortet die Parametrisierung das *what to adapt* — wie muss das Spiel aufgebaut sein, damit es sich anpassen läßt. Dies ist eine Frage der Konzeption, und somit eine Frage des *Game Design*.

Dabei geht aus dem Adaptivitätszyklus (Abbildung 4.1) hervor, dass die Art der Anpassung in (5), also außerhalb der *Present*-Ebene gewählt wurde. Dies muss während der Konzeption berücksichtig werden.

Kapitel 5

Konzeption

In diesem Kapitel wird –basierend auf dem zuvor vorgestellten Verfahren zum Erzeugen adaptiver Spiele (Abschnitt 4.2)– das Spiel konzipiert, beginnend mit der Projektvorbereitung.

5.1 Projektvorbereitung

In der Projektvorbereitung wird der allgemeine Rahmen des Projekts definiert, die Problemstellung gegeben, und ein (überprüfbares) Ziel genannt, mit dessen Hilfe das Projekt später bewertet wird. Der allgemeine Rahmen gibt einen Überblick über das Projekt, schränkt es ein und fungiert als Wegweiser für die Entwicklung.[1]

Darüber hinaus gilt es weitere projektmanagerische Überlegungen (Zeitplanung, Teamplanung, Budgetplanung, Dokumentation, ...) zu tätigen. Diese sind aber in Bezug auf die wissenschaftliche Arbeit irrelevant und werden nicht besprochen.

Problemstellung. Im Zuge der Erforschung der DDA am IOSB soll ein Prototyp eines Wimmelbildspiels als *Testbed* entwickelt werden, um die Parametrisierung adaptiver Spiele testen zu können [109].

Ziele. Das Wimmelbildspiel soll adaptierbar und parametrisierbar sein; es soll sich entsprechend von DDA-Parametern adaptieren (AR, Siehe Unterabschnitt 3.2.4).

Anwendungskontext. Der Anwendungskontext ist die (milit.) Bildauswertung (Abschnitt 3.5). Sie stellt die Wissensdomäne der Forschungsgruppe dar und bildet die Grundlage für das Wimmelbildspiel. Dieses bildet eine Übungsaufgabe aus der Bildauswertung ab, und soll dazu genutzt werden, dass angehende Bildauswerter die Typenkunde trainieren können.

Projektbedingungen und Einschränkungen. Das Projekt soll unter Verwendung von Web-Technologien erstellt werden, um in die vorhandene Testing-Struktureingebunden werden zu können. Es stehen keine benutzbaren Spielinhalte (Grafiken, Modelle) zur Verfügung.

[1] Ähnlich der Einleitung einer wissenschaftlichen Ausarbeitung.

Im nächsten Schritt werden Anwendungskontext und Zielgruppe näher betrachet, um eine zielgruppenrelevante Ideenfindung, Konzeption und Implementierung gewährleisten zu können.

5.2 Kontext- und Zielgruppenanalyse

Kern jedes menschengerichteten Unterfangens ist der Mensch. Diesen Ansatz nennt man UCD. Da das Wimmelbildspiel von angehenden Bildauswertern gespielt werden soll, müssen diese so früh wie möglich in die Konzipierung einbezogen werden, da die Bedürfnisse der Zielgruppe [118, S. 145] die Form des Spiels, die vorhandenen Funktionen und Mechaniken, und die generelle Richtung der Konzeption vorgeben. Auch muss ihr Einsatz- und Arbeitsgebiet untersucht werden.

Dies geschieht auf Grundlage der Expertise des IOSB, das sich seit mehreren Jahren mit dem Themenkomplex befasst, Vorstudien zur Zielgruppe erstellt hat und in Kooperation mit dem Ausbildungszentrum für abbildende Aufklärung der Luftwaffe (AZaaLw) steht.

Normalerweise würden für das UCD echte Bildauswerter bereitstehen, die für eine Zielgruppenanalyse befragt werden könnten, dies war aber auf Grund der COVID-19-Pandemie nicht möglich.

Deshalb wird in diesem Abschnitt –basierend auf öffentlich auffindbaren Informationen zum Themengebiet– eine Kontext- und Zielgruppenbetrachtung durchgeführt, um zu zeigen, wie UCD in Zeiten eines globalen „Lockdowns“ zumindest ansatzweise möglich ist. Auch lassen sich Rückschlüsse aus der Beschreibung des Wimmelbildspiels zu ziehen.

5.2.1 Bildauswertung

Anwendungskontext ist die Bildauswertung (Abschnitt 3.5), genauer gesagt die militärische Bildauswertung. Die Aufgabe eines Bildauswerters ist es, Luft-, Satelliten- und Radarbilder auszuwerten.

Im Weiteren werden Beruf und Tätigkeit eines Bildauswerters aus Sicht der militärischen Bildauswertung beschrieben, basierend auf der öffentlichen Stellenauschreibung der Bundeswehr [19]:

5.2.2 Militärische Bildauswertung

Der Beruf als „Spezialistin / Spezialist für Luftbildauswertung“ ist eine Offizierslaufbahn (Laufbahn der Feldwebel) [19]:

> Der Luftbilddienst der Luftwaffe unterstützt die Aufklärung sowohl im Inland als auch in den Einsatzländern. Als Spezialistin bzw. Spezialist für Luftbildauswertung analysieren Sie die Bild- und Sensordaten der Aufklärungsmittel der Streitkräfte. Mit Ihren fundierten Berichten und Empfehlungen tragen [Sie] zur Sicherheit Ihrer Kameradinnen und Kameraden im Einsatzgebiet bei.

Die Bundeswehr bietet unentschlossenene Bewerbern eine kostenlose Orientierungshilfe auf Basis von sechs Multiple-Choice-Fragen [18]. Die folgenden Antworten[2] führen zum o.g. Beruf.:

- Wo würden Sie lieber arbeiten? Draußen **Drinnen**.
- Möchten Sie in Ihrem Beruf auch eine Schusswaffe verwenden? **Ja.** Nein.
- Mögen Sie körperliche Herausforderungen? Ja. **Nein.**
- Treffen Sie gerne Entscheidungen und übernehmen Sie dafür auch die Verantwortung? **Ja.** Nein.
- Arbeiten Sie lieber mit Menschen oder mit Technik? Menschen. **Technik.**
- Haben Sie auch Interesse an medizinischen Berufen? Ja. **Nein.**

Der Punkt „Gebrauch der Schusswaffe“ mag auf den ersten Blick bei einem „Drinnen“-Beruf verwundern, gehört aber zur Laufbahn der Offiziere dazu — die oben getätigten Eingaben führen zu 45 Berufen der Offizierslaufbahn. Dabei wird der „Gebrauch der Schusswaffe“ auch von „IT-Administratoren“, „Mediengestaltern“, „Logistikern“ oder „Orchestermusikern“ gefordert[3].

Aufgaben eines militärischen Bildauswerters sind das Auswerten, Analysieren, Dokumentieren und Organisieren von Bild- und Sensordaten, das eigenverantwortliche Erstellen von Messungen und Berechnungen, die Präsentation der erstellten Analysen und Messungen, und die Teilnahme und Durchführung von Ausbildungen, Fortbildungen und Lehrgängen. Die Bildauswertung erfordert *„ein gewisses Auge“* [15, Ca 2:10] und die Erkennung von komplexen, oft geographischen und geschichtlichen Zusammenhängen — beispielsweise die kartographische Orientierung auf einem fremden Erdteil [15, Ab 2:10].

Die Bildauswertertätigkeit erfordert eine bundesweite Versetzbarkeit und ist mit der Teilnahme an Auslandseinsätzen verbunden. Bei der militärischen Bildauswertung handelt es sich um eine computerbasierte Schreibtischtätigkeit, die in die technischen Infrastruktur der Bundeswehr eingebunden ist. Gleichzeitig wird in der Laufbahn als Feldwebel *„körperliche Fitness [benötigt], um auch im heißen Klima während eines Auslandseinsatzes alle Aufgaben erfüllen zu können“* [16]. Dabei sind Bildauswerter in ihrer Tätigkeit als Feldwebel *„militärische Vorgesetzte, das heißt, dass sie Soldatinnen und Soldaten führen und ausbilden.“* [16].

Die Bundeswehr stellt keine hohen Anforderungen an angehene Bildauswerter — sie müssen nur mindestens 17 Jahre alt sein und einen mittleren Schulabschluss (oder eine Berufsausbildung). Dafür gibt es ein Höchstalter von 29 Jahren[4] [17]. Der Frauenanteil bei der Bewerbung auf Offiziersstellen beträgt 24% (Stand 2019) [12].

Eine Nicht-Bundeswehr-Stellenausschreibung fordert von einzustellenden Bilauswertern „stereoskopische Sehfähigkeit“ [39]. Es ist anzunehmen, dass dies auch auf

[2] Die Anworten wurden gekürzt.
[3] James Bond läßt grüßen.
[4] Mit Ausnahmen

die Bundeswehr zutrifft.

Zusammengefasst läßt sich die Zielgruppe der angehenden Bildauswertern als heterogene, verantwortungsbewußte, abenteuerlustige, technikaffine überwiegend männliche Gruppe im Alter zwischen 17 und 29 Jahren bezeichnen. Diese Beschreibung deckt sich mit Streicher et al. [115, S. 28], die die Zielgruppe angehender Bildauswerter als *„students, mainly from the Generation Y, which are eager to play and are going to be trained as image interpreters“* bezeichnen.

5.2.3 Wimmelbildspiel

Abseits der oben ausgeführten Stellenbeschreibung laßen sich weitere Erkenntnisse zur Zielgruppe aus der Beschreibung des Wimmelbildspiels erkennen. Dieses bildet die Ausbildungswirklichkeit der Typenkunde ab. Das dazugehörige Konzept stammt vom IOSB (Abschnitt 5.1).

Das Wimmelbildspiel besteht aus einer gezeigten Bildszene, auf der bestimmte Zielobjekte zu identifizieren sind. Neben den Zielobjekten befinden sich weitere Störobjekte (Distraktoren) in der Szene, die den Zielobjekten ähnlich sind, und so die Identifikation erschweren (Typenkunde). Das Bild muss innerhalb einer gewissen Zeit bearbeitet werden. Am Ende wird eine Bewertung durchgeführt.

Betrachtet man die Übungsaufgabe aus der Perspektive der Spielertypographien (Abschnitt 3.4), so läßt sie sich dem rechten unteren Quadranten von Bartle's Spielertypen (*Explorer*) bzw. Kim's *Social Action Matrix* einordnen: Ein Bildauswerter erkundet akribisch eine Karte (*explore, view*), um dort relevante Objekte zu finden (*search, find, collect*) und zu annotieren (*curate, review*).

Ausgehend von der Annahme, dass die Typenkunde (das richtige Erkennen und Annotieren von Objekten) ein wichtiger Teilaspekt der Bildauswertung ist, und ein guter Bildauswerter gerne seiner Arbeit nachgeht und die oben genannte Übungsaufgabe mit Bravour meistert, kann gesagt werden, dass ein guter Bildauswerter zum Spektrum der *Explorer* gehört.

Natürlich ist die Arbeit eines Bildauswerters vielschichtiger als diese Betrachtung zeigt. Beispielsweise ist anzunehmen, dass ein Bildauswerter die annotierte Karte mit anderen Bildauswertern und/oder Vorgesetzten besprechen wird, was eine soziale Tätigkeit ist (*collaborate, share, contribute, comment*). Sie bildet aber einen guten ersten Schritt, um mit der Ideenfindung anzufangen.

5.3 Ideenfindung

Kern der Konzeption ist die Ideenfindung. Sie schlägt die Brücke zwischen Problemstellung und Lösung. Am Ende der Ideenfindung ensteht ein Konzept, mit dem Problem gelöst wird.

Bei der Ideenfindung handelt es sich um ein kreatives und iteratives Vorgehen. Wichtig ist hierbei die Tatsache, dass Kreativität einen sozialen Aspekt birgt [102,

S. 65][27]. Aus diesem Grund wurde in der kreativen Phase eng mit der Foschungsgruppe am IOSB zusammengearbeitet, um Ideen zu diskutieren und zu verwerfen, um ein kreatives und iteratives Vorgehen zu gewährleisten.

Zum Finden von Ideen gibt es eine Vielzahl an Möglichkeiten ([78, Tab. 1, S. 1302][77, Tab. 5, S. 227]). Für die Konzeption in dieser Arbeit wurde eine Mischung aus Brainstorming[5], Moodboards, Mindmaps, Spaziergängen und Schlaf benutzt.

Zur Validierung wurde das Konzept als visuelle Diskussionsgrundlage in mehreren Mockups umgesetzt und iterativ diskutiert, um sich auf einen Ansatz für das Game Design zu einigen.

Da Kreativität und Ideenfindung für jeden Menschen unterschiedlich sind, und im Rahmen der Konzeption eine große Anzahl an Kreativartefakten erzeugt wurden, wird an dieser Stelle nur auf die nachfolgende Konzeption verwiesen, in der die Mockups vorgestellt und das Game Design diskutiert wird.

5.4 Game Design

Das Game Design gibt den Bauplan für die Implementierung vor. Es enthält alle Erkenntnisse der vorangegangenen Phasen, auf technische Machbarkeit validiert. Dabei wird wird versucht Hintergründe und Schwierigkeiten bei der Konzeption darzustellen.

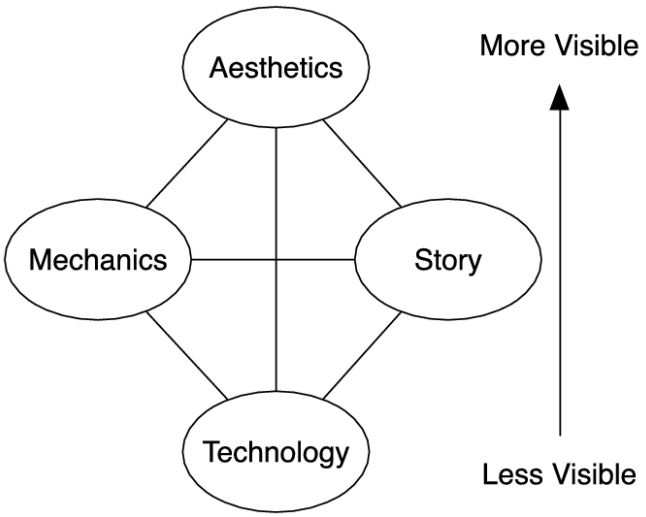


Abbildung 5.1: Die Spielelement-Tetrade. Von [99], Grafik aus [126]

Laut Schell [99, S. 51f] kann das Game Design in vier gleichwertige Spielelemente unterteilt werden — *the element tetrad* (Abbildung 5.1):

- *Mechanics.* Spielmechaniken und Regeln, mit denen der Spieler interagiert.
- *Aestethics.* Das Aussehen des Spiels. Es steht in der Tetrade ganz oben, da dies das erste ist, was der Spieler sehen wird. Spricht ihn das Aussehen nicht an, legt er das Spiel zur Seite. Es kann als *Marketing* gesehen werden.

[5] Sturm im Sinne von „Eine Festung stürmen“

- *Story.* Die Geschichte, die das Spiel erzählt. Davon hängt das Aussehen des Spiels ab.
- *Technology.* Die zu Grunde liegende Technologie, die das Spiel zusammenhält. Sie ist nur im Hintergrund wahrnehmbar — meistens fällt sie nur auf, wenn sie kaputt ist (und der Vertiefungszustand dadurch beeinträchtigt oder gebrochen wird).

Da sich –wie im Schaubild zu sehen– alle vier Elemente gegenseitig beeinflussen, und sie aus einer iterativen Kreativphase hervorgegangen sind, ist es schwierig, ihre Enstehung chronologisch wiederzugeben.

Auch die Parametrisierung wird separat betrachtet, obwohl sie (hautpsächlich, aber nicht nur) zu *Mechanics* gehört und iterativ im Zusammenspiel mit den anderen Elmenenten konzipiert wurde.

Hierbei sei angemerkt, dass der verwendete Begriff „Wimmelbildspiel“ wiederholt diskutiert wurde. Dieser hatte sich während der Konzeption als Spielbezeichnung etabliert, da „spielbasierte Bildauswerteraufgabe zum Erlernen der Typenkunde“ zu lang war.

Hauptkritikpunkt war die Tatsache, dass falsche Erwartungen geweckt werden könnten, da Wimmelbilder „wimmeln“, also chaotisch und voller Leben sind (Abschnitt 3.7), während eine Bildauswerter-Aufgabe eben nicht „wimmeln“ sollte, um die Ausbildungsrealität nicht zu verfälschen. Die aus der Kritik angeregte Diskussion hat folglich ein Mißverständnis bezüglich der Position des Wimmelbildspiels offenbart: Das Wimmelbildspiel –als Spiel– bildet die Ausbildungsrealität in Form eines Spiels ab; es verläßt die Wirklichkeit und agiert in einem *magic circle* — einer ungebundenen Zone jenseits der Realität [99, S. 44]: Das Wimmelspiel darf wimmeln, muss es aber nicht.

Entstanden ist der Begriff des Wimmelbildspiels aus einer Suche nach verwandten Spielen. Dabei ist das Genre der „hidden object“ Spiele und die Wimmelbild-Bücherreihe „Findet Waldo“ aufgekommen. Da sich das Spielprinzip des Wimmelbildspiels leicht von dem der „hidden object“ Spiele unterscheidet (in Perspektive, Form der zu findenden Objekte, Anwendungskontext und Erzählweise), wurde schließlich als Alternative der Begriff Wimmelbildspiel gewählt. Das Wimmelbildspiel und „hidden object“ Spiele sind verwandt, aber nicht identisch. Sie gehören beide zum Genre der Such-, Bilder- und Rätselspiele. Alternative Begriffe sind Wimmelspiel, Bilderrätsel, Bilderspiel oder Bilderpuzzle.

5.4.1 Story

Die Geschichte des Spiels bildet Anwendungskontext und Einsatzgebiet des Spiels ab. Sie ist kurz erzählt: Ein Bildauswerter soll auf einer Bildszene Objekte finden.

Diese Geschichte entspricht nicht der klassichen Vorstellung einer Geschichte im Sinne der AAA-Videospielindustrie, ist aber im Bereich der Mini- und Independent Games nicht unüblich — und auch die Videospiel-Legende „Pong“ kommt ohne große Story aus.

5.4.2 Mechanics

Die Spielbeschreibung des Wimmelbildspiels basiert auf der Spielbeschreibung des IOSB.

Das Wimmelbildspiel hat die folgenden Regeln und Spielmechaniken:

- Das Spiel findet auf einer Karte statt.
- Kartenausschnitt kann manipuliert werden (Bewegung, Zoom, ...).
- Auf der Karte sind Objekte vorhanden, die es zu finden gilt (Zielobjekte).
- Auf der Karte sind Störobjekte (Distraktoren) vorhanden, die den Zielobjekten ähneln und den Spieler ablenken sollen.
- Der Spieler kann die Objekte auf der Karte markieren.
- Der Spieler kann Fehler machen, in dem er die falschen Objekte markiert.
- Es existiert eine Vorgabe, welche Objekte der Spieler finden soll.
- Die Karte ist in einem Zeitraum zu bearbeiten.
- Die Karte wird an einem Stück bearbeitet.
- Am Ende erfolgt eine Bewertung.

Abbildung 5.2 zeigt den dazugehörigen Entwurf (Mockups). In der Szene sind verschiedene Objekte (Raumschiffe *u.Ä.*) auf einer Karte positioniert (a). Am linken Rand befindet sich die User Interface (UI oder Benutzerschnittstelle). Sie zeigt die zu findenden Objekte an. Ebenfalls gezeigt wird die verbleibende Bearbeitungszeit. Am rechten oberen Bildschirmrand befindet sich eine Punkteanzeige. Macht ein Spieler einen Fehler (b), werden ihm Punkte abgezogen und eine Warnung angezeigt. Tippt er richtig (c) werden ihm Punkte gutgeschrieben und die UI aktualisiert.

Dabei fällt auf, dass Fehler rot, und richtige Treffer blau markiert werden. Diese Farben wurden in Anbetracht des häufigen Vorkommens einer Rot-Grün-Schwäche oder Farbenblindheit bei Männern (9% der Bevölkerung) und des hohen Anteils an Männern an der Zielgruppe gewählt.[25] — In jedem Projekt sollte die Frage nach Bedürfnissen der Zielgruppe hinsichtlich *Accessibiliy*[6] gestellt und beantwortet werden. Gleichzeitig kann nicht ausgeschlossen werden, dass eine Rot-Grün-Schwäche/Farbenblindheit ein Ausschlusskriterium für den Beruf eines Bildauswerters ist. Die Stellenausschreibung der Bundeswehr (**??**) hat darauf keine Antwort gegeben.

5.4.3 Aestethics

Farbgebung, Form und Design des Spiels werden von der Zielgruppe bestimmt. Da das Spiel in erster Linie als *Testbed* eingesetzt werden soll, und kein Budget oder Spielinhalte vorhanden sind, müssen geeignete Spielinhalte gesucht werden, wobei

[6] Barrierefreiheit

(a) Die Spielansicht des Wimmelbildspiels

(b) Es wurde ein Fehler gemacht

(c) Der Spieler hat richtig getippt

Abbildung 5.2: Entwurf des Wimmelbildspiel

nur in begrenztem Maße Rücksicht auf die Bedürfnisse der Zielgruppe genommen werden kann.

Zuerst werden die für das Spiel benötigten Spielinhalte aufgelistet: Das Spiel besteht aus einer Spielszene (Karte), auf der sich Objekte (Zielobjekte und Distraktoren) befinden. Weiterhin gehört eine UI zur Szene, die die zu findenden Objekte und weitere Spielinformationen (Bearbeitungszeit, Punkte) anzeigt.

Entsprechend der *Story* nimmt die Spielszene die Perspektive einer Luft- oder Satellitenbildaufnahme an. Dabei handelt es sich um eine Aufnahme von oben, also um ein Senkrecht- oder Geneigtbild (Abschnitt 3.5).

Perspektive

Für Spielinhalte gibt es drei Perspektiven: 2D, 3D und Isometrisch (2.5D). Sie alle eignen sich um die gewünschte Perspektive der Szene nachzubilden. Dabei zeichnet sich ab, dass der Aufwand der Entwicklung mit der Zunahme der Dimensionen steigt (2D $\rightarrow$ 2.5D $\rightarrow$ 3D). So finden sich (im kostenlosen Bereich) sehr viel mehr zweidimensionale Spielinhalte, als es 3D-Inhalte gibt.

- 3D-Inhalte (meist 3D-Modelle) sind sehr detailliert und versatil einsetzbar sind. Sie eignen sich für jede erdenkliche Perspektive und gehen oft in Richtung Fotorealismus. Gleichzeitig sind 3D-Modelle unhandlich und nicht ohne Expertise modifizierbar. Auch haben 3D-Modelle oft eine große Dateigröße und werde nicht von jeder Game Engine unterstützt.
- 2D-Inhalte (*Sprites*) sind handlicher: Sie haben eine geringere Dateigröße, können einfach nachbearbeitet oder verändert werden, und werden von jeder Game Engine unterstützt. 2D-Inhalte eignen sich hauptsächlich für zweidimensionale Perspektiven –meist aus der Vogelperspektive oder von der Seite– und sind deshalb oft abstrakt. Auch ist es einfacher 2D-Animationen zu kreieren.
- 2.5D-Inhalte (Isometrisch) erzeugen durch ihre clevere Perspektive eine Illusion von Räumlichkeit, sind aber ansonsten 2D-Bilder.

Da sich alle drei Perspektiven für das Wimmelbildspiel eignen, wurde der Weg des geringesten Wiederstandes gegangen, und eine 2D-basierte Perspektive gewählt. Damit auch der Bereich der Geneigtbilder abgedeckt und auch Verdeckungen und Überlappungen von Objekten im Spiel untersucht werden können, wurde sich für eine isometrische Perspektive entschieden.

Sprites

Die benötigten Bildmaterialien müssen

- sich für die gesuchte Perspektive der Spielszene eignen.
- sich als Stör- und Zielobjekte eignen.
- eine permissive Lizenz haben (damit sie überhaupt benutzt werden können)

Abbildung 5.3: Kenney Isometric Vehicles Sample Preview

- ohne Kosten sein (da das Budget 0 € beträgt).

Dabei wird darauf geachtet, dass sie

- qualitativ hochwertig sind.
- kohärent sind und einheitlich aussehen. Bestenfalls stammen sie alle aus einem einzigen Set an Bilddateien.
- für die Zielgruppe ansprechend wirken.

Diese Anforderungen werden durch *Sprites* der Kenney-Serie gedeckt Abbildung 5.3, da sie

- eine hohe Qualität besitzen.
- sich in der Öffentlichen Domäne (CC0 1.0 Universal) befinden, und somit sorgenfrei benutzt und verändert werden können.
- in hoher Varianz die angestrebte Domäne abzeichnen (Stadt, Straße, Landschaft, Fahrzeuge).
- in gesuchter Form (isometrisch) die angestrebte Domäne abzeichnen.

Während der Konzeption war angedacht, dem Spiel –angelehnt an den „großen Bruder“ Lost Earth 2307 mit gleicher Zielgruppe [115, S. 28]– ein buntes, futuristisches Aussehen zu geben. Dies zeigt sich auch in den Mockups (Abbildung 5.4).

In der finalen Diskussion wurde diese Überlegung in Anbetracht des Anwendungskontextes und der heterogenen Zielgruppe zu Gunsten eines kontemporären, sachlichen und neutralen Stils geändert. An Stelle von Raumschiffen *u.Ä.* wird –basierend auf dem in Abbildung 5.3 gezeigten Sprite-Pack– eine auf der Erde angesiedelte Landschaft mit Städte, Straßen und Fahrzeugen gezeigt.

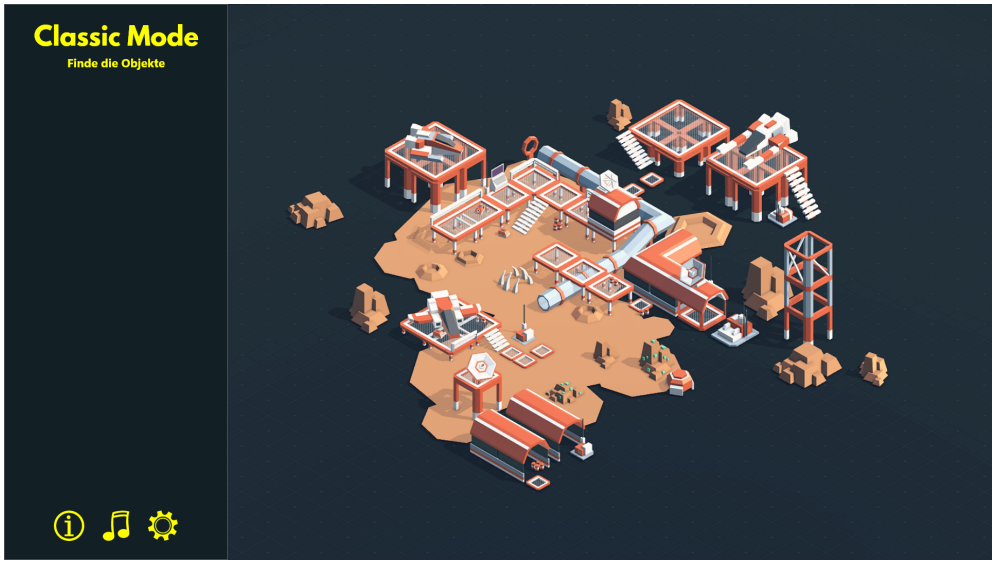


Abbildung 5.4: Futuristisches Wimmelbild

Sound

Ein weiterer Faktor der *Aestethics* ist das Ambiente und die Musik. Da das Spiel aber in erster Linie als *Testbed* eingesetzt wird, und keine geeigneten Musikstücke gefunden werden konnten, wurde auf den Einsatz von Musik verzichtet[7].

5.4.4 Technology

Die Frage der Technologie ist immer auch eine Frage der Machbarkeit — kann das Spiel in der konzipierten Form erstellt werden, oder müssen Kompromisse eingegangen werden? (Das Eingehen von Kompromissen muss nicht zwingend schlecht für das Spiel sein, da sich dabei ungeahnte Möglichkeiten für das Spiel aufzeigen können [99, S. 92]).

Anforderung an das Spiel ist, dass es unter Verwendung von Web-Technologie erstellt wird, um in die vorhandene Test-Struktur eingebunden werden zu können.

Web-Technologie bezeichnet in diesem Fall die Verwendung von HTML, CSS und Javascript, die zum Aufbau und Erstellen von Webseiten und Web-Apps genutzt werden.

Die Verpflichtung zur Nutzung von Web-Technologien hat Auswirkungen auf die Projektarchitektur, beispielsweise auf die Wahl der Game Engine.

Game Engine

Da das Rad nicht neu erfunden werden muss, und die für das Projekt zur Verfügung stehende Zeit limitiert ist, empfiehlt sich die Verwendung einer Game Engine.

[7] Die während der Ideenfindung gefundenen Stücke wurden alle nach Diskussion als unpassend abgelehnt.

Eine Game Engine unterstützt die Spiele-Entwicklung, in dem sie in Spielen häufig benutzte Funktionen zur Verfügung stellt und dem Entwickler die *low-level* Programmierung abnimmt — dazu gehören u.a. Grafik, Physik, Kamera, Sound und Steuerung [132]. Dies ermöglicht es, dass sich ein Entwickler voll auf die wertschöpfende Handlung –das Programmieren des eigentlichen Spiels– konzentrieren kann, da die zuvor genannten Funktionen nicht mehr neu in das Spiel eingebaut werden müssen.

Dabei geht es nicht darum, das beste, sondern ein geeignetes Framework zu finden, mit dem das Spiel realisiert werden kann.

Gundermann (2016) [46] schlägt –basierend auf einer *Game Engine Analysis*[116] und im Rahmen eines von den Anforderungen ähnlichen Projekts– die Kombination aus Phaser und React für die Umsetzung eines isometrischen, web-basierten Spiels vor.

- Phaser ist ein *„fun, free and fast 2D game framework for making HTML5 games for desktop and mobile web browsers, supporting Canvas and WebGL rendering“* [28].
- React ist *„a declarative, efficient, and flexible JavaScript library for building user interfaces.“* [36]

Aber eignen sich Phaser und React auch für dieses Projekt? Es empfiehlt sich Phaser und React an Hand der in der *Game Engine Analysis* genannten Kriterien –Lizenz, Dokumentation, Entwicklung, Stabilität und Funktionsumfang– auf Aktualität und Relevanz zu prüfen. Weiterhin müssen sie auch unter Gesichtspunkten der Bestandsaufnahme betrachtet werden, um eine Eignung für das Projekt feststellen zu können.

Lizenz. Sowohl Phaser als auch React unterliegen der MIT License. Sie sind dementsprechend frei, permissiv und dürfen ohne Einschränkungen verwendet werden. Dies ist wichtig, da zum einen kein Budget für eine teure Engine zur Verfügung steht, und zum anderen nicht ausgeschlossen werden kann, dass das Projekt (oder Teile davon) in einer kommerziellen Anwendung verwendet werden. Aus diesem Grund sind auch die so genannten „copy left“-Lizenzen [131] wie GPL problematisch.

Dokumentation. Sowohl Phaser als auch React sind gut dokumentiert. Es gibt eine Reihen an Online-Ressourcen, Anleitungen und Entwicklungen. Phaser liefert 80 mio. Google-Suchergebnisse, React 800+ mio. Dafür besticht Phaser mit einer Sammlung an *hands on* Beispielen samt Quellcode, um den schnellen Einstieg in die Programmierung zu erlauben.

Entwicklung. Sowohl Phaser als auch React verfügen über eine aktive Entwickler-Community. Während React als Karriere-Pfad[8] beinahe 300.000 Fragen auf der Programmier-Platform StackOverflow aufweist[9], bietet Phaser eine kleine, enge

[8] Job-Titel: React Developer
[9] „Questions tagged [reactjs]“

und sehr aktive Community, in der die Ersteller des Frameworks regelmäßig aktiv sind, um Fragen zu beantworten.

Stabilität/Maturität. Sowohl Phaser als auch React wurden seit der *Game Engine Analysis* aktiv weiterentwickelt [10].

Funktionsumfang. Phaser als Game Engine bietet alle Funktionen, die von einer Game Engine erwartet werden. Da Phaser eine 2D-Engine ist, fehlt Unterstützung für 3D. React ist keine Game Engine, komplementiert aber Schwächen von Phaser hinsichtlich Bau einer UI.

Es zeigt sich, dass sowohl Phaser als auch React aktuell und weiterhin relevant sind.

Eignen sich Phaser und React für das Vorhaben? Beide erfüllen die einzige Anforderung — die Verwendung von Web-Technologie. Weiterhin passen sie mit ihren Lizenzen in die Einschränkungen des Projekts. Dementsprechend spricht nichts gegen ihre Benutzung.

Einziger Punkt der Kritik ist die fehlende Unterstützung für 3D-Modelle. Da aber keine 3D-Materialien verwendet werden, bildet sich daraus kein Hindernis.

Alternativen. Alternativ bietet sich der Einsatz von Cross-Platform Game-Engines (Unreal Engine/Unity 3D) an. Diese machen aber unter Berücksichtigung der bestehenden Expertise in der Forschungsgruppe und der Anforderung zur Benutzung von Web-Technologie wenig Sinn.

Mobile

Neben der Game Engine stellt sich auch die Frage nach der Platform, auf der das Spiel laufen soll. Die Verwendung von Web-Technologie ermöglicht das Spielen auf allen Platformen, da Browser auf allen Endgeräten verfügbar sind. Gleichzeitig unterscheiden sich die Platformen voneinandern, besonders in Hinblick auf die Bildschirmgröße — Tablets und vor allem Smartphones sind phsysisch kleiner als ein Desktop-Monitor, und erfordern ein eigenes Design.

Diese Frage beantwortet sich durch Betrachtung der existierenden Laborumgebung für das *Testbed*; Sie besteht aus regulären Desktop-PCs und Monitoren, da auch die Bildauswertung am Computer durchgeführt wird. Aus diesem Grund wird das Spiel für den Desktop konzipiert — Dies ist auch an den Mockups (Abbildung 5.2) zu erkennen.. Es ist aber anzunehmen, dass sich das Desktop-Design auch für Tablets eignet.

5.4.5 Parametrisierung des Wimmelbildspiels

Besonderer Fokus liegt auf der Parametrisierung bzw. Parametrisierbarkeit des Wimmelbildspiels. Wenn man sich die drei W-Fragen (*When, what and how to adapt* [115, S. 23f]) ins Gedächtnis ruft, dann beschreibt die Parametrisierung

[10] Zum Zeitpunkt dieser Arbeit steht React auf Version 17.0.2, während Phaser 3 mit Version 3.54 abgeschlossen ist und die Arbeit an Phaser 4 begonnen hat [89].

das *What to adapt* — Aufbau und Vorkommen der *Hebel*, mit denen das Spiel adaptiert wird. Auf das Game Design bezogen, findet die Parametrisierung auf Ebene der *Mechanics* statt.

Die Parametrisierung des Wimmelbildspiels wird durch die DDA-Parameter (*Adaptivity Response*, Siehe Unterabschnitt 3.2.4) bestimmt. Sie beschreiben, wie sich das Spiel verändern soll: Leichter, schwieriger, mehr Hilfe, weniger Hilfe, Störeffekte, Herausforderungen. *„Potentially, all components that are considered at game development can become adaptive.“* [67, S. 4] — dementsprechend gilt es basierend auf *Mechanics* und Anwendungskontext geeignete Komponenten zu finden.

Die *Mechanics* geben Ausschluss über die Parametrisierung: Da das Spiel aus einer einzelnen Szene besteht, und diese sich nach Spielbeginn nicht verändert, muss die Anpassung des Spiels vor Spielbeginn feststehen. Für das adaptive Wimmelbildspiel bedeutet das, dass die Karte für jeden Spieler individuell erstellt und Zielobjekte und Distraktoren neu platziert werden müssen.

Hierfür bietet sich die PCG an [138]. Mit ihr müssen Spielinhalte nicht händisch erstellt und platziert werden, sondern können algorithmisch erzeugt werden. Dies erlaubt es dem adaptiven System, autonom Entscheidungen zu fällen , und Miminmalanpassungen vorzunehmen, ohne, dass ein Mensch für jeden Sonderfall entsprechende Inhalte oder Schablonen angelegt haben muss.

Da die Spielszene *vor* Spielbeginn erstellt wird, handelt es sich um eine *offline generation* [67, S. 6].

Schwierigkeit in der Typenkunde entsteht aber nicht nur durch die Platzierung der Objekte, sondern auch durch ihre Unterscheidung. Deshalb muss die Parametrisierung des Wimmelbildspiels auch den Anwendungskontext Bildauswertung berücksichtigen, und Möglichkeiten zur Anpassung der Zielobjekte und Distraktoren bereitstellen. Geeignete Charakteristiken von Objekten dafür finden sich in Lillesand et al. [64, S. 195ff] (Siehe Abschnitt 3.5): Größe, Form, Farbe, Muster, Textur, Schatten, Platzierung, Auflösung, Lokalisierung, Assoziation, *et cetera*.

Abseits der oben genannten Spielekomponenten (Spielfeld, Objekte) gibt es eine dritte Komponente, die adaptiv sein kann — die UI. Sie zeigt Informationen zum Spiel. Hier bietet sich das taktische Ausblenden oder Hinzufügen von Informationen als Schwierigkeitsmechanik an.

Da es drei anpassbare Spielkomponenten und drei DDA-Parameter (*Performance*, *Assistance* und *Skill*) gibt, wurde versucht jedem DDA-Parameter einen Komponentenbereich zuzuweisen, der angepasst wird:

- *Performance* gibt Ausblick auf die aktuelle Leistung des Benutzers. Deshalb wurde ihr die Generierung der Spielszene zugeordnet: Die Größe der Karte hat einen direkten Einfluss auf das Abschneiden des Spielers. Dabei gilt: Je größer (und unübersichtlicher) die Karte, desto höher die Schwierigkeit.
- *Assistance* spiegelt das Niveau der Hilfestellung wieder. Hierbei bietet sich die UI als Zuordnung an, da sie keine *Performance*- oder *Skill*-relevanten Rückschlüsse erlaubt, und auch während des aktiven Spiels angepasst werden

könnte.

- *Skill* gibt die *Performance* über einen längeren Zeitraum an. Hierfür wurden die bildauswertungsspezifischen Anpassungen der Zielobjekte und Distraktoren gewählt — es ist anzunehmen, dass die Änderung der Kartengröße eine Frage der Geschwindigkeit und der Suchstrategie ist, während das Unterscheiden von Objekten einen höheren und längfristigeren Lernaufwand erfordert. Eine lernzielspezifische Zuordnung der Parameter ist aber schwierig, da *Skill* und *Performance* aufeinander aufbauen. Dies ist eine Frage des *How to adapt* und jenseits dieser Arbeit.

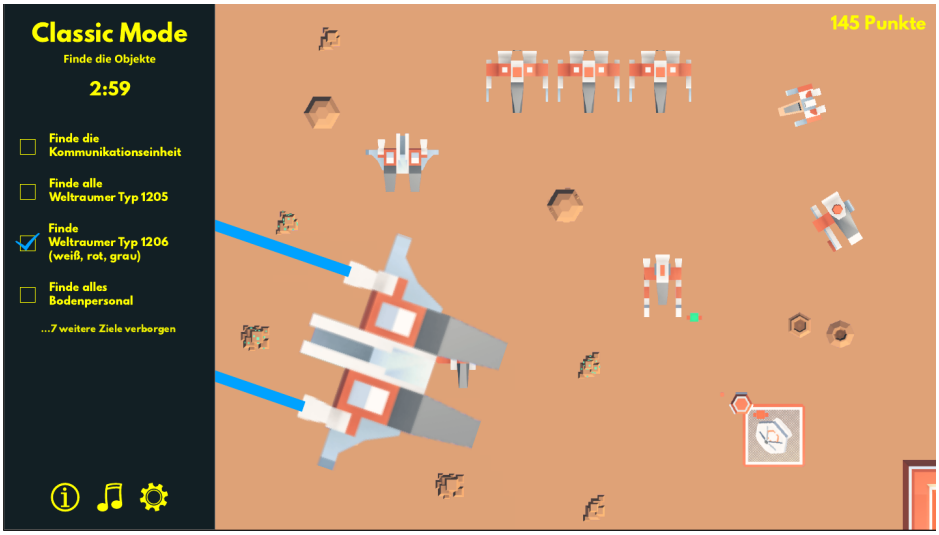
Ausgehend von dieser Zuordnung wird das Spielkonzept konkretisiert: Die Bildszene des Wimmelbildspiel wird zu einem Zeitpunkt Null erstellt. Dazu gehört das prozedurale Generieren der *Game World* [67, S. 6] bzw. Karte (Landschaft, Straßen, Städte) und die Platzierung der Objekte (Fahrzeuge). Die Objekte unterscheiden sich in Form, Größe, Farbe, und Orientierung, stellvertretend für die von Lillesand et al. [64, S. 195ff] genannten Charakteristiken. Während des Spiels werden Störeffekte angezeigt.

Mögliche Störeffekte wurden in einer eigenen Kreativphase gesucht und konzipiert. Sie umfassen...

- animierte Distraktoren, die über den Bildschirm fliegen (Abbildung 5.5a)
- die automatische Rotation des Spielfelds (5.5b)
- Wettereffekte: Regen, Wolken, Schnee, ... (5.5c)
- einen Tag-Nacht-Zyklus (5.6a)
- die Veränderung der Jahreszeiten (5.6b)
- verwackelte Kamera
- Bildstörungen (Rauschen, Verpixelung, grafische Artefakte, Zerrungen, Dopplungen (5.6c))
- einen 16-Bit Farbmodus
- die Verkleinerung des Sichtfeldes
- eine Drehung

Auf Grund technische Machbarkeit, verfügbare Entwicklungszeit und Budget musste die Auswahl an möglichen Störeffekten aber stark reduziert werden — am Ende wurden nur die animierten Distraktoren und der Tag-Nacht-Zyklus gewählt.

Die UI zeigt spielrelevante Informationen wie verbleibende Spielzeit, Punkte und zu suchende Zielobjekte an. Diese Informationen können graphisch oder textuell sein — dies ist ebenfalls in den Mockups zu sehen (Vergleiche Abbildung 5.5a und Abbildung 5.5b). Auch werden je nach Hilfestellungslevel keine Objekte, alle Objekte oder nur die Zielobjekte graphisch hervorgehoben, um ihre Entdeckung zu erleichtern.

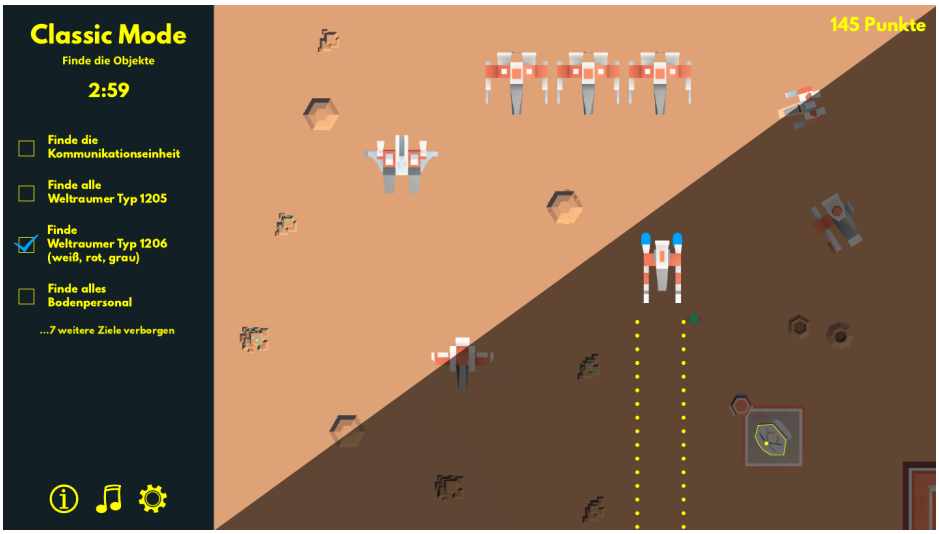


(a) Animierte Distraktoren

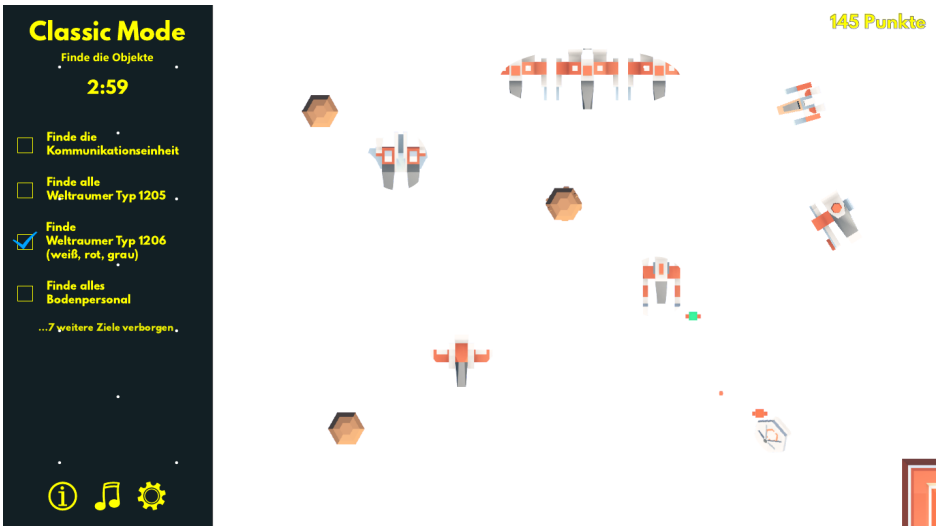
(b) Automatische Rotation der Karte

(c) Wettereffekte

Abbildung 5.5: Schwierigkeitsanpassungen



(a) Tag-Nacht-Zyklus



(b) Jahreszeiten

(c) Dopplungen

Abbildung 5.6: Schwierigkeitsanpassungen (Fortgesetzt)

Dabei zeigt sich, dass die Parametrisierung trotz PCG nicht ohne menschgemachte Inhalte auskommt: Die oben beschriebene Generierung basiert auf dem Vorhandensein von Sprites, aus denen die Szene aufgebaut wird. Auch die Zielobjekte und Distraktoren werden durch Sprites dargestellt. Dies deckt sich mit den Erfahrungen der Spieleindustrie — große Spiele (beispielsweise Skyrim[Beleg!]) werden durch eine Mischung aus PCG und menschgemachter Details erstellt. Hierin liegt auch die Herausforderung: Je mehr Kontrolle der Designer über die Generierung hat, desto flexibler wird sie [67, S. 8].

Eine weitere Komponente zur Be- und Entschleunigung ist die Musik. Beispielsweise könnte im Spiel Horrorspiel-Musik eingesetzt werden, um den Spieler auf den Beinen zu halten, oder Fahrstuhlmusik, um ihn zu entspannen.

Kapitel 6

Implementierung

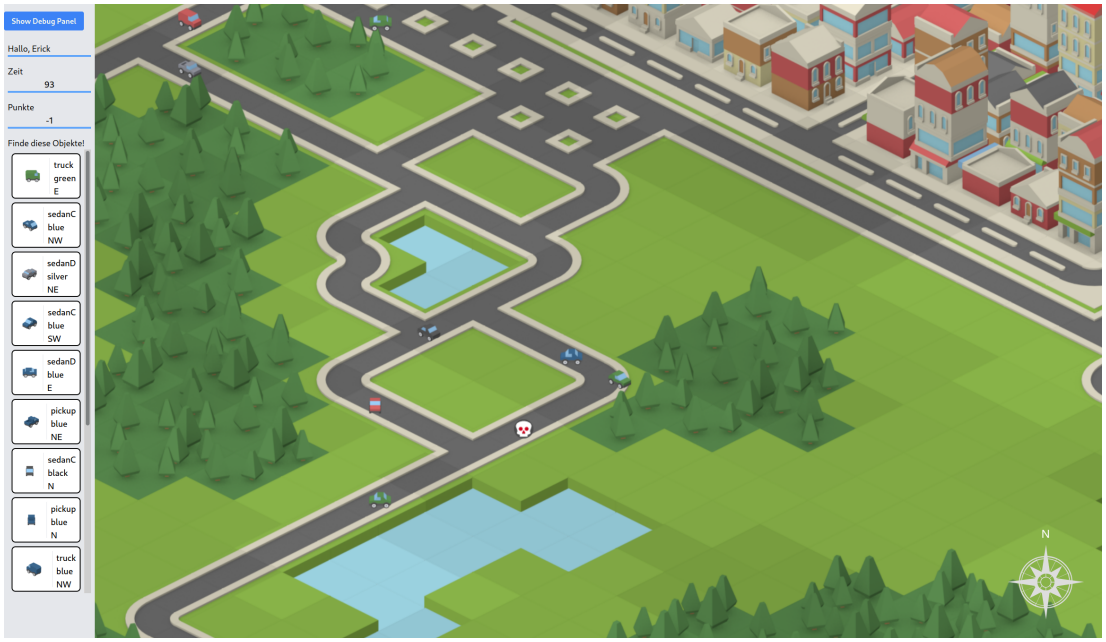


Abbildung 6.1: Das Wimmelbildspiel

Die Implemtierung des Wimmelbildspiels (Abbildung 6.1) erfolgt anhand des erstellten Konzepts. Dieses wird in einem agilen und iterativen Verfahren umgesetzt. Da das Entwicklerteam nur aus einer Person besteht[1], wurde dazu eine SCRUM-Variante in reduzierter Form gewählt. Dabei wird das Projekt in zweiwöchentlichen Schritten (*Sprints*) geplant und mit den externen Stakeholdern besprochen, so dass immer ein lauffähiger Prototyp zur Diskussion steht.

Dieses Verfahren wurde gewählt, da die Erfahrung zeigt, dass während der Entwicklung größerer Projekten häufig unbekannte Unabwägbarkeiten und unvorhergesehene Schwierigkeiten auftauchen, die während der Konzeption und Projektplanung nicht einsehbar waren.

[1] Sofern man bei einer Person überhaupt von einem Team sprechen kann

Um handelbares Wissen zu vermitteln und so die zukünftige Entwicklung adaptiver Spiele zu unterstützen, werden eigene wichtige Erfahrungen und *best practices* geteilt. Das fertige Programm wird im Kapitel Benutzertest (Kapitel 7) evaluiert und im Kapitel Diskussion (Kapitel 8) besprochen.

Die Implementierung erfolgt mit Web-Technologien. Als Game Engine wird Phaser verwendet, als Hilfsmittel für die UI React. Die verwendete Programmiersprache ist TypeScript (TS) (bzw. für React TSX). Getestet wurde mit Jest.

Im folgenden wird die Implementierung der einzelnen Komponenten besprochen. Dazu gehören

- Architektur des Spiels
- Generierung der Spielwelt
- Platzierung der Fahrzeuge
- Einrichten der Kamera und Steuerung
- Anzeige der UI
- Parametrisierung

6.1 Vorarbeiten

Bevor das Spiel implementiert werden kann, müssen gewisse Vorarbeiten erledigt werden. Dazu gehören das Aufsetzen des Web-Servers und das Vorbereiten der Spielinhalte (*Sprites*).

6.1.1 Setup

Moderne Browser blockieren aus Sicherheitsgründen den Zugriff auf Dateien, die nicht von der selben Quelle stammen [73]:

> The same-origin policy is a critical security mechanism that restricts how a document or script loaded from one origin can interact with a resource from another origin. It helps isolate potentially malicious documents, reducing possible attack vectors.

Dies macht die Verwendung eines eigenen, lokalen Entwicklungsservers notwendig [88].

Auch mussten dazu npm und webpack eingerichtet werden, da TS in einem *build process* in JavaScript (JS) umgewandelt werden muss.

6.1.2 Sprite Atlas

Die während der Konzeption (Unterunterabschnitt 5.4.3) gewählten *Sprites* müssen in eine für Phaser verwendbares Format gebracht werden. Dies geschieht in Form eines *Sprite Atlas*. Dieser besteht aus einem *Spritesheet*, auf dem alle Spielinhalte (*Sprites*) zusammengefasst sind (Abbildung 6.2), und einer Liste (JavaScript

Object Notation (JSON)-Format), in der notiert wird, an welcher Position sich welcher *Sprite* befindet, und wie dieser *Sprite* heißt.

Dies hat den Vorteil, dass im Spiel nicht viele einzelne Bilder aufgerufen werden müssen, sondern nur ein einziges Bild geladen und wiederverwendet wird.

Die Aufbereitung der Sprites hat unerwartet viel Arbeit in Anspruch genommen, da die Sprites aus mehreren Quellen stammen und unterschiedliche Namensgebungen hatten: So bezeichnen *carRed1_04* und *carRed2_05* beide einen *Sprite* mit Fahrtrichtung Norden und müssen einheitlich umbenannt werden. Auch mussten unbenötigte Sprites entfernt werden, damit im Spiel nur die Sprites vorhanden sind, die tatsächlich benutzt werden. Dazu mussten alle ca. 1000 Sprites einzeln angeschaut werden. Als besonders problematisch haben sich dabei die Fahrzeuge herausgestellt, da die dazugehörigen Bilder sehr klein sind und geringfügige Unterschiede nur unter einer großen Zoomstufe gesehen werden konnten.

Für die Erstellung der Spritesheets hat sich ein Ordner-basiertes Vorgehen als optimal herausgestellt. Dabei werden die Sprites ihrem Anwendungskontext entsprechend in Ordner sortiert. Diese Ordern werden dann zu einem Spritesheet gepackt, wobei jedes Sprite seinen Ordnernamen als Präfix bekommte. Dies verhindert, dass alle Sprites in einem einzigen Ordner zusammengeworfern werden müssen. Dadurch können die Sprites dynamisch umbenannt oder ergänzt werden. Auch wird so die Verwendung generischer Dateinamen möglich, ohne, dass es zu Dopplungen kommt. Die Bilder heißen alle ähnlich, werden aber durch das ordnerbasierte Vorgehen zu sinnvollen Namen zusammengesetzt:

- *tile/forest/000.png* → *tile_forest_000.png*
- *tile/forest/001.png* → *tile_forest_001.png*
- *tile/forest/002.png* → *tile_forest_002.png*
- *car/red/N/000.png* → *car_red_N_000.png*
- *car/red/N/001.png* → *car_red_N_001.png*

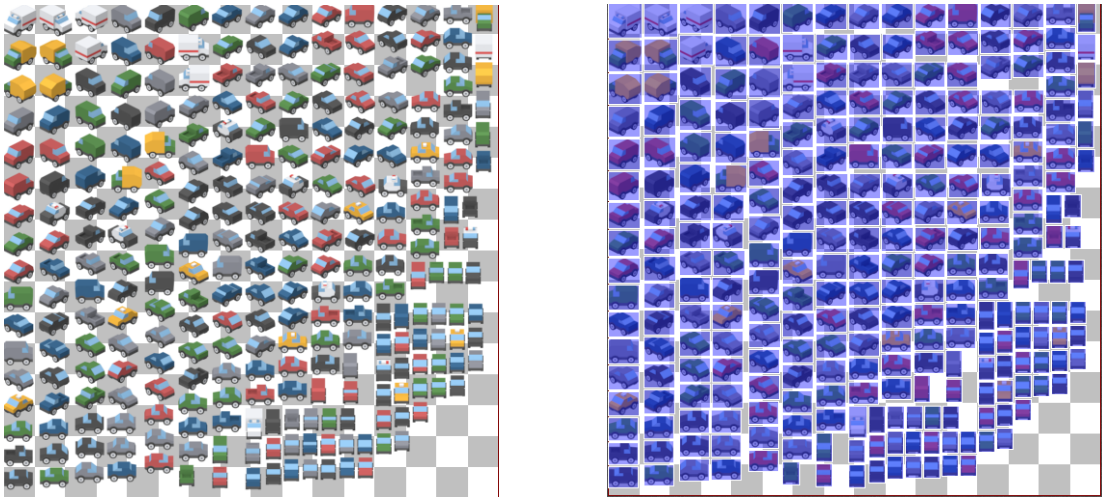
Um die verschiedene Einsatzgebiete der Sprites auseinanderzuhalten, wurden am Ende drei Spritesheets verwendet:

- Prozedruale Generierung: Landschaft & Gebäude
- Fahrzeuge
- Störeffekte. Im wesentlichen nur Raumschiffe. Als *Easteregg* waren Wildgänse geplant[2], dieses Feature wurde aber mangels geeigneter Sprites nicht umgesetzt.

6.2 Spielarchitektur

Das Spiel ist in drei Bildschirme unterteilt aufgebaut:

[2]Eine Hommage an den Roman „Die wunderbare Reise des kleinen Nils Holgersson mit den Wildgänsen“ von Selma Lagerlöf



(a) Spritesheet

(b) Spritesheet mit eingezeichneten Umrandungen

Abbildung 6.2: Spritesheet für die im Spiel verwendeten Fahrzeuge

- Startbildschirm mit Parametrisierungsmöglichkeiten (Abbildung 6.13),
- Spielbildschirm mit Wimmelbildspiel und UI, *und*
- Ergebnisbildschirm mit Spielergebnissen.

Da React für die UI verantwortlich ist, wird das Programm als React-App erstellt. Dabei fungiert das Wimmelbildspiel als Hauptkomponente der App: Alle Programmlogik erfolgt im Spiel, während React nur für das Präsentieren und Aktualisieren der UI verantwortlich ist. Auch die Generierung erfolgt im Spiel.

Der Startbildschirm enthält Parametrisierungsmöglichkeiten, um die Generierung des Spiels testen zu können. Alle drei DDA-Parameter (*Performance, Assistance, Skill*) können hier eingestellt, und so die Funktionsweise eines adaptiven Systems simuliert werden (*Select* im Adaptivitätszyklus).

Der Ergebnisbildschirm zeigt das Spielergebnis. Dieses Ergebnis würde im adaptiven System zur Weiterarbeit aufgenommen werden (*Capture* im Adaptivitätszyklus).

Die drei Bildschirme bilden eine Schleifen: Start → Spiel → Ergebnis → Start.

6.3 Generierung der Spielwelt

Die Spielwelt wird prozedural generiert. Es gibt verschiedene Möglichkeiten zur Umsetzung der prozeduralen Generierung. In dieser Arbeit wurde das *tiling* gewählt, eine weit verbreitete Generierungmethode die sich gut für zweidimensionale Karten eigenet.

Dabei wird die Karte aus viereckigen Kacheln (*tiles*) zusammengesetzt — sie bilden die kleinste Maßeinheit im Spiel. Jede Kachel bildet ein *sprite* ab, entspricht also einer Textur. Weiterhin werden mehrere Kacheln zu einem Block (*Chunk*)

gefasst. Im Spiel ist dafür eine Blockgröße von 20 × 20 Kacheln gewählt. Blöcke erleichtern die Handhabung –es ist einfacher von einem Block zu reden, als von 400 Kacheln– und können für die erweiterte Kartengenerierung verwendet werden (*Chunking*), beispielsweise um Blöcke aus dem Arbeitsspeicher zu entfernen, wenn sie nicht mehr zu sehen sind. Kacheln und Blöcke sind in Abbildung 6.3 zu sehen: Eine generierte Szene aus 60 × 60 (3600) Kacheln ist in neun Blöcke unterteilt.

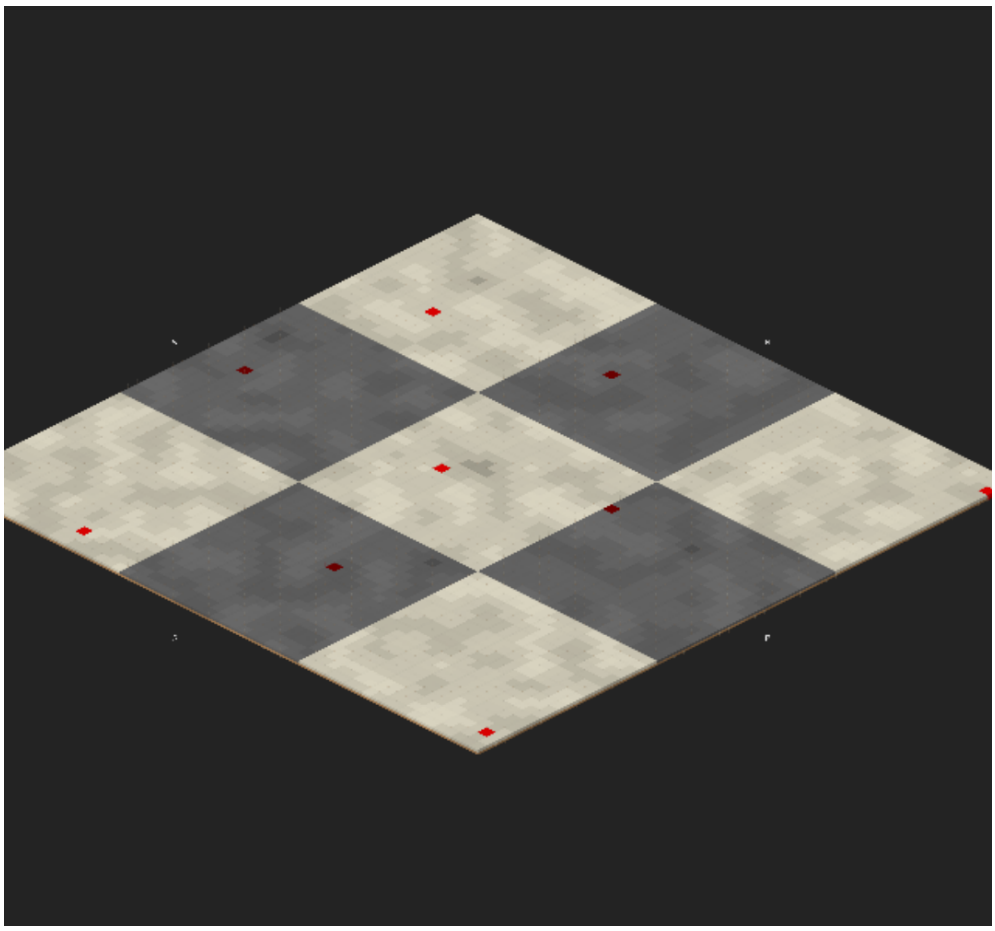


Abbildung 6.3: 3600 Kacheln in neun Blöcken

Damit jeder Kachel ein individuelles Bild zugeordnet werden kann –beispielsweise um sie rot einzufärben, wie in Abbildung 6.3 zu sehen–, müssen sie identifizierbar sein. Deshalb liegt der Szene ein einfaches, kartesisches Koordinatensystem zu Grunde: Jede Kachel hat eine Position (X, Y).

Schaut man sich Abbildung 6.3 noch einmal an, fällt auf, dass die Szene gedreht ist. Dies ist der isometrischen Perspektive geschuldet. Diese erfordert auch die Einführung eines zweiten Koordinatensystems, des isometrischen Koordinatensystems.

Tatsächlich wird im Spiel noch ein drittes Koordinatensystem –das tatsächliche Koordinatensystem; die Screen-Koordinaten– als Zwischenschritt benötigt, damit die Abstände zwischen den Kacheln gewahrt wird. Dieses entspricht dem Kartesischen, außer, dass jede Koordinate mit der Größe der Kacheln multipliziert wird, um das Seitenverhältnis der dazugehörigen Bild zu wahren. Diese Größe

ist im Spiel auf den Wert 64 gesetzt. Die vierte Kachel in der erseten Reihe mit der kartesischen Position (1, 4) hat somit eine tatsächliche Position von (64, 256). Dieses kartesische Koordinatensystem wird für die Umwandlung der Kacheln in das isometrische System gebraucht. Um Verwechslungen zu vermeiden und die Handhabung und das Debugging zu vereinfachen wird das erste, einfache kartesische System als ID der Kacheln genommen, während die beiden letztgenannteren Koordinatensystem für die Platzierung der Kacheln im Spiel zuständig sind.

Zusammengefasst hat eine Kachel drei verschiedene Koordinaten:

- *ID.* Beschreibt die einfache kartesische Position → (1, 4).
- *Kartesische Koordinaten.* Die Position der Kachel ohne isometrische Projektion unter Berücksichtigung der Kachelgröße → (64, 256)
- *Isometrische Koordinaten.* Finale isometrische Position der Kachel → (-192, 160)

Hierbei wird auch der Nutzen der ID bei der Fehlersuche ersichtlich: Es ist weitaus einfacher eine Nachbarschaft der Kacheln (1,1) und (2,1) zu erfassen, als eine Nachbarschaft von (64, 64) und (128, 64) bzw. (0, 64) und (64, 96).

Im Gegensatz zu der in der Implementierung gewählten Variante eines globalen Addressraums der IDs, wäre es auch möglich gewesen, die Koordinaten Block-relevant zu gestalten, ähnlich einer IP-Addresse. Dabei wird jeder Kacheln ihr zugehöriger Block in der Form *BlockID.KachelID* als Präfix zugeteilt: Eine Kachel mit einer ID 401 könnte somit in der Form 1.1 beschrieben werden[3].

Die Verwendung der drei Koordinatensystemen entstammt der Verlegenheit, dass zu Beginn der Entwicklung auf die Phaser-eigene Karten- und Koordinatenverwaltung gesetzt wurde (*Tilemaps*), diese sich aber nicht in der gewünschten Form prozedural generieren lassen. Eine *Tilemap* ermöglicht es, die Karte in einer Kurzform (Welches Bild an welcher Stelle) zu beschreiben, und die Game Engine die Arbeit abnehmen zu lassen — namentlich das Heraussuchen der Bilder, das Erstellen und Positionieren der Kacheln, und ihre logische Verknüpfung.

Dementsprechend musste ein eigenes Kartensystem aufgesetzt werden.

Zum Ende der Implementierungsphase wurde die benötigte *tilemap*-Funktionalität in einem Update (Phaser Version 3.5) gebracht, zu diesem Zeitpunkt war die Entwicklung aber zu weit fortgeschritten, als dass die neue Funktionalität ohne weitere Anpassungen des Spiels zu implementieren gewesen wäre.

Für die Karte wurden drei prozeduralen Komponenten gewählt:

- Landschaft
- Straßen
- Städte

Dabei wurde für jede Komponente eine eigene prozedurale Technik gewählt.

[3]Bei einer Chunkgröße von 20 „verwaltet“ jeder Block einen Address-Bereich von 0 bis 399

6.3.1 Landschaft

Abbildung 6.4: Prozedural erzeugte Landschaft (mit Straßen)

Die Landschaft beschreibt die Oberfläche der Karte. Sie besteht aus verschiedenen Biomen — Wiesen, Wälder und Gewässer (Abbildung 6.4). Sie wird mittels Perlin-Noise erstellt.

Dazu wird mit einem Phaser-Plugin ein Rauschen erstellt, interpoliert und unter die Karte „gelegt“ (Unterabschnitt 3.6.3). So kann jeder Kachel eine *noise value* zugeordnet werden. Diese Werte befinden sich in einem Intervall $[-1, 1]$ und bilden räumliche „Cluster“, also lokale Minima und Maxima (zu sehen in Abbildung 3.7). Der Startwert (*seed*) der Rauschenfunktion ist das aktuelle Datum[4].

Zu diesem Zeitpunkt existiert im Spiel eine Karte voller Kacheln, denen jeweils ein Rauschen-Wert zugeordnet wurde. Diese Werte werden in Wertebereiche unterteilt[5]:

- Kleiner X: Gewässer
- Zwischen X und Y: Wiese, Weide, Flur
- Größer Y: Wald, Gehölz

Da die Wertebereiche von Wald und Gewässer nicht aneinandergrenzen, werden im Normalfall keine Wasserkacheln neben Waldkacheln zu sehen sein.

Schließlich wird jeder Kachel entsprechend ihres Biom ein *sprite* gesucht, und in der Szene angezeigt.

Um die Karte anreizender zu gestalten, ihr mehr „Leben“ zu verleihen, und die Monotonie zu bekämpfen, wird ein zweites Rauschen (mit einer gröberen Körnung) erzeugt, um jeder Kachel eine Tönung zu geben. Im Spiel wird für Wiesen und Gewässer nur jeweils ein einziges Bild verwendet, Dennoch sieht die in Abbildung 6.4 gezeigte Landschaft nicht monoton aus.

Ein weiterer Effekt für eine bessere Optik ist Tiefe — beispielsweise sind die Gewässerkacheln tiefergelegt als das umliegende Grasland. Dieser Effekt sollte

[4]Vergangene Millisekunden seit dem 1. Januar 1970

[5]Die genauen im Spiel verwendeten Wertebereiche (X, Y) sind unerheblich, da sie sich von Spiel zu Spiel unterscheiden.

auch für die Wälder genutzt werden, um ihnen einen Höhenunterschied (*Offset*) zu verleihen. Dazu war ein weiteres Anwenden der Rauschenfunktion angedacht. Aus Spielbarkeitsgründen wurde dieser Ansatzt aber verworfen, da der Effekt entweder nicht sichtbar war, oder den Spielfluß gestört hat.

Dies zeigt aber auch, dass die PCG beliebig ausgeweitet werden kann, um weiterführende Effekte zu erzeugen und das visuelle Ergebnis zu verbessern. Auch die Basisgeneration könnte durch Anwendung wiederholter Rauschenfunktion verbessert werden — anstelle der simplen Unterteilung in drei Wertebereiche könnte eine komplexere Unterteilung auf Grundlage von Höhenrelief, Luftfeuchtigkeit, Bodenbeschaffenheit, ... durchgeführt werden.

6.3.2 Straßengenerierung

Die Überlandstraßen (Abbildung 6.4) bilden die Platzierunggrundlage für die Fahrzeuge.

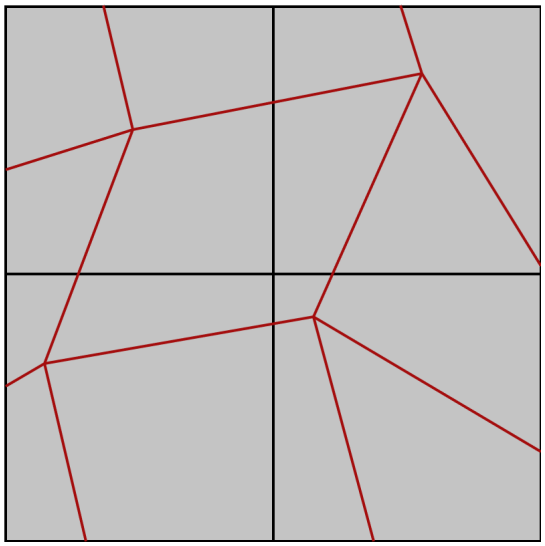


Abbildung 6.5: Straßengenerierung. In jedem Block wird ein zufälliger Punkt ausgewählt und mit den umliegenden Blöcken verbunden.

Dazu wird in jedem Block ein zufälliger Punkt gesucht, und mit den jeweils umliegenden vier Blöcken verbunden (Vierer-Nachbarschaft; Abbildung 6.5). Hat ein Block in einer Richtung keine Nachbarn, so wird stattdessen ein zufälliger Punkt auf der Außenkante gewählt. Die zufälligen Punkte (rot) sind in Abbildung 6.3 zu sehen.

Da die Karte aus viereckigen Kacheln besteht, können die Straßen nicht wie in der Abbildung gezeigt schnurgerade angelegt werden. Auch sollen sie das vorhandene Terrain berücksichtigen, und –wie in der Realität– um einen See herum führen, anstatt quer darüber.

Aus diesem Grund wurde für die Verbindung der Punkte ein Wegfindungsalgorithmus (A*[6]) verwendet. Dieser sucht den kürzesten Weg zwischen zwei Punkten.

[6]Die genaue Funktionsweise ist für das Verständnis der Arbeit irrelevant.

Auch läßt er sich gewichten, so dass im Präferenzfall vorhandene Straße wiederverwendet werden, und nur im Notfall eine Brücke über ein Gewässer geschlagen wird. Da der Algorithmus immer nur den Weg zwischen zwei Punkten suchen kann, wird er rekursiv angewandt, bis alle Straßen generiert sind. Aus diesem Grund wird er auch synchron angewandt, damit bereits generierte Straßen in der Generierung berücksichtig werden.

6.3.3 Städte

Aus prozeduraler Sicht ist das Erstellen von Städten eine schwierige Angelegenheit, da sie sehr komplexe, nicht natürliche Strukturen bilden [56]. Zwar gibt es geometrisch geordnete Städte (Planstädte wie New York, Karlsruhe, ...), allerdings sind die meisten Städte auf Grund einer Vielzahl an Faktoren –beispielsweise *„location, geography, cultural influences, planning trends, etc“* [56, S. 104]– organisch gewachsen.

Es gibt eine Vielzahl an Ansätzen und Möglichkeiten zum prozeduralen Erstellen von Städten. In dieser Arbeit werden Städte als Planstädte durch eine Fraktalähnliche, rekursive Genierierung unter Zuhilfenahme von Schildkrötengrafiken generiert (Unterabschnitt 3.6.2).

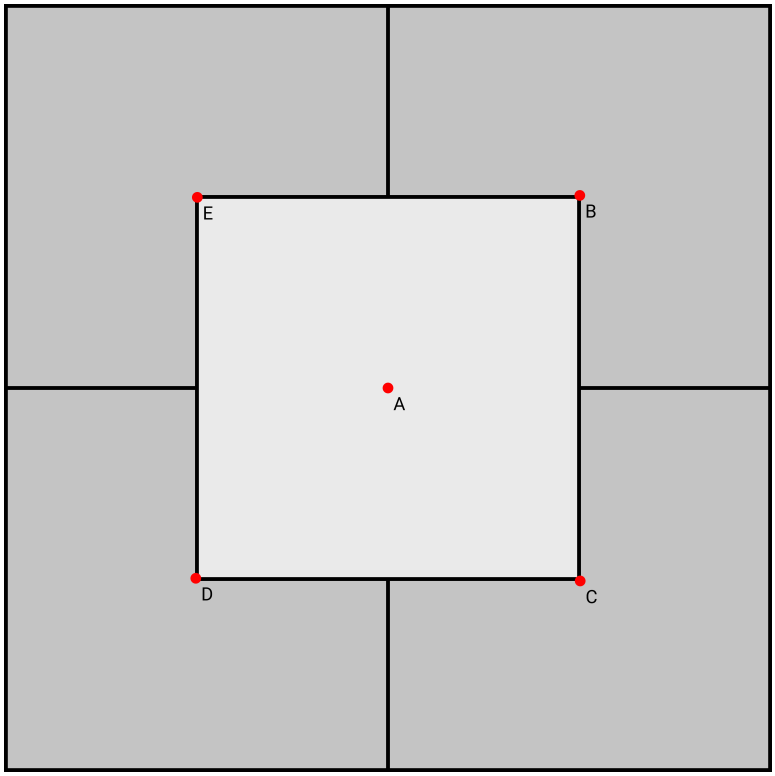


Abbildung 6.6: Rekursive Stadtgenerierung. Punkt A bildet den Mittelpunkt des Startvierecks, während die Eckpunkte B, C, D und E als Startpunkte weiterer Stadtviertel dienen.

Dazu wird ein zufälliger Punkt auf der Karte als Mittelpunkt der Stadt genommen (Abbildung 6.6, Punkt A). Um diesen Mittelpunkt wird ein Viereck gezeichnet. Der

Umriß des Viereckts fungiert dabei als Straße, während das Innere als Parkanlage, Plaza oder Wohnviertel dient. Rekursiv werden die vier Eckpunkte (B, C, D und E) als jeweils neue Startpunkte für die Generierung verwendet, ohne aber existierende Straßen und Gebäude zu überschreiben. Dieser Vorgang kann beliebig wiederholt werden. Die erstellte Stadt ist somit fraktal (Unterabschnitt 3.6.1)

Abbildung 6.7: Die generierte Stadt

Parkanlage und Plaza werden mittels Schildkrötengrafik erstellt. Diese wandert über das Innere des Vierecks und platziert Bäume und Springbrunnen. Hier ist es auch denkbar vorgefertige Baupläne zu verwenden, beispielsweise im JSON-Format.

6.4 Fahrzeuge

Die Fahrzeuge stellen die Zielobjekte und Distraktoren dar. Sie werden nach der Generierung basierend auf der Schwierigkeit auf der Karte platziert. Dabei können sie auf Kacheln mit freien Straßen oder freien Feldern platziert werden, wobei darauf geachtet werden muss, dass Kacheln nicht doppelt belegt werden, oder durch Gebäude *o.Ä.* blockiert sind. Auch muss darauf geachtet werden, dass genügend Plätze für die zu platzierenden Fahrzeuge vorhanden sind.

Die Fahrzeuge existieren in in mehreren Varianten (Abbildung 6.8):

- Sonderfahrzeuge (Polizeiauto, Krankenwagen, Müllabfuhr, Taxi)
- Zivil-Fahrzeuge (Sedan, Truck, Pickup)
  - Zivil-Fahrzeuge können in den folgenden Farben vorkommen: Rot, Grün, Blau, Schwarz, Silber

Abbildung 6.8: Fahrzeuge und Markierungen

Weiterhin können alle Fahrzeuge in eine von acht möglichen Richtungen fahren (N, W, S, E, NW, NE, SW, SE).

Für die Platzierung werden erst die Zielfahrzeuge ausgewählt, und dann mit den Distraktoren vermischt, um eine zufällige Platzierung zu gewährleisten.

Hierbei haben sich während der Konzeptionierung unbedachte Spielvarianten herausgestellt — so ist es möglich, Zielfahrzeuge und Distraktoren strikt zu trennen, aber auch, sie zu vermischen und Dopplungen zu erlauben. Beispiel: Die Szene enthält drei Krankenwagen, aber nur einer soll gefunden werden. Jeder der drei Krankenwagen ist ein valides Ziel, bis einer von ihnen ausgewählt wurde. Ab diesem Zeitpunkt gelten die andern beiden Wagen als Distraktoren und geben einen Fehler, sollten sie gewählt werden.

Wird ein Fahrzeug ausgewählt, wird überprüft, ob der Spieler eine richtige oder eine falsche Auswahl getroffen hat, und eine entsprechende Markierung gesetzt — ein blauer Haken oder ein roter Totenkopf (Abbildung 6.8). Auch werden dementsprechend Punkte gegeben oder abgezogen.

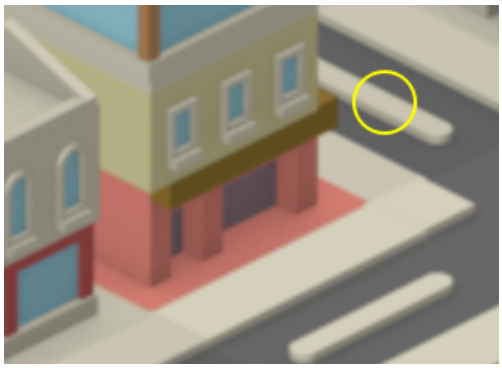
Die Verdeckung der Fahrzeuge war nicht wie gewünscht umsetzbar. Da die einzelnen Sprites nur Bilder sind, und diese sequenziell übereinander gesetzt werden, kommt es zu ungewollten Überlagerungen Abbildung 6.9: Das Fahrzeug schwebt über dem Baum, da in der Reihenfolge der Generierung zuerst die Landschaft und dann die Fahrzeuge platziert wurden. Dies zeigt sich auch in Abbildung 6.10: Der Krankenwagen ist hinter dem Müllauto, wird aber in der falschen Reihenfolge gezeichnet.

Problematisch ist die gewollte Verdeckung von Fahrzeugen hinter Gebäuden. Abbildung 6.11 zeigt das Problem: Der Mauszeiger befindet sich ungefähr an der Position des gelben Zirkels (a), es wird aber ein daneben befindliches Gebäude rot hervorgehoben. Der Umriß des Gebäudesprites gibt Aufschluss über das Problem (b): Das Gebäude ist größer als gedacht, es wird auf Grund der Platzierungsreihenfolge hervorgehoben, obwohl der Spieler eigentlich auf die Straße (und ein dort befindliches Fahrzeug) klicken wollte.

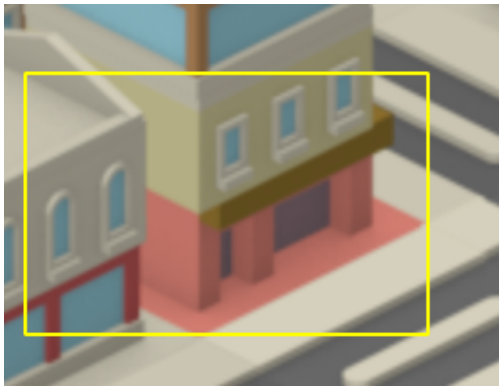
Abbildung 6.9: Das Fahrzeug „schwebt“ über dem Baum.

Abbildung 6.10: Verdeckung der Fahrzeuge

Während sich die Überlagerung der Fahrzeuge (Abbildung 6.10) durch eine Änderung der Zeichenreihenfolge beheben lassen würde, ist die gewollte Verdeckung von Fahrzeugen durch Gebäude *o.Ä.* ein schwieriges Problem: Während der Generierung muss überprüft werden, dass kein Fahrzeug völlig verdeckt ist, da es sonst nicht gefunden werden kann (Abbildung 6.12). Diese Überprüfung basiert auf dem Rahmen der Bilder. Da die Gebäude aber einen „falschen“ Rahmen haben, kann dieser nicht für die notwendige Überprüfung benutzt werden. Als Alternative kann um jedes Gebäude ein eigener Rahmen (*bounding box*) gezeichnet werden. Dies ist arbeitsintensiv, erfordert manuelle Nachbearbeitung und ist nicht skalierbar, da die Sprites unterschiedliche Größen haben. Als weitere Schwierigkeit zeigt sich der akzeptable Grad der Verdeckung: Ab wann gilt ein Fahrzeug als völlig verdeckt und somit unauffindbar? Somit müssen auch die Fahrzeuge in kleinere Teile aufgeteilt werden, mit denen die Verdeckung geprüft werden kann.



(a) Mausposition (gelb) und hervorgehobenes Gebäude (rot)



(b) Tatsächliche Größe des Gebäudesprites

Abbildung 6.11: Problem der Verdeckung

Abbildung 6.12: Verdeckung der Fahrzeuge. Beide Fahrzeuge sind „verdeckt“.

6.5 Kamera und Steuerung

Die im Spiel verwendete Kameraperspektive basiert auf dem Anwendungskontext und zeigt das Spiel leicht schräg von oben herab, so dass die isometrische Perspektive gewahrt bleibt.

Die Kamera kann bewegt und gezoomt werden. Dazu kann die Tastatur (Bewegung mit WASD, Zoom mit QE) und die Maus (Scrollrad, Mouse-Drag) verwendet werden.

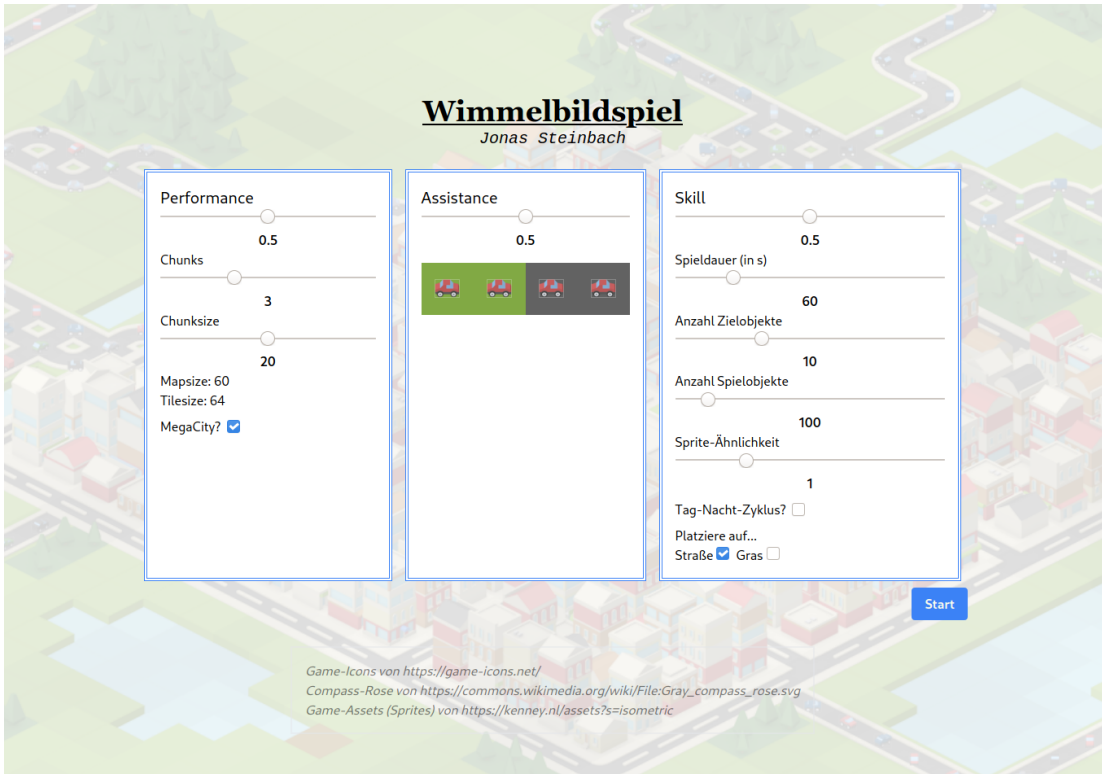


Abbildung 6.13: Der Startbildschirm mit den Einstellungsmöglichkeiten der DDA-Parametern

6.6 UI

Die UI wird mit React erstellt und zeigt den Status des Spiels (Punkte, Zeit, Zielvorgaben), ohne, dass sie über eine eigene Spiellogik verfügt.

Die Zielvorgabe greift dabei auf den *Sprite Atlas* aus dem Spiel zurück, um die Performance zu steigern. Dabei wird die Position und Größe des *Sprites* des Zielobjekts in Phaser ausgelesen und an die UI weitergeleitet. Diese lädt den Atlas und fokussiert ihn auf die ermittelte Größe und Position. Dies verhindert, dass für die UI die Spielinhalte ein zweites Mal geladen werden müssen.

Farblich ist die UI in grau, blau und weiß gehalten, um neutral den wissenschaftlichen Anwendungskontext zu verdeutlichen (Abbildung 6.1, Abbildung 6.13 und Abbildung 6.8).

Um die Spielwelt einzunorden und die Fahrtrichtung der Fahrzeuge bestimmen zu können, wird in der unteren rechten Ecke des Spiels ein Kompass angezeigt (Abbildung 6.1).

6.7 Distraktoren

Neben den unterschiedlichen Fahrzeugen kommen im Spiel weitere Distraktoren zum Einsatz.

Abbildung 6.14: Nachtzyklus mit überfliegendem Raumschiff

In unregelmäßigen Abständen fliegen –basierend auf der Schwierigkeit– Raumschiffe über das Spielfeld (Abbildung 6.14).

Die Spielzeit läuft –basierend auf der Schwierigkeit– langsamer oder schneller ab.

Das Spiel kann einem Tag-Nacht-Zyklus unterworfen werden. Dabei wird langsam die Opazität eines sich vor der Kamera befindlichen Bildes verändert, so dass die Illusion einer Helligkeitsveränderung ensteht (Abbildung 6.14).

Kapitel 7

Benutzertest

Um die Implementierung zu validieren, wurde ein Benutzertest durchgeführt. Im Fokus lagen dabei a) Spielbarkeit und b) Parametrisierung.

Covid19-bedingt konnte der Test nur im Freundes und Bekanntenkreis durchgeführt werden. Dafür wurden fünf Probanden gewählt, drei in der Zielgruppe (17–29) und zwei außerhalb (29+). Die beiden Probanden außerhalb der Zielgruppe wurden gewählt, um eventuelle zielgruppenspezifische Abweichungen des Spiels entdecken und vergleichen zu können.

Der Benutzertest erfolgte im privaten Rahmen unter Einhaltung aller von der Bundesregierung vorgegebenen Hygieneregeln — innerhalb des selben Haushalts im geschlossenen Raum, ansonsten im Freien mit einem Laptop. Auf einen Test der Anwendung mittels Remote-Software wurde verzichtet, um die Probanden während der Bedienung beobachten zu können (Mimik, Gestik, Mausbewegungen, ...).

Von den Probanden innerhalb der Zielgruppe verfügte eine Person über Bundeswehrerfahrung und bestätigt alle in der Zielgruppenbetrachtung aufgestellten Kriterien (Jung, Verantwortungsbewußt, Technikaffin, Abenteuerlustig), während die beiden anderen Probanden nur eine geringe Abenteuerlust angeben. Alle drei „jungen“ Probanden sind männlich.

Die beiden Probanden außerhalb der Zielgruppe gaben keine hohe technikaffinität an, bejahten aber alle anderen Fragen (und fühlten sich „innerlich jung“). Eine Person ist männlich, die andere weiblich.

7.0.1 Ablauf

Der Test bestand aus drei Teilen. Zuerst wurde den Probanden das Spiel und die Steuerung erklärt, dann mussten sie eine Runde spielen, ohne, dass ihnen von außen geholfen wurde. Dabei wurden sie aufgefordert laut zu denken. Danach fand eine kurze Befragung statt, in der Schwierigkeiten geklärt wurden. Anschließend wurde das Spiel noch einmal gespielt, diesmal im Dialog.

In der ersten Phase wurde die Spielbarkeit des Spiels überprüft: Ist das Spiel spielbar? Gibt es Fehler oder Abstürze? Unklarheiten beim Spielziel? Schwierigkeiten in der Bedienung? Auch wurde für die erste Phase eine lange Bearbeitungszeit gewährt, um jedem Probanden die Möglichkeit zu geben, sich an das Wimmelbildspiel zu gewöhnen.

Während der Diskussion wurden beobachtete Schwierigkeiten besprochen, Feedback und Kritik gesammelt und Tipps zum Spiel gegeben.

Die zweite Phase des Spieltests diente dem Test der Parametrisierung. Während der ersten Phase konnte das Fertigkeitenlevel der Probanden eingeschätzt werden. Dementsprechend wurde die Schwierigkeit durch Veränderung der Parameter angepasst, um zu beobachten, ob sich eine Auswirkung auf das Spielerverhalten ergeben hat.

Vier der fünf Probanden bestanden auf eine inoffizielle dritte Phase, in der versucht wurde, den Highscore zu knacken. Der Proband mit Bundeswehrerfahrung gewann.

7.0.2 Ergebnisse

Inklusive Diskussion und dritter Phase hat jeder Benutzertest im Schnitt 30 Minuten gedauert.

Es konnten verschiedene Spiel- und Suchstrategien beobachtet werden. Die meisten Probanden haben aus einer niedrigen Zoomstufe von weit oben auf die Spielwelt geschaut und nach den Objekten gesucht. Die Limitierung der Zoomfunktion auf großen Spielkarten wurde kritisiert.

Die Steuerung des Spiels wurde als nachvollziehbar und „aus anderen Spielen bekannt“ bezeichnet. Die Steuerung per Maus wurde präferiert, da sie schneller war.

Die Sprites und Zielvorgaben sind zu klein. Bei großer Ähnlichkeit der zu suchenden Fahrzeuge auf der Karte (Bsp. Fahrtrichtung Norden vs. Süden) haben sich alle Probanden zum Bildschirm gebeugt, um die Sprites besser zu sehen.

Die Distraktoren (fliegende Raumschiffe) wurden als „lustig“ und „nervig“ bezeichnet. Der Tag-Nacht-Zyklus wurde als „sehr nervig“ bezeichnet, und hat wiederholt den Suchfluss unterbrochen.

Die UI ist als reine Informationsquelle gedacht, wurde aber von den Probanden als interaktiv betrachtet — die meisten Probanden haben in ihrer ersten Spielrunde versucht, auf die Zielvorgaben zu klicken.

Die Punkte/Bewertung wurde als minimal und „nicht sehr informativ“ bezeichnet. Es wurde gewünscht, dass erweiterte Bewertungsstatistiken eingesehen werden können, beispielsweise eine Statistik zu Richtig/Falsch oder auch ein zeitlicher Graph.

Während des Benutzertests kam es zu keinen Abstürzen, die Anwendung lief stabil und flüßig. Es wurden jedoch Fehler in der Steuerung und Punktevergabe gefunden.

Alle Probanden bewerteten das Spiel als spielbar. Das Spiel hat Spaß gemacht, und es wurde auch jenseits des Benutzertests angefragt, ob sie das Spiel nicht noch einmal gespielt werden könnte.

Für die Vergleichbarkeit „unter Wettkampfbedinungen“ wurde von mehreren Probanden die Teilbarkeit und Wiederverwendbarkeit der Karten gewünsch, beispielsweise durch einen gleichen Startwert der Rauschenfunktion (*Seed*).

Einem der Probanden ist die veränderte Zeit im Spiel aufgefallen. Dabei wurde auch eine Pausenfunktion gewünscht.

Die Probanden der Zielgruppe hatten keine Probleme mit der Steuerung, da sie ihnen aus anderen Spielen vertraut ist. Die beiden Probanden jenseits der Zielgruppe hatten Schwierigkeiten. Auch hatten die Probanden jenseits der Zielgruppe Schwierigkeiten, zwischen gefundenen und zu suchenden Objekten zu unterscheiden. Der blaue Rahemn und gesetzte Haken, mit denen die Zielvorgabe bei richtiger Auswahl in der UI versehen wurde, wurde oftmals ignoriert und bereits gefundene Objekte erneut gesucht. Mehrere Probanden haben sich gewünscht, dass gefundene Objekte in der Zielvorgabe „nach unten rutschen“.

Durch den selben Ablauf der zweiten und dritten Phase, und der somit gegebenen Vergleichbarkeit konnte eine Schwäche des Testablaufs nachgewiesen werden — die Eingewöhnungszeit der ersten Phase ist nicht genug, um Aussagen zur Schwierigkeisveränderung der zweiten Phase zu treffen. Die Probanden sind bei gleichbleibender Schwierigkeit in der dritten Phase kontinuierlich besser geworden, bis sie zu einer für sie optimalen Suchstrategie gefunden haben. Deshalb wurde zum Ende der dritten Phase die Schwierigkeitsadjustierung der zweiten Phase wiederholt. Diesmal konnten Veränderungen festgestellt werden.

Keiner der Probanden hat die verschachtelt generierten Straßenzüge kommentiert.

In der Diskussion sind weitere Anregungen zum Spiel vorgebracht worden. So sollten die Distraktoren (Raumschiffe) weiter ins Spiel einbezogen werden und ebenfalls auswählbar sein. Auch gab es Vorschläge zur Personalisierung des Spiels — beispielsweise könnte ein Spieler statt Raumschiffen Flugzeuge oder Wildvögel als Distraktoren auswählen.

Schließlich vermeint ein Proband mit Erfahrung als Kunsttherapeut im Wimmelbildspiel gestaltpsychologische[1] Merkmale (Farben, Formen, Richtungen) zu entdecken. Auf Grund mangelnder Expertise wird diese Behauptung unkommentiert wiedergegeben.

[1] Gestaltpsychologie/Gestalttheorie (Wolfgang Metzger)

Kapitel 8

Diskussion

Um die Parametrisierung eines adaptiven Spiels zu zeigen, wurde der Prototyp eines adaptiven Wimmelbildspiels konzipiert, implementiert und in einem Benutzertest evaluiert. Dabei wurde das adaptive Spiel in Spiel und System unterteilt. Die Parametrisierung wurde dem Spiel zugeordnet und dementsprechend konzipiert.

Es gibt keine allgemeingültige Anleitung zum Erstellen von adaptiven Spielen, es konnten aber Methoden aus dem Game Design und Gamification Design angewandt werden, um das Spiel zu konzipieren. Besonderer Fokus lag dabei auf der Parametrisierung, die sich aus den drei DDA-Parametern der AR ergeben hat.

Problematisch war der COVID-19-bedingte mangelnde Kontakt zur Zielgruppe angehender Bildauswerter, es konnte dennoch benutzerorientert vorgegangen werden. Dafür wurde die Expertise der Stakeholders verwendet, und um eine Anwendungskontext- und Zielgruppenbetrachtung erweitert.

Das Wimmelbildspiel wurde mit Web-Technologie erstellt. Dafür waren mehrere Vorbereitungsschritte notwendig. Es musste ein lokaler Entwicklungs-Server aufgesetzt werden, da Web-Browser nur eingeschränkten Zugriff auf das lokale Dateiensystem haben. Auch mussten weitere für die Entwicklung benötigte Tools eingerichtet werden. Weiterhin mussten die für das Spiel benötigten Sprites sortiert und umbenannt werden. All dies hat mehr Zeit veranschlagt als dafür eingeplant war.

Die gefundene Game Engine (Phaser) konnte erfolgreich für die Erstellung des Wimmelbildspiels eingesetzt werden. Es gab aber Schwierigkeiten mit der Verwendung von *Tilemaps* und der isometrischen Perspektive und PCG. Deshalb wurde die benötigte Engine-Funktionalität in einer Alternativlösung selbst umgesetzt. Sie umfasst die Verwendung von drei Koordinatensystemen. Damit konnte die PCG erfolgreich umgesetzt werden. Gleichzeitig ist diese Lösung nicht skalierbar und auf den Prototypen zugeschnitten. Bei einer Weiterentwicklung des Prototypen sollte die offizielle Phaser-Lösung verwendet werden — diese wurde zum Ende der Entwicklungsphase als neues Phaser-Update veröffentlicht, und verspricht die Probleme zu beheben. Eine Änderung der Programmstruktur hat zu diesem

Zeitpunkt aber keinen Sinn gemacht, da das Projekt zu weit fortgeschritten war.

Als Programmiersprache für das Spiel wurde TS gewählt. Diese hat verhindert, dass Typenfehler aufgetaucht sind. Gleichzeitig gab es aber Probleme in der Verwendung, da viele Anwendungs- und Programmierbeispiele für Phaser und React in JS geschrieben sind, und dementsprechend konvertiert werden müssen. Dafür waren die Phaser und TS Online-Communities sehr hilfreich — während der Entwicklungsphase konnten wiederholt Probleme nachgefragt und gelöst werden.

Hierbei ergeben sich Überlegungen zu Wartbarkeit und Komplexität des Programms: *„Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it“* (*Kernighan's Law*, benannt nach Brian Kernighan in [57]). Dementsprechend wurde bei der PCG auf fortgeschrittene Programmierkonzepte (beispielsweise Generics) verzichtet, um die Wartbarkeit des Programms nicht zu beeinträchtigen. Auch sollte das Programm für neue, unerfahrene Mitglieder der Forschungsgruppe verständlich sein, da die Forschungsgruppe aus einem heterogenen Verbund verschiedener Expertisen besteht, und eine Fluktuation von Gruppenmitglieder nicht ausgeschlossen werden kann.

Allgemein war die Implementierung schwierig zu testen, da die Ergebnisse der PCG graphisch sind, und nur schwierig in automatischen Tests überprüft werden können. Deshalb mussten die Algorithemn manuell angepasst werden, was „Augenmaß“ bei der Auswahl der Ergebnisse erfordert hat. Dazu gehört beispielsweise die Intervallgröße der Rauschenfunktion, um eine ansprechende Verteilung der Biome zu gewährleisten.

Die Spielszene wird prozedural generiert. Dies passiert auf Basis einer Rauschenfunktion, die die Grundlagen für die Verteilung von Biomen (Gewässer, Wiesen, Wälder) bildet. Die Generierung eignet sich für die Erstellung der Karte, ist aber ausbaubar. Beispielsweise kann die Karte um weitere Biome erweitert werden, die basierend auf multiplen Rauschenfunktionen erstellt werden und separate Eigenschaften wie Luftfeuchtigkeit, Bodenbeschaffenheit, Höhe über NN, ... ausdrücken. Die Biome könnten auch Auswirkungen auf Distraktoren –beispielsweise Wettereffekte– haben.

Das Einbringen von Schattierungen mittels einer Rauschenfunktion ist ein effektiver und visueller Gewinn für das Spiel.

Die prozedural generierten Straßen sind ästethisch wenig anspruchsvoll, da sie oft einen verschachtelten Bandwurm an Straßenzügen bilden, wie es ihn in der Realität nicht gibt (Abbildung 8.1). Hierbei sei angemerkt, dass das Spiel jenseits der Realität liegt und nicht an ihr gemessen werden sollte. Dennoch macht es im Hinblick auf den Anwendungskontext Sinn, dass hier nachgebessert wird. Die Problematik der Straßengenerierung stammt aus der Grundstruktur der Karte. Die Kacheln sind viereckig, weshalb nur gerade Wegverbindungen möglich sind. Die verwendete Art der Generierung liegt aber auch an den gefunden Sprites.

Für eine Verbesserung der Generierung sollten Straßenbilder gefunden werden, die sich in einer Achter-Nachbarschaft (N, W, S, E, NW, NE, SW, SE) verbinden

Abbildung 8.1: Bandwurmstraße

lassen. Für die Generierung bieten sich viele alternative Ansätze an [56], beispielsweise die Verwendung einer Mesh-basierten Generierung an. Dies erfordert aber die Nutzung von 3D-Modellen.

Die Städte konnten mit dem gewählten rekursiven Verfahren generiert werden. Dabei wird aber nur eine einzige Variante einer Stadt erstellt. Hier bietet sich die Erweiterung des Verfahrens um Baupläne an. Dabei werden Schildkrötengrafik-freundliche Beschreibungen von Städten und Siedlungen erstellt, und dann im Spiel an einer gewünschten Stelle generiert. Diese Beschreibungen könnten mit einem Karten-Editor (beispielsweise (Tiled)) erstellt und kuratiert werden, um den Anwendungskontext des Wimmelbildspiels zu erweitern. Dies gäbe auch Anreize für eine mögliche Personalisierung und könnte sogar als *user-generated content* Verwirklicht werden — dabei erstellen die Benutzer im Spiel anzuzeigende Städte und Strukturen. Hier muss aber die Frage der Lernziele im Blick bleiben.

Es stellt sich auch die Frage nach Art der Generierung. In der aktuellen Fassung des Wimmelbildspiels wird die Spielwelt lokal im Spielclient generiert. Dies ist Zeitintensiv (besonders bei großen Kartengrößen) und schmälert die Spielbarkeit. Es ist anzunehmen, dass eine Auslagerung der Spielweltgeneration an einen externen Server Vorteile bringt.

Die im Spiel verwendeten Fahrzeuge eignen sich für den Anwendungskontext. Sie zeigen unterschiedliche Fahrzeuge und können unterschieden werden. Gleichzeitig sind die verwendeten Sprites viel zu klein und müssen verbessert werden. Dies hat bereits während der Erstellung der Spritesheets zu Schwierigkeiten geführt und hätte schon dort erkannt werden müssen.

Die Problematik der kleinen Sprites konnte während des Spielertests bestätigt werden. Die Probanden haben sich wiederholt näher an den Bildschirm gebeugt,

um die Fahrzeuge besser unterscheiden zu können. Auch waren einige Spritevarianten nur schwer von einander zu unterscheiden. Hier bietet sich die Konzeption einer interaktive Vorschaufunktion an: Bei Klick auf die Zielanzeige wird eine große Vorschau des zu suchenden Objekts gegeben.

Die Problematik der zu kleinen Sprites deckt sich mit der von Lillesand et al. [64, S. 195ff] gemachten Bemerkung zur *Resolution* — bei geringer Auflösung ist es schwierig Details zu erkennen/unterscheiden.

Als Alternative zu den im Spiel verwendeten PNG-Grafiken bieten sich SVG-Grafiken an. Diese sind vektorbasiert und können beliebig skaliert werden, während PNG-Grafiken pixelbasiert sind und nur mit Unschärfe vergrößert werden können.

Die verwendete isometrsiche Perspektive eignet sich nur bedingt für das Spiel. Sie wurde gewählt, um Verdeckungen von Fahrzeugen zu ermöglichen. Dies konnte aber auf Grund der Struktur der isometrischen Sprites nicht umgesetzt werden. Es empfiehlt sich, für die Verdeckung 3D-Sprites zu verwenden.

Die Platzierung der Fahrzeug findet Kachelbasiert statt. Auf jeder Kachel wird maximal ein Fahrzeug platziert. Komplexere Platzierungen (mehrere Fahrzeuge vor und hintereinander) wurden wegen der Verdeckung nicht umgesetzt. Hierbei bieten sich auch Überlegungen zum *Clustering* der Fahrzeuge an — anstatt dass sie zufällig über die Straßen verteilt werden, könnten sich Ansammlungen von Fahrzeugen bilden und die Verteilung der Fahrzeuge parametrisiert werden [35].

Fehler und Erfolge werden im Spiel durch Marker angezeigt, die auf die Karte gesetzt werden. Dabei werden die angewählten Fahrzeuge entfernt/verdeckt. Alternativ könnten diese auch weiterhin im Spiel bleiben, um so den selben Fehler mehrmals zu ermöglichen. Es wird dabei kein Marker gesetzt, sondern nur kurz Rückmeldung über Erfolg oder Misserfolg der Spieleraktion gegeben.

Die Bewertung des Spielers erfolgt durch eine Punktevergabe im Spiel. Diese ist ausbaubar, was auch während des Benutzertests angemerkt wurde. Besonders fiel dabei die Bitte der Probanden nach einer besseren und nachvollziehbareren Bewertung auf, was sich mit den Erkenntnissen aus [91, S. 288] deckt. Auch wurden nicht alle konzipierten Features der Punktevergabe umgesetzt — so fehlen Boni, Mali und Multiplikatoren.

Hierbei zeigt sich auch eine konzeptionsrelevanz des Spiels für die *Capture*-Phase des Adaptivitätszyklus. Die für den Spieler generierte Statistik benutzt die selben Daten, die auch für das adaptive System relevant sind. Es gilt also diese Daten nicht nur aufzunehmen, sondern sie auch dem Spieler anzuzeigen. Die Verwendung und Anzeige von Spiel-internen Daten bringt auch Vorteile in Bezug auf Datenschutz und Ethik: Der Spieler kann genau einsehen, welche Daten über ihn aufgenommen werden.

Die Steuerung im Spiel ist –basierend auf dem Benutzertest– für technikaffine Jugendliche und junge Erwachsene gut nachvollziehbar. Gleichzeitig wurden PCG-spezifische Mängel festgestellt. Der Kamera-Zoom sollte mit der Größe der Karte skalieren. Ebenfalls wurde die Bedienbarkeit durch eine Maussteuerung als höher

als die Bedienbarkeit durch die Tastatur bescheinigt — hier muss nachbessert werden. Weiterhin sollte sich das Spiel –wie von den Probanden gefordern– pausieren lassen.

Die UI, besonders die Zielvorgabe, wurde während des Benutzertests als Schwachpunkt ausgemacht. Sie wurde als interaktiv missverstanden. Auch sollten die zuvor erwähnten Vorschaubilder anzeigbar sein. Ansonsten gab es weder positive noch negative Kommentare zur grafischen Gestaltung der UI. Die Verwendung von React für die UI hat sich jedoch bewährt.

Das Spiel wird basierend auf den drei DDA-Parametern generiert. Diese steuern die Generierung. Die Aufteilung der Parameter hat gut funktioniert, das Spiel konnte in der Schwierigkeit angepasst werden.

Der *Performance*-Parameter steuert die Generierung der Karte. Damit wird bis jetzt nur die Größe der Karte geändert, ohne Einfluss die unterliegende Generierung zu nehmen. Hier bietet es sich an, auch weiterführende Parameter der Generierung zu steuern, beispielsweise Art und Aussehen der Städte, Vorkommen von Biomes, Komplexität der Straßen, ... Weiterhin sollte der Startwert der Rauschenfunktion (*Seed*) einstellbar und teilbar sein, so dass schwierige Level geteilt und wiedergespielt werden können.

Der *Assistance*-Parameter steuert die Hilfestellung. Sie wurde durch zwei Varianten realisiert: Hervorhebung der Fahrzeuge und Anzeigen/Verdecken der Vorschaubilder in der Zielanzeige. Die Hervorhebung der Fahrzeuge funktioniert gut, wird allerdings schnell zu einfach, besonders, wenn nur noch die zu suchenden Objekte hervorgehoben werden. Die Verdeckung in der UI ist schwierig. Besonders die textuelle Variante ohne Vorschaubild erfordert Vorwissen zum Aussehen der Fahrzeuge. In einem professionellen Anwendungsumfeld kann hierbei auf Wissen aus der Ausbildung zurückgegriffen werden, für den Laien sind die textuellen Beschreibungen aber nicht nachvollziehbar. Bei der Gestaltung der UI lohnt es sich, zielgruppenrelevante User Experience (UX)-Tests durchzuführen.

Für die Hilfestellung bieten sich weitere Möglichkeiten, wie beispielsweise die Hervorhebung der Zielfahrzeuge durch Richtungspfeile oder durch Konversation mit einem Non-Player Character (NPC), der Ratschläge und Tips gibt. Darauf aufbauend, könnte im Spiel auch eine längere Geschichte erzählt werden, beispielsweise durch Missionen und Quests.

Die Parametrisierung des *Skill*-Parameters wurde am weitesten ausgebaut. So konnten verschiedene Fahrzeugvarianten, die Verteilung der Fahrzeuge auf der Karte, die Häufigkeit der Fahrzeuge, die Spieldauer und der Tag-Nacht-Zyklus bestimmt werden. Besonders die Distraktoren wurden von den Probanden als „kreativ“ und „vielversprechend“ bezeichnet und enthusiastisch diskutiert. Die Konzeption hat gezeigt, dass hier großes kreatives Potetntial besteht, und sehr viele Distraktoren möglich sind. Auch die Personalisierung der Distraktoren sollte ins Auge gefasst werden. Allgemein bietet sich für die PCG ein User-generated Content (UGC)-Ansatz an, der sich mit der Gamification des adaptiven Systems deckt.

Die Verwendung von Zeit als anpassbare Wahrnehumngsdimensionen zur Erzeugung von Stressituationen ist ein vielversprechender Ansatz, der weiter ausgebaut und erforscht werden sollte. In diesem Zusammenhang muss auch die auditive Gestaltung des Spiels erwähnt werden. Sie eignet sich ebenfalls um im Spiel die Stimmung zu ändern und Stress zu erhöhen (Horrorfilm-Musik) oder zu verringern (Fahrstuhlmusik).

Neben der gezeigten statischen Spielkarte könnten auch weitere Spielmodi realisert werden, beispielsweise ein *Sidescroller*, bei dem die Spielkarte sich langsam in eine Richtung bewegt. Auf der einen Seite werden fortlaufend neue Spielinhalte generiert, während sie auf der anderen Seite „verschluckt“ werden. Der Spieler muss nun alle Fahrzeuge finden, bevor sie verschwinden.

Auch auf der statischen Spielkarte lassen sich weitere Spielmodi einbauen. Beispielsweise könnten sich die Fahrzeuge nach jeder Auswahl bewegen, oder es müssen so viele Fahrzeuge wie möglich in einer bestimmten Zeit gefunden werden, ohne, dass es eine Obergrenze gibt.

Viele Parametrisierungsmöglichkeiten haben sich erst während der Implementierung ergeben, da größere Schwierigkeiten während der Konzeption nicht ersichtlich waren. Auch der Benutzertest war kreativ wertvoll. Er bestätigt das benutzerzentrierte Vorgehen und darf in keiner Konzeption fehlen. Dementsprechend ist die Entwicklung des Wimmelbildspiels nicht abgeschlossen sondern erst am Anfang, und sollte im selben iterativen Verfahren weiterentwickelt werden!

Die Spielertypen wurden während der Konzeption nur am Rande verwendet, es ist aber anzunehmen, dass sie für das Lernermodell relevant sind. In diesem Zusammenhang stellt sich auch der UCD-Ansazu in Frage: Ein adaptives System sollte sich an alle Benutzer anpassen — macht es dabei noch Sinn, auf einzelne Zielgruppen zu achten? Gleichzeitig war die Analyse der Zielgruppen und des Anwendungskontext während der Konzeption sehr hilfreich, besonders bei der Erstellung der Distraktoren und der Auswahl der *Sprites*.

Die Lernziele standen während der Entwicklung nicht im Vordergrund. Es ist anzunehmen, dass sie eine weitere Konzeptionsphase des Spiel mit sich bringen.

Das Wimmelbildspiel ist –gemäß den in Abschnitt 5.1 aufgestellten Anforderungen– adaptierbar und parametrisierbar; die Schwierigkeit des Spiels kann basierend auf den DDA-Parametern (AR) angepasst werden.

Die Hauptzielgruppe von Frauen für Wimmelbildspiele hat die Spielbarkeit nicht getrübt. Alle männlichen Probanden fanden das Spiel für geeignet und spaßig.

Kapitel 9

Fazit

In dieser Arbeit wurde gezeigt, wie ein adaptives Spiel parametrisiert wird. Dazu wurde ein adaptives Wimmelbildspiel konzipiert, implementiert und evaluiert.

Das adaptive Spiel läßt sich in Spiel und System teilen. Die Parametrisierung ist gemäßt dem Adaptivitätszyklus (Unterabschnitt 3.2.1) auf der Seite des Spiels (*Present*-Ebene) zu sehen.

Die Parametrisierung bestimmt die Erstellung und Anpassbarkeit der Parameter im Spiel. Deshalb erfogt sie während der Konzeption. Sie läßt sich als *Mechanics*-Element von Schell's [99] *Game Element Tetrad* sehen. Im Spiel wurde sie durch eine Mischung aus PCG und Nicht-PCG umgesetzt. Dabei bestätigt sich, dass sich alle Spieldimensionen und -Elemente adaptiv gestalten lassen. Auch ist anzunehmen, dass sich der verwendete Konzeptionsansatz für andere adaptive Spiele eignet.

Die PCG eignet sich gut für die Parametrisierung. Sie bietet viel Spielraum für Anpassungen der Spielwelt. Auch eignet sich die PCG für die Umsetzung von Distraktoren. Diese sollten auch im Zusammenhang mit Personalisierung des adaptiven Systems und damit verbundener UGC-Ansätze gesehen werden.

Während und nach der Implementierung wurden weitere Möglichkeiten zur Parametrisierung gefunden. Ein agiles und iteratives Vorgehen ist also sinnvoll.

Kapitel 10

Ausblick

Während der Arbeit haben sich weiterführende Überlegungen zum Themengebiet ergeben.

Zum einen bietet sich eine Untersuchung der Distraktoren im Hinblick auf ihre Personalisierbarkeit an. Auch die Sprites und die PCG können im Rahmen eines UGC-Ansatzes personalisiert werden. Besonders die Verwendung von *Bauplänen* für die Generierung der Städte ist im Hinblick auf eine Aufgabenteilung zwischen Game Design und Lernzielen interessant.

Im Hinblick auf den Adaptivitätszyklus stellt sich die Frage, ob sich ein adaptives Spiel selbst evaluieren und verifizieren läßt. Auch ist anzunehmen, dass der UCD-Ansatz mit Blick auf die Anpassbarkeit des Spiels noch einmal untersucht werden sollte.

Auch stellt sich die Frage, wie Spielertypen oder psychographische Merkmale der Zielgruppe besser in die Konzeption einfließen können.

Weiterhin fällt auf, dass *adaptive game design* nicht formal definiert ist. Hier bietet sich ein Einbezug der Parametrisierung an. Diese ist auch in Bezug auf andere Adaptivitätsmechanismen abseits der DDA oder AR interessant.

Auf den Anwendungskontext fokussiert ist die Untersuchung von Suchstrategien im Wimmelbildspiel interessant. Eignet es sich, um erweiterte Suchstrategien zu vermitteln?

Literatur

[1] Harold Abelson und Andrea Disessa. „Turtle Geometry“. In: *Cambridge and London* (1982). DOI: 10.7551/mitpress/6933.001.0001.

[2] Vincent Aleven u. a. „Instruction Based on Adaptive Learning Technologies“. In: *Handbook of research on learning and instruction* (2016), S. 522–560.

[3] Emad Ahmad Alghamdi. „An Investigation of the Efficacy of Multimedia Glosses in Incidental EFL Vocabulary Learning and Retention through Playing an Online Hidden-Object Game“. PhD Thesis. Missouri State University, 2014.

[4] Raed S. Alsawaier. „The Effect of Gamification on Motivation and Engagement“. In: *Technology* 35.1 (2018), S. 56–79. DOI: 10.1108/ijilt-02-2017-0009.

[5] Franz Aurenhammer. „Voronoi Diagrams—a Survey of a Fundamental Geometric Data Structure“. In: *ACM Computing Surveys (CSUR)* 23.3 (1991), S. 345–405. DOI: 10/bgct8t.

[6] Gonçalo Baptista und Tiago Oliveira. „Gamification and Serious Games: A Literature Meta-Analysis and Integrative Model“. In: *Computers in Human Behavior* 92 (2019), S. 306–315. DOI: 10/gf9psx.

[7] Richard Bartle. „Hearts, Clubs, Diamonds, Spades: Players Who Suit Muds“. In: (1996).

[8] Carole Beal u. a. „Intelligent Modeling of the User in Interactive Entertainment“. In: *AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*. 2002, S. 8–12.

[9] Florian Berger, Antonios Liapis und Georgios Yannakakis. „Prototyping an Adaptive Educational Game for Conflict Resolution“. In: *ITS 2012 Workshop: Emotion in Games for Learning*. 2012.

[10] C. Biegemeier. *Web-Basierte Schnittstelle Zur Analyse Und Adaption von Serious Games*. KIT, 2016.

[11] John Bird. „Whitewashing the Fence, Turning Work Into Play, Making Teaching and Learning Interesting and Engaging“. In: (2015).

[12] BMVG. *Im Visier*. 10. Sommer 2019. URL: https://www.bmvg.de/resource/blob/96176/a46bbe320309eb2c3bb8d6f38f49463c/im-visier-10-sommer-2019-data.pdf (besucht am 22. 04. 2021).

[13] Matthias Bopp. „Immersive Didaktik: Verdeckte Lernhilfen und Framingprozesse in Computerspielen“. In: (2005), S. 19. DOI: https://nbn-resolving.org/urn:nbn:de:0228-200506024.

[14] Isabel Briggs-Myers und Peter B. Myers. „Gifts Differing: Understanding Personality Type“. In: (1995).

[15] Bundeswehr, director. *Eye in the Sky – Luftbildauswerter Bei Der Arbeit - Bundeswehr*. 9. Jan. 2015. URL: https://www.youtube.com/watch?v=2zTW1qnrrjQ (besucht am 15. 12. 2020).

[16] Bundeswehr. *Feldwebel*. 2021. URL: https://www.bundeswehrentdecken.de/soldatenberuf/laufbahnen/feldwebel (besucht am 22. 04. 2021).

[17] Bundeswehr. *Karriere Als Feldwebel*. Mai 2019. URL: https://www.bundeswehrkarriere.de/blueprint/servlet/blob/168060/8a626b8d91604b11a8a96ad88broschuere-feldwebel-data.pdf (besucht am 22. 04. 2021).

[18] Bundeswehr. *Welcher Beruf passt zu Ihnen?* 2021. URL: https://www.bundeswehrkarriere.de/ihre-berufung/orientierungshilfe (besucht am 22. 04. 2021).

[19] bundeswehr. *Spezialist für Luftbildauswertung*. 2020. URL: https://www.bundeswehrkarriere.de/ (besucht am 15. 12. 2020).

[20] Marc Busch u. a. „Player Type Models: Towards Empirical Validation“. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*. The 2016 CHI Conference Extended Abstracts. San Jose, California, USA: ACM Press, 2016, S. 1835–1841. ISBN: 978-1-4503-4082-3. DOI: 10/ghphr5. URL: http://dl.acm.org/citation.cfm?doid=2851581.2892399 (besucht am 14. 12. 2020).

[21] Nils Büttner. *Hieronymus Bosch*. Bd. 2516. CH Beck, 2012.

[22] R. Caillois. *Man, Play, and Games*. Hrsg. von N/A. Urbana: University of Illinois Press, 2001.

[23] Paul Cairns, Anna Cox und A. Imran Nordin. „Immersion in Digital Games: Review of Gaming Experience Research“. In: *Handbook of digital games* 1 (2014), S. 767. DOI: 10/ghh794.

[24] Darryl Charles u. a. „Player-Centred Game Design: Player Modelling and Adaptive Digital Games“. In: (2005).

[25] Carolin Collin. *Farbenschwäche / Farbenblindheit*. Apotheken-Umschau. 5. Dez. 2011. URL: https://www.apotheken-umschau.de/krankheiten-symptome/augenkrankheiten/farbenschwaeche-farbenblindheit-733701.html (besucht am 30. 04. 2021).

[26] Thomas M. Connolly u. a. „A Systematic Literature Review of Empirical Evidence on Computer Games and Serious Games“. In: *Computers & Education* 59.2 (Sep. 2012), S. 661–686. ISSN: 03601315. DOI: 10/gcm4tv. URL: 10.1016/j.compedu.2012.03.004 (besucht am 30. 07. 2020).

[27] Mihaly Csikszentmihalyi. „Flow and the Psychology of Discovery and Invention“. In: *HarperPerennial, New York* 39 (1997).

[28] Richard Davey. *Photonstorm/Phaser*. 30. Apr. 2021. URL: https://github.com/photonstorm/phaser (besucht am 30. 04. 2021).

[29] Sebastian Deterding. „The Lens of Intrinsic Skill Atoms: A Method for Gameful Design“. In: *HUMAN–COMPUTER INTERACTION* 30 (2015), S. 294–335. DOI: 10/gf6726.

[30] Sebastian Deterding u. a. „From Game Design Elements to Gamefulness: Defining"Gamification"“. In: *Proceedings of the 15th International Acade-*

mic MindTrek Conference: Envisioning Future Media Environments. 2011, S. 9–15. DOI: 10/ctbr.

[31] Digitale Kultur. *Demoscene*. 2021. URL: https://www.digitalekultur.org/en/ (besucht am 01. 04. 2021).

[32] M.J. Dondlinger. „Educational Video Game Design: A Review of the Literature“. In: *J Appl Educ Technol* 4.1 (2007), S. 21–31.

[33] Yellowlees Douglas und Andrew Hargadon. „The Pleasure Principle: Immersion, Engagement, Flow“. In: *Proceedings of the Eleventh ACM on Hypertext and Hypermedia*. 2000, S. 153–160. DOI: 10/bs2bfk.

[34] David S. Ebert u. a. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, 2003.

[35] JOse Egas. „Generierung Virtueller Karten Für Digital Game Based Learning“. Bachelor. 2016.

[36] facebook. *React*. Facebook, 30. Apr. 2021. URL: https://github.com/facebook/react (besucht am 30. 04. 2021).

[37] Jiashi Feng u. a. „Purposive Hidden-Object-Game: Embedding Human Computation in Popular Game“. In: *IEEE Transactions on Multimedia* 14.5 (Okt. 2012), S. 1496–1507. ISSN: 1941-0077. DOI: 10.1109/TMM.2012.2198801.

[38] Jackie Fenn und Hung LeHong. „Hype Cycle for Emerging Technologies“. In: *Gartner, Stamford* (2011).

[39] Feuerwehr Hamburg. *Wir Suchen Eine/n Geowissenschaftlerin Bzw. Geowissenschaftler Oder Kartographin Bzw. Kartograph (m/w/d) Für Die Luftbildauswertung*. www.hamburg.de. 2021. URL: http://fhh.hrecruiting.de/service/preview_anzV3.php?a=zT60rb2Ydp6e1xUOVW (besucht am 14. 04. 2021).

[40] Theresa M. Fleming u. a. „Serious Games and Gamification for Mental Health: Current Status and Promising Directions“. In: *Frontiers in Psychiatry* 7.215 (2017), S. 1.

[41] Julian Frommel u. a. „Emotion-Based Dynamic Difficulty Adjustment Using Parameterized Difficulty and Self-Reports of Emotion“. In: *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. 2018, S. 163–171.

[42] G5 Entertainment. *Developers*. G5 Entertainment News. 2021. URL: https://www.g5e.com/developers (besucht am 19. 04. 2021).

[43] G5 Entertainment. *Earnings Q4 2020*. 2020.

[44] Howard E. Gardner. *Frames of Mind: The Theory of Multiple Intelligences*. Hachette Uk, 2011.

[45] Kiel M. Gilleade und Alan Dix. „Using Frustration in the Design of Adaptive Videogames“. In: (2004). DOI: 10/dxxkpd.

[46] Alexander Gundermann. „A Web-Based Serious Game for Joint Training“. 2016. 106 S.

[47] J. Hamari und V. Lehdonvirta. „Game Design as Marketing: How Game Mechanics Create Demand for Virtual Goods“. In: *Int. Journal of Business Science and Applied Management* 5.1 (2010), S. 14–29. URL: http://www.business-andmanagement.org/download.php?file=2010/5_1--14-29-Hamari,Lehdonvirta.pdf.

[48] Juho Hamari und Veikko Eranti. „Framework for Designing and Evaluating Game Achievements.“ In: *Digra Conference*. Bd. 10. 1.224. Citeseer, 2011, S. 9966.
[49] Martin Handford. *Wo Ist Walter?* Sauerländer, 1989.
[50] Mark Hendrikx u. a. „Procedural Content Generation for Games: A Survey“. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9.1 (2013), S. 1–22. DOI: 10/gd54kf.
[51] K. Huotari und J. Hamari. „A Definition for Gamification: Anchoring Gamification in the Service Marketing Literature“. In: *Mark* 27 (2016), S. 21–31. DOI: 10/f9ztqz.
[52] Kai Huotari und Juho Hamari. „Defining Gamification: A Service Marketing Perspective“. In: *Proceeding of the 16th International Academic MindTrek Conference*. 2012, S. 17–22. DOI: 10.1145/2393132.2393137.
[53] Elisavet Ioannidou. „Neo-Victorian Hidden Object Games, Participatory Culture, and the Interaction Between Past and Present“. In: *Adaptation* 11.2 (6. Aug. 2018), S. 159–170. ISSN: 1755-0637. DOI: 10.1093/adaptation/apx008. URL: https://academic.oup.com/adaptation/article/11/2/159/3782659 (besucht am 06. 06. 2020).
[54] Martin Jennings-Teats, Gillian Smith und Noah Wardrip-Fruin. „Polymorph: Dynamic Difficulty Adjustment through Level Generation“. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. 2010, S. 1–4. DOI: 10/dzs2c4.
[55] David Keirsey und Marilyn Bates. *Please Understand Me*. Prometheus Nemesis Del Mar, CA, 1978.
[56] George Kelly. „A Survey of Procedural Techniques for City Generation“. In: (2006). DOI: 10/gh594h. URL: https://arrow.dit.ie/itbj/vol7/iss2/5/ (besucht am 01. 03. 2021).
[57] Brian W. Kernighan und Phillip James Plauger. *Elements of Programming Style*. McGraw-Hill, Inc., 1974.
[58] Kristian Kiili. „Digital Game-Based Learning: Towards an Experiential Gaming Model“. In: *The Internet and Higher Education* 8.1 (Jan. 2005), S. 13–24. ISSN: 10967516. DOI: 10/b57nwm. URL: 10.1016/j.iheduc.2004.12.001 (besucht am 30. 07. 2020).
[59] A.J. Kim. „Beyond Player Types“. In: (15. Jan. 2014). URL: http://amyjokim.com/blog/2014/02/28/beyond-player-types-kims-social-action-matrix/.
[60] Kenneth R. Koedinger u. a. „Using Data-Driven Discovery of Better Student Models to Improve Student Learning“. In: *International Conference on Artificial Intelligence in Education*. Springer, 2013, S. 421–430. DOI: 10/gjtn9k.
[61] Raph Koster. *Theory of Fun for Game Design*. Ö'Reilly Media, Inc.", 2013.
[62] Shinil Kwon und Sungdeok Cha. „CAPTCHA-Based Image Annotation“. In: *Information Processing Letters* 128 (1. Dez. 2017), S. 27–31. ISSN: 0020-0190. DOI: 10/gjqxr5. URL: https://www.sciencedirect.com/science/article/pii/S002001901730131X (besucht am 19. 04. 2021).

[63] Eva Lehmann. „Entwicklung Und Akzeptanzanalyse Eines Kartenbasierten Lernspiels“. In: *Bachelor thesis, Hochschule Karlsruhe-Technik und Wirtschaft* (2015).

[64] Thomas Lillesand, Ralph W. Kiefer und Jonathan Chipman. *Remote Sensing and Image Interpretation*. John Wiley & Sons, 2015.

[65] Aristid Lindenmayer. „Mathematical Models for Cellular Interactions in Development I. Filaments with One-Sided Inputs“. In: *Journal of theoretical biology* 18.3 (1968), S. 280–299.

[66] Craig A. Lindley und Charlotte C. Sennersten. „Game Play Schemas: From Player Analysis to Adaptive Game Mechanics“. In: (2008). DOI: 10/c5cjmd.

[67] Ricardo Lopes und Rafael Bidarra. „Adaptivity Challenges in Games and Simulations: A Survey“. In: (2011). DOI: 10/d2dwc9.

[68] Ricardo Lopes, Elmar Eisemann und Rafael Bidarra. „Authoring Adaptive Game World Generation“. In: (2017).

[69] Luftbilddatenbank. *Kampfmittelvorerkundung - Luftbilder 2. Weltkrieg und Luftbildauswertung*. 2020. URL: https://www.luftbilddatenbank.de/main/index.php?webcode=luftbildauswertung (besucht am 15. 12. 2020).

[70] B. Magerko. „Adaptation in Digital Games“. In: *IEEE Computer* 41.6 (Juni 2008), S. 87–89. DOI: 10/bwvcbw.

[71] Benoit B. Mandelbrot und Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. Bd. 1. WH freeman New York, 1982.

[72] J. McGonigal. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. New York: Penguins Press, 2011.

[73] MDN. *Same-Origin Policy - Web Security | MDN*. 2021. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy (besucht am 25. 04. 2021).

[74] K. Mikami, B. H. D. Fernandez und K. Kondo. „Adaptable Game Experience Based on Player's Performance and EEG“. In: *2017 Nicograph International (NicoInt)*. 2017 Nicograph International (NicoInt). Juni 2017, S. 1–8. DOI: 10/ghgp8t.

[75] *Minecraft Official Site | Minecraft*. URL: https://www.minecraft.net/en-us/ (besucht am 02. 04. 2021).

[76] Hee-Seung Moon und Jiwon Seo. *Dynamic Difficulty Adjustment via Fast User Adaptation*. 2020. arXiv: 2006.15545.

[77] Benedikt Morschheuser u. a. „How to Design Gamification? A Method for Engineering Gamified Software“. In: *Information and Software Technology* 95 (2018), S. 219–237. DOI: 10/gc6dmr.

[78] Benedikt Morschheuser u. a. „How to Gamify? A Method for Designing Gamification“. In: *Proceedings of the 50th Hawaii International Conference on System Sciences 2017*. University of Hawai'i at Manoa, 2017. DOI: 10/ggsmqj.

[79] Lennart E. Nacke, Chris Bateman und Regan L. Mandryk. „BrainHex: A Neurobiological Gamer Typology Survey“. In: *Entertainment Computing* 5.1 (Jan. 2014), S. 55–62. ISSN: 18759521. DOI: 10/gft2p4. URL: https://linkinghub.elsevier.com/retrieve/pii/S1875952113000086 (besucht am 14. 12. 2020).

[80] Diana Oblinger. „Simulations, Games, and Learning“. In: *ELI White Paper* 1.1 (2006).

[81] Adam C. Oei und Michael D. Patterson. „Enhancing Cognition with Video Games: A Multiple Game Training Study“. In: *PLoS ONE* 8.3 (13. März 2013). Hrsg. von Joy J. Geng, e58546. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0058546. URL: https://dx.plos.org/10.1371/journal.pone.0058546 (besucht am 30. 07. 2020).

[82] Karin A. Orvis, Daniel B. Horn und James Belanich. „The Roles of Task Difficulty and Prior Videogame Experience on Performance and Motivation in Instructional Videogames“. In: *Computers in Human behavior* 24.5 (2008), S. 2415–2433. DOI: 10/d3c9bd.

[83] Alexandros Paramythis und Susanne Loidl-Reisinger. „Adaptive Learning Environments and E-Learning Standards“. In: *Second European Conference on E-Learning*. Bd. 1. 2003. 2003, S. 369–379.

[84] Yoav IH Parish und Pascal Müller. „Procedural Modeling of Cities“. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 2001, S. 301–308. DOI: 10/fnfjfr.

[85] Ken Perlin. „An Image Synthesizer“. In: *ACM Siggraph Computer Graphics* 19.3 (1985), S. 287–296. DOI: 10/bvwq8c.

[86] Ken Perlin. „Chapter 2: Noise Hardware“. In: (2002), S. 26.

[87] Johannes Pfau, Jan David Smeddinck und Rainer Malaka. „Deep Player Behavior Models: Evaluating a Novel Take on Dynamic Difficulty Adjustment“. In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, S. 1–6. DOI: 10/ggcj9t.

[88] photonstorm. *Getting Started with Phaser 3: Part 1 - Introduction - Learn - Phaser*. 14. Feb. 2018. URL: https://phaser.io (besucht am 25. 04. 2021).

[89] photonstorm. *Phaser 3.50.0 Released: Welcome to the Biggest Release of Phaser, Ever! New Renderer, Isometric Tilemaps, Layers, Post Processing Effects, Layers and Lots, Lots More!* 2020. URL: https://www.phaser.io/news/2020/12/phaser-350-released (besucht am 14. 04. 2021).

[90] Jan L. Plass, Bruce D. Homer und Charles K. Kinzer. „Foundations of Game-Based Learning“. In: *Educational Psychologist* 50.4 (2015), S. 258–283. DOI: 10/gfgxhw.

[91] Jan L. Plass und Shashank Pawar. „Toward a Taxonomy of Adaptivity for Learning“. In: *Journal of Research on Technology in Education* 52.3 (2020), S. 275–300. DOI: 10/gjrbws.

[92] Marc Prensky. „The Digital Game-Based Learning Revolution“. In: (2001), S. 19.

[93] Marc Prensky. „The Motivation of Gameplay“. In: *On the horizon* (2002). DOI: 10.1108/10748120210431349.

[94] Przemyslaw Prusinkiewicz. „Graphical Applications of L-Systems“. In: *Proceedings of Graphics Interface*. Bd. 86. 86. 1986, S. 247–253.

[95] Przemyslaw Prusinkiewicz. *The Algorithmic Beauty of Plants*. 1997.

[96] Rahmad Parlindungan Rangkuti. „The Effect of Hidden Object Game on the Students' Mastery in Vocabulary“. PhD Thesis. UMSU, 2019.

[97] Nicolas Rey u. a. „Detecting Animals in African Savanna with UAVs and the Crowds“. In: *Remote Sensing of Environment* 200 (1. Okt. 2017),

S. 341–351. ISSN: 0034-4257. DOI: 10/gb4rdh. URL: https://www.sciencedirect.com/science/article/pii/S0034425717303942 (besucht am 02. 04. 2021).

[98] K. Salen und E. Zimmerman. „Rules of Play: Game Design Fundamentals“. In: (2004).

[99] Jesse Schell. *The Art of Game Design: A Book of Lenses*. CRC press, 2008.

[100] Ulrike Schuckert. „Luftbildauswertung“. In: *Handbuch Naturschutz und Landschaftspflege* (2014), S. 1–14.

[101] David Williamson Shaffer u. a. „Video Games and the Future of Learning“. In: *Phi delta kappan* 87.2 (2005), S. 105–111. DOI: 10/gcsmgn.

[102] Ben Shneiderman u. a. „Creativity Support Tools: Report from a US National Science Foundation Sponsored Workshop“. In: *International Journal of Human-Computer Interaction* 20.2 (2006), S. 61–77. DOI: 10/dt6qt3.

[103] Valerie Shute und Fengfeng Ke. „Games, Learning, and Assessment“. In: *Assessment in Game-Based Learning: Foundations, Innovations and Perspectives*. 25. Mai 2012, S. 43–58. ISBN: 978-1-4614-3545-7. DOI: 10.1007/978-1-4614-3546-4_4.

[104] Valerie Shute und Brendon Towle. „Adaptive E-Learning“. In: *Educational Psychologist* 38.2 (Juni 2003), S. 105–114. ISSN: 0046-1520, 1532-6985. DOI: 10/fq2gs9. URL: 10.1207/S15326985EP3802_5 (besucht am 30. 07. 2020).

[105] Valerie J. Shute und Diego Zapata-Rivera. „Adaptive Educational Systems“. In: (2012). DOI: 10/ggcj8q.

[106] Ruben M Smelik u. a. „A Survey of Procedural Methods for Terrain Modelling“. In: (2009), S. 10.

[107] Jonas Steinbach. „Anwendung der Prozeduralen Generierung zum Erstellen einer Fortschrittsanzeige in Form wachsender Abbilder realer Baumarten in 2D“. In: (2017), S. 63.

[108] Alexander Streicher. „Adaptive und adaptierbare Wissensvermittlung“. In: (2020), S. 8.

[109] Alexander Streicher, Lukas Bach und Wolfgang Roller. „Usage Simulation and Testing with xAPI for Adaptive E-Learning“. In: *European Conference on Technology Enhanced Learning*. Springer, 2019, S. 692–695.

[110] Alexander Streicher, Natalie Dambier und Wolfgang Roller. „Task-Centered Selection of Learning Material“. In: (2012). DOI: 10/ggckbz. URL: 10.13140/2.1.3629.7767 (besucht am 30. 07. 2020).

[111] Alexander Streicher und Eva Lehmann. „Das Technologieakzeptanzmodell Für Kartenbasierte Lernspiele in Der Bildauswertung“. In: *DeLFI 2016–Die 14. E-Learning Fachtagung Informatik* (2016).

[112] Alexander Streicher, Sebastian Leidig und Wolfgang Roller. „Eye-Tracking for User Attention Evaluation in Adaptive Serious Games“. In: *Lifelong Technology-Enhanced Learning*. Hrsg. von Viktoria Pammer-Schindler u. a. Bd. 11082. Lecture Notes in Computer Science. Springer. Cham: Springer International Publishing, 2018, S. 583–586. ISBN: 978-3-319-98571-8 978-3-319-98572-5. DOI: 10.1007/978-3-319-98572-5_50. URL: http://link.springer.com/10.1007/978-3-319-98572-5_50 (besucht am 30. 07. 2020).

[113] Alexander Streicher und Wolfgang Roller. „Interoperable Adaptivity and Learning Analytics for Serious Games in Image Interpretation“. In: *Data Driven Approaches in Digital Education*. Hrsg. von Élise Lavoué u. a. Bd. 10474. Lecture Notes in Computer Science. Springer. Cham: Springer International Publishing, 2017, S. 598–601. ISBN: 978-3-319-66609-9 978-3-319-66610-5. DOI: 10.1007/978-3-319-66610-5_71. URL: http://link.springer.com/10.1007/978-3-319-66610-5_71 (besucht am 30. 07. 2020).

[114] Alexander Streicher und Wolfgang Roller. „Towards an Interoperable Adaptive Tutoring Agent for Simulations and Serious Games“. In: 2015, S. 4.

[115] Alexander Streicher und Jan D. Smeddinck. „Personalized and Adaptive Serious Games“. In: *Entertainment Computing and Serious Games*. Hrsg. von Ralf Dörner u. a. Bd. 9970. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, S. 332–377. ISBN: 978-3-319-46151-9 978-3-319-46152-6. DOI: 10.1007/978-3-319-46152-6_14. URL: http://link.springer.com/10.1007/978-3-319-46152-6_14 (besucht am 30. 07. 2020).

[116] Daniel Szentes, Bela-Andreas Bargel und Alexander Streicher. „Enhanced Value Benefit Analysis of Game Frameworks as a Tool for Digital Serious Game Development“. In: *ICERI2012 Proceedings*. IATED, 2012, S. 5634–5641.

[117] Daniel Szentes u. a. „Computer-Supported Training for the Interpretation of Radar Images“. In: (2008), S. 4.

[118] Phit-Huan Tan, Siew-Woei Ling und Choo-Yee Ting. „Adaptive Digital Game-Based Learning Framework“. In: *Proceedings of the 2nd International Conference on Digital Interactive Media in Entertainment and Arts*. 2007, S. 142–146. DOI: 10/dkkn2z.

[119] Tim Tijs, Dirk Brokken und Wijnand IJsselsteijn. „Creating an Emotionally Adaptive Game“. In: *International Conference on Entertainment Computing*. Springer, 2008, S. 122–133.

[120] Gustavo F. Tondello, Alberto Mora und Lennart E. Nacke. „Elements of Gameful Design Emerging from User Preferences“. In: (2017). DOI: 10/gf7h9d.

[121] John J. Trainor. „The Role of the Teacher in Quintilian“. In: (1947).

[122] Janne Tuunanen und Juho Hamari. „Meta-Synthesis of Player Typologies“. In: *Proceedings of Nordic Digra 2012 Conference: Games in Culture and Society, Tampere, Finland*. 2012.

[123] Mark Twain. *The Adventures of Tom Sawyer by Mark Twain*. Tauchnitz, 1876.

[124] Giel Van Lankveld u. a. „Incongruity-Based Adaptive Game Balancing“. In: *Advances in Computer Games*. Springer, 2009, S. 208–220.

[125] Luis Von Ahn und Laura Dabbish. „Labeling Images with a Computer Game“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2004, S. 319–326. DOI: 10/d5n25t.

[126] Wolfgang Walk, Daniel Görlich und Mark Barrett. „Design, Dynamics, Experience (DDE): An Advancement of the MDA Framework for Game Design“. In: *Game Dynamics*. Springer, 2017, S. 27–45.

[127] Nia WEARN und Esther MACCALLUM-STEWART. „“Case 88: Once Upon a Crime” Defining Hidden Object Games as Literary Narratives.“ In: Literature and Video Games: Beyond Stereotypes. University of St Andrews, 21. Mai 2018. URL: https://arts.st-andrews.ac.uk/lincs/scotland-2/ (besucht am 19. 04. 2021).

[128] Eric W. Weisstein. *Koch Snowflake*. 2021. URL: https://mathworld.wolfram.com/KochSnowflake.html (besucht am 01. 04. 2021).

[129] Wikipedia. *Puzzle Video Game*. In: *Wikipedia*. 15. Apr. 2021. URL: https://en.wikipedia.org/w/index.php?title=Puzzle_video_game&oldid=1017870427 (besucht am 18. 04. 2021).

[130] Wikipedia. *Schule*. In: *Wikipedia*. 22. Feb. 2021. URL: https://de.wikipedia.org/w/index.php?title=Schule&oldid=209082317 (besucht am 07. 03. 2021).

[131] wikipedia. *Copyleft*. In: *Wikipedia*. 6. Jan. 2021. URL: https://de.wikipedia.org/w/index.php?title=Copyleft&oldid=207323115 (besucht am 14. 04. 2021).

[132] wikipedia. *Spiel-Engine*. In: *Wikipedia*. 4. Juli 2020. URL: https://de.wikipedia.org/w/index.php?title=Spiel-Engine&oldid=201551697 (besucht am 13. 04. 2021).

[133] Shaomei Wu und Tao Lin. „Exploring the Use of Physiology in Adaptive Game Design“. In: *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*. IEEE, 2011, S. 1280–1283. DOI: 10/fg2pnq.

[134] Su Xue u. a. „Dynamic Difficulty Adjustment for Maximized Engagement in Digital Games“. In: *Proceedings of the 26th International Conference on World Wide Web Companion*. 2017, S. 465–471.

[135] Georgios N. Yannakakis und Julian Togelius. „Experience-Driven Procedural Content Generation“. In: *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 2015, S. 519–525. DOI: 10/ggk6b5.

[136] Georgios N. Yannakakis u. a. „Siren: Towards Adaptive Serious Games for Teaching Conflict Resolution“. In: (2010).

[137] Nick Yee. „Motivations for Play in Online Games“. In: *Cyber Psychology & Behavior* 9.6 (2007), S. 772–775. DOI: 10/cm9hfc.

[138] Alexander Zook u. a. „Skill-Based Mission Generation: A Data-Driven Temporal Player Modeling Approach“. In: *Proceedings of the The Third Workshop on Procedural Content Generation in Games*. 2012, S. 1–8. DOI: 10/gjrpmn.