Entwicklung eines allgemeinen Ansatzes für ein Framework zur dynamischen Anpassung der Schwierigkeit in digitalen Spielen

Natascha D. Kuncic

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Oktober 2012

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 7. Oktober 2012

Natascha D. Kuncic

Inhaltsverzeichnis

Vorwort

Das Balancing von Computerspielen war seit jeher ein sehr spannendes Thema für mich und seit einer Vorlesung in *Game Art and Level Design* war ich Feuer und Flamme für die adaptive Spielschwierigkeit. Obwohl mich das Thema sehr interessiert hat, war es schwer, sich durch die größtenteils trockenen Ausführungen der bisherigen Forschungsergebnisse zu arbeiten. Das Erste, was mir dabei aufgefallen ist, war, dass es bisher keinen erfolgreichen Ansatz gibt, der einfach und allgemeingültig einsetzbar ist.

Ich hoffe diese Arbeit kann Entwicklern von Independent Games als Inspiration dienen, die nicht die Möglichkeit haben ausgiebig in diesem Gebiet zu forschen. Ich hoffe damit einen kleinen Teil zur Verbesserung des Spielerlebnisses von diesen Spielen beizutragen und dadurch etwas für die vielen Stunden Spaß, die mir Independent Games bereits bereitet haben, zurückgeben zu können.

Ich möchte mich bei jenen bedanken, die mich bei der Entstehung dieser Arbeit unterstützt haben, allen voran bei meinem Betreuer Wolfgang Hochleitner, für die unendliche Geduld und konstruktiven Diskussionen und bei Roman Divotkey für seine Zeit und Motivation aber auch für das viele Hinterfragen des entwickelten Ansatzes. Auch viele meiner Kommilitonen waren an der erfolgreichen Ausarbeitung mit ihren kritischen Gedanken beteiligt und haben in zahlreichen Diskussionen das Für und Wider einiger Aspekte der dynamischen Schwierigkeitsanpassung hinterleuchtet. Ich bedanke mich bei Andreas Volk, für seine vielen Verbesserungsvorschläge, falls sich eigenwillige Formulierungen in meine Sätze eingeschlichen hatten.

Natürlich möchte ich auch meiner Familie, ganz besonders bei meinem Vater, danken, die mir in den intensivsten Phasen nicht übel genommen hat, dass ich mit meinen Gedanken immer bei meiner Arbeit und selten bei aktuellen Tischgesprächen war und die immer an mich geglaubt hat.

Es würde die Ressourcen dieses Blattes übersteigen, würde ich alle namentlich aufzählen, die mich auf meinem dieser Arbeit vorangegangenen Weg begleitet haben, aber auch ihnen möchte ich meinen tiefempfundenen Dank aussprechen.

Kurzfassung

Dynamische Schwierigkeitsanpassung (DDA für engl. *dynamic difficulty adjustment*) ist eine neue Technik in der Entwicklung digitaler Spiele zur Optimierung des *Flow* von Spielern, indem das System auf die erbrachten Leistungen des Spielers reagiert und die Herausforderungen an seine Fähigkeiten anpasst. Diese Technik ist bisher wenig verbreitet, daher versucht diese Arbeit, basierend auf dem Vergleich mehrerer Ansätze, einen allgemeinen, möglichst genreunabhängiges Ansatz in Form eines Framework zur Verwendung von DDA in sowohl einfachen als auch komplexen Spielen zu finden. In einer prototypischen Implementierung werden die essentiellsten Elemente des Frameworks getestet.

Abstract

Dynamic difficulty adjustment (DDA) is a new technique in the development of games in order to optimize *flow* for the players. It observes the player's behaviour and her performance and reacts accordingly to it by adjusting the difficulty of pending challenges or by lending a hand while facing a challenge.

This technique is not used often yet due to insufficient general approaches therefore this work is aiming to create an universal solution based upon previous attempts and their positive findings. The solution should find a use in simple and complex games alike. A prototype is testing basic ideas of this new solution and many demands of DDA in games are discussed in this work as well.

Kapitel 1

Einleitung

Dynamische Schwierigkeitsanpassung (kurz: *DDA*) ist eine noch sehr junge Technik in der Gestaltung von digitalen Spielen. Sie soll es in Zukunft ermöglichen, vor allem die Schwierigkeit eines Spiels so anzupassen, dass der Spielspaß maximiert bzw. Frustration und Langeweile beim Spielen minimiert werden. Die Möglichkeiten, die dynamsiche Schwierigkeitsanpassung bietet, sind vielfältig.

1.1 Motivation

Wer ein erfolgreiches Spiel herausbringen möchte, muss versuchen, ein möglichst großes Publikum zu erreichen. Mehrere Millionen Spieler (z. B. *World of Warcraft* mit 12 Mio. Abonnenten [1]) zeichnen wirtschaftlich erfolgreiche Spiele aus und sind ein Beispiel für die stetig wachsende Spielergemeinschaft.

Es ist die Aufgabe des Level-Designers, die Spielerfahrung zu optimieren, das heißt, die Herausforderungen des Spiels so anzuordnen und anzupassen, dass die Fähigkeiten des Spielers (nach dessen Wünschen) ausgereizt werden. Diese Aufgabe wird mit der zunehmenden Zahl und Diversität der Spieler immer schwieriger und trotzdem können Spieldesigner nur auf wenige Tools, ihre Intuition und ihren Erfahrungsschatz zurückgreifen, um Entscheidungen über das Balancing zu treffen [21].

DDA ist ein theoretischer Ansatz des Level-Designers, indem die Aufgabe des Balancing automatisiert wird, um den Spielspaß aller Spieler unabhängig von ihren Fähigkeiten, Eigenschaften und Erfahrungen zu erhöhen.

1.2 Zielsetzung

Ziel der Arbeit ist es, einen allgemeingültigen Ansatz mit einigen Richtlinien zur Erstellung von DDA zu entwickeln, der genreübergreifend arbeiten kann und einen hohen Wiederverwendungswert besitzt, da bisherige Arbeiten lediglich auf einige spezielle Eigenheiten von Genres eingehen. Ausserdem soll

diese Arbeit als Basis für eine Anregung zur Diskussion über dynamische Schwierigkeitsanpassung und ihre Ausprägung dienen können, indem sie die vielfältigen, u. a. in den bisherigen wissenschaftlichen Arbeiten vorgestellten Möglichkeiten zusammenfasst und miteinander in Beziehung stellt, da Bewusstseinsbildung in diesem Gebiet erforderlich ist, um die Konzepte des DDA nicht (unabsichtlich) falsch zu verwenden.

1.3 Aufbau der Arbeit

In Kapitel 2 werden einige sehr wichtige psychologische Aspekte des Spielers beleuchtet, die der Grund für die Notwendigkeit der dynamischen Schwierigkeitsanpassung sind. Dazu zählen die Erläuterung von *Flow* im Abschnitt 2.2.1, aber auch die Erklärung der Diversität der Spieler in 2.1.

Im Anschluss an die geschichtliche Entwicklung der Schwierigkeiten in Spielen (Abschnitt 2.3) wird im Kapitel 3 eine elementare Erläuterung und Zusammenfassung diverser in diesem Bereich bekannter Ansätze gegeben.

Folgend auf diese Einführung in das Gebiet der dynamischen Schwierigkeitsanpassung wird anhand der gefundenen positiven Aspekte einiger Ansätze ein eigener Ansatz in Kapitel 4 vorgestellt, der als Prototyp an einer existierenden Game-Engine getestet wurde (Abschnitt 4.3).

Am Ende der Arbeit (Kapitel 5) werden einige Sinnfragen zum Thema dynamische Schwierigkeitsanpassung gestellt, die bei der Verwendung ebendieser in eigenen Projekten beachtet und überdacht werden sollten. Die Liste der Themen ist bei weitem unvollständig und sollte nur als Denkanstoß fungieren.

Kapitel 2

Grundlagen

Dynamische Schwierigkeitsanpassung ist ein sehr komplexes Thema. Zum vollen Verständnis müssen viele psychologische und technische Aspekte gleichermaßen berücksichtigt werden.

2.1 Grundlagen zu Spielern

Obwohl der Gegenstand dieser Arbeit nicht die Psychologie und Motivation hinter Spielen ist, müssen die Grundelemente des *Spaß* am Spielen erörtert werden, da *dynamische Spielschwierigkeitsanpassung* als zentrale Aufgabe hat, den Spaß des Spielers aufrecht zu erhalten. Für diese Arbeit ist es ausreichend, sich mit einzelnen ausgesuchten Punkten zu befassen, für tieferes Verständnis ist weiterführende Fachliteratur anzuraten.

2.1.1 Die Motivatoren der Spieler

Es ist ein weitläufiges und spannendes Thema, warum Menschen dem Spielen zugeneigt sind. Neben diversen evolutionären Gründen, gibt es einen viel einfacheren Grund, dessen Sensibilität und Wichtigkeit durch die Definition von *Spiel* zur vollen Geltung kommt.

Bei der Begriffserklärung von *Spiel* (der englische Begriff *Game* ist noch weniger abgegrenzt), scheiden sich die Geister. Die einzige Übereinstimmung aller Versuche, *Game* zu definieren, besteht darin, dass es bisher keine, die Materie prägnant und vollständig definierende Beschreibung gibt, in [40] schreibt Schell:

> For the most part, we all know what we are talking about when we say games. It is true that the idea of that a game (or any term) means will vary a bit from person to person, but mostly we all know what a game is.

Spezialisten verschiedener wissenschaftlicher Gebiete definieren *Spiel* unterschiedlich, so ist *game* für Katie Salen und Eric Zimmermann [39]:

> [...] a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome.

Ein Spieler begibt sich ihnen zufolge in ein System mit einem künstlich erstellten Konflikt und vordefinierten Regeln, um schlussendlich zu einem messbaren Ergebnis zu kommen. Obwohl dies noch nicht besonders motivierend klingt, bereitet diese Umschreibung die Basis für einige weitere Erläuterungen bezüglich Spiele.

Sid Meier, Game Designer von *Civilization*, definiert *Spiele* als eine Serie von Entscheidungen [26], Cowley als Aktionsfolgen [13]. In [26] beschreibt Koster:

> Games are puzzles to solve, [...]. We learn the underlying patterns, grok them fully, and file them away so that they can be rerun as needed. The only difference between games and reality is that the stakes are lower with games.

Rückblickend kann man Puzzles bereits mit der Beschreibung von Salen und Zimmermann definieren – Konflikt, Regel und Ergebnis. Koster hat aber drei weitere, wichtige Aspekte hervorgehoben: den des Lernens, des Meisterns („grok them fully“) und den Unterschied zwischen der Spiel- und echten Welt.

Als hauptsächlichen Zweck von *Spielen* sieht Chanel in [10] „to provide emotional experiences such as fun and excitement“, auch Schell zitiert in [40] ein bekanntes Zitat von George Santayana:

> Play is whatever is done spontaneously and for its own sake.

Aus den vorangegangenen Ausführungen wird deutlich, dass Spielen als *freiwillige* Beschäftigung gesehen werden kann, in der sich Menschen einer Tätigkeit hingeben, die in den meisten Fällen keinen produktiven Zweck erfüllt. Sie unterwerfen sich ohne äusseren Zwang den geltenden Regeln [13, 39], um ein meist vordefiniertes, quantifizierbares Ziel zu erreichen – kooperativ mit anderen Spielern, konkurrierend oder auch alleine. Desweiteren besteht ein gemeinsamer Konsens, dass diese Spiele in simulierten, bzw. zeitlich und örtlich abgegrenzten, Umgebungen stattfinden [26, 37, 39] und sich die Spieler dessen bewusst sind.

Die Bereitwilligkeit des Spielers, sich den Regeln eines Spiels zu unterwerfen und Zeit und Energie aufzuwenden, das Spiel zu spielen (je nach Spielkonzept auch öfters), muss daher vom Spiel selbst gefördert und unterstützt werden. Es dem Spieler zusätzlich zu erschweren, in die Spielwelt einzutauchen, erhöht das Risiko, dass er es nicht mehr spielen wird.

Konkurrierendes Spielen

Der klassische Beweggrund eines Spieler ist der Trieb eine Fähigkeit zu meistern. Sei es gegen künstlich gesteuerte Gegner, reale Gegner oder die eigenen

Leistungen – der Drang immer besser zu werden, ist weit verbreitet. Vorderer [41] schreibt, dass *Wettbewerb* für den Unterhaltungsfaktor eines digitalen Spiels mitverantwortlich ist.

Wichtige Werkzeuge um diesen inneren Antrieb anzusprechen, den Spaß des Spielers und damit die Wiederspielbarkeit eines Spieles zu erhöhen sind u. a.:

- Einzigartige Spielmechanik,
- Herausforderungen,
- Siege,
- Belohnungen,
- Achievments,
- Turniere und Wettbewerbe,
- High-Score Listen.

Wie bereits in der Erklärung des Begriffs *Spiel* ausgeführt, sehen manche Forscher sogar einen essentiellen Aspekt eines *Spiels* darin, ein messbares Resultat zu erreichen [39]. Einen Sieger am Ende des Spiels zu küren, ist eine weit verbreitete Methode[1], eine andere ist das Abschließen des Spiels mit einem Punktestand, der sich dann in eine High-Score Liste eintragen lassen könnte[2].

Ebenfalls üblich ist die angemessene Belohnung nach überwundenen Herausforderungen in kurzweiliger (sogenannte „Power Ups“) oder langfristiger Form (z. B. Ausrüstungsgegenstände oder einen Speichercode).

Achievements sind spezielle Herausforderungen, die üblicherweise nur mit einem höheren Zeitaufwand gemeister werden können. Abseits möglicher Belohnungen oder Punkte sind die erlangten Auszeichnungen innerhalb der Spielergemeinschaft ein öffentliches Symbol für gesteigerte Spielfertigkeiten und verleihen dem Spieler ein höheres Ansehen, ähnlich den ersten Plätzen in einer High-Score Liste.

Digitale Spiele lassen sich in diesem Fall sehr gut mit Sportveranstaltungen vergleichen. Analog zu diesen stellen bei Turnieren die professionellsten Spieler ihre Fähigkeiten gegeneinander unter Beweis. Manche Veranstaltungen schreiben hohe Preisgelder bis in Millionenhöhe aus und auch Sponsoren investieren hohe Summen in professionelle Spieler.

[1] *League of Legends*, *Mario Kart* und viele andere Spiele bieten den konkurrierenden Parteien die Möglichkeit, über ihre Mitstreiter zu triumphieren)

[2] www.kongregate.com ist eine Plattform, auf der viele kleinere Spiele veröffentlicht werden und die Teilnehmer auf der Plattform in die jeweiligen High-Score Listen des Spiels eingetragen werden. Auch bekanntere, sogenannte Tripple-A Produktionen haben High Score bzw. sogenannte *Ladder* (z. B. *Starcraft II: Wings of Liberty* Listen

Kooperatives Spielen

Viele Spiele bieten zusätzlich zum üblichen Spielschema weitere Anreize, die mehrere Mitspieler erfordern und dem Spiel einen gesellschaftlichen Charakter verleihen. Die möglichen Interaktionen mit anderen Spielern bieten neue Komplexität und viele neue Herausforderungen, die zu meistern sind, aber auch viele gesellschaftliche Aspekte wie z. B. die Möglichkeiten um Hilfe zu fragen, zu Plaudern oder zu Handeln. Wesentliche Aspekte eines gemeinschaftlichen Spiels sind

- Kommunikation
- Handel
- Teambildung
- Sich ergänzende Spielrollen

Kooperativ heißt aber nicht nur, dass mehrere Spieler zeitgleich miteinander spielen, sondern, dass sie sich gegenseitig unterstützen, um ein gemeinsames Ziel zu erreichen.

Das bedeutet, für das Gemeinwohl des Teams auch eine andere Rolle im Spiel einzunehmen, als man ursprünglich wollte oder auf langsamere Teammitglieder zu warten bzw. auf einem anderen Weg Rücksicht zu nehmen (z. B. in *World of Warcraft*, *Counter Strike: Source*, und andere Team-Spiele). Im Einzelspielermodus oder im Wettkampf mit anderen Spielern würden derartige Aktionen und Verhaltensweise nicht erfordert sein.

Turniere sind auch mit digitalen Teamspielen sehr beliebt, so hat beispielsweise Dota2[3] bereits in der Beta-Phase einen Wettbewerb veranstaltet, dessen Siegerteam 1 Million Dollar Preisgeld gewonnen hat.

2.1.2 Spielereigenschaften profilieren

Ein Spieler kann durch viele Eigenschaften beschrieben werden. Viele sind bereits in die Entwicklung von Spielen mit eingeflossen, beispielsweise das Alter, das Geschlecht, Hobbies oder Interessen. Für behindertenfreundliche Versionen, die manche Spielentwickler anbieten, wird zusätzlich auf weitere Eigenschaften wie das Farbsehen oder die physiologischen Einschränkungen eines möglichen Spielers Rücksicht genommen. Für dynamische Schwierigkeitsanpassung sind einige weitere Eigenschaften eines Menschen von großer Bedeutung, die, um die Spielerfahrung besser individualisieren zu können, berücksichtigt werden sollten. Je nachdem, wie stark die Anpassung ausgeprägt sein soll, können auch weitere, in dieser Arbeit nicht behandelte Besonderheiten des Menschen miteinbezogen werden, z. B. der Spieler-Typ, wie in [2, 8] nachvollzogen werden kann.

[3] http://www.dota2.com/

Ziele und Ansprüche

Bei Spielern ist einer der stärksten Unterschiede ihr Ehrgeiz und die Bereitschaft, verschiedene Ressourcen für das Spiel aufzuwenden [28]. Die Ziele, die sich ein Spieler setzt, variieren sehr stark und sind essentiell für die Ansprüche, die er an das Spiel und seine Leistung stellt.

Dieser Unterschied ist einer der wichtigsten Gründe, warum dynamische Schwierigkeitsanpassung Spiele für eine breitere Masse gleichermaßen interessant machen könnte. Die Anpassungen des Spiels an den Spieler sollten ihn im Erreichen seiner Ziele und erfüllen seiner Anforderungen unterstützen.

Im folgenden wird die klassische Kategorisierung der Spieler nach Spielintensität erläutert, die im Fachbereich gebräuchlich ist und daher mit englischen Bezeichnungen nach folgenden Definitionen in dieser Arbeit verwendet wird:

Casual Gamer: Typische *Casual Gamer* spielen Spiele nur zum Zeitvertreib und wollen sich dahergehend nicht mit schweren Spielmechaniken auseinander setzen. Kurzweilige und wenig anspruchsvolle Spiele sind ihre Wahl, da sie ihren Anspruch auf Einfachheit am ehesten erfüllen.

Sie spielen, um zu gewinnen, sie spielen um einen leichten Sieg zu erringen. Einen Gegner wollen sie schnell besiegen und Rätsel schnell lösen. Nur selten wollen sie sich länger mit einer Herausforderung beschäftigen.

Midcore Gamer: Der *Midcore Gamer* ist der Spieler im typischen Mittelfeld. Er zeichnet sich durch weit gestreute Spielinteressen aus und zeigt Enthusiasmus für viele Genres. Im Vergleich zum *Casual Gamer* möchte er sich auch mit komplexen Spielmechaniken auseinandersetzen und spielt gerne, jedoch nicht sehr lange.

Er spielt um zu siegen und will sich seinen Sieg erarbeiten. Er wird gerne herausgefordert und kann damit umgehen, an bestimmten Aufgaben länger zu beschäftigen. Trotzdem wird er nach mehreren Versuchen meistens aufgeben.

Hardcore Gamer: *Hardcore Gamer* spezialisieren sich häufig auf ein paar Spiele, versuchen diese zu meistern und ihre Fähigkeiten zu perfektionieren. Sie sind enorm ehrgeizig und stellen hohe Anforderungen an sich und jedes Spiel. Daher sind sie sehr überlegt mit ihrer Spielauswahl und oft an menschlichen, ebenbürtigen Kontrahenten interessiert, mit denen sie sich auf Wettkämpfen und kleineren Turnieren messen.

Hardcore Gamer spielen für die Herausforderung. Sie wollen sich ihren Sieg hart erkämpfen. Ein Spiel, welches sie nicht genügend herausfordern kann, fesselt sie nicht. Aponte schreibt in [4]:

> For the hardcore players, who prefer more challenge, to mostly optimize their satisfaction, [...]

Pro Gamer: *Pro Gamer* unterscheiden sich nicht wesentlich von *Hard-core Gamern*, ausser, dass sie ihre Leidenschaft zum Beruf gemacht haben oder damit eine Nebeneinkunft zu beziehen. Sie lassen sich ihre Turnierteilnahme von Sponsoren bezahlen und wollen die teilweise sehr hohen Preisgelder gewinnen.

Newbie: Der *Newbie* kann aus jeder bisher genannten Spielertypen stammen. *Newbie* steht lediglich für *new comer* und bezeichnet jemanden, der in dem aktuellen Spiel noch neu ist und aufgrund der fehlenden Erfahrung die Wahrscheinlichkeit, Fehler zu machen, höher ist.

In der Eingewöhnungsphase für ein Spiel bevorzugen die meisten Menschen niedrigere Herausforderungen, erklärt Klimmt in [25]. Je nach Lernkurve und Erfahrung in ähnlichen Spielen kann die *Newbie*-Zeit von sehr kurz bis sehr lang ausfallen.

Erfahrung

Die Erfahrung eines Spielers in einem bestimmten Spiel oder auch Spielgenre, beispielsweise *Mario Land 1* im Genre *Jump and Run*, hat großen Einfluss auf sein Verhalten in diesem oder ähnlichen Spielen. Anfänger machen noch viele Fehler, die beispielsweise das Verhalten im Spiel oder die Steuerung des Spiels beinhalten. Wer *Mario Land 1* beherrscht, wird in *Mario Land 2* weniger Probleme mit der Steuerung oder den Spielmechaniken haben, so wie Goetschalckx et al. in [16] schreiben:

> [...] as not every person is the same, with some people being fast learners and others taking more time, and some people having more experience with certain kinds of games, the prior expected abilities (before the player gets any experience playing this specific game) and the rate at which these change depend on this too.

Auch Yun et al. verwenden u. a. die Erfahrung des Spielers als Ausgangsbasis für ihre Adaptierungen [46].

Lernkurve

Ritter [36] leitet seine Arbeit über Lernkurven mit folgenden Worten ein:

> Most tasks get faster with practice. This is not surprising because we have all seen this and perhaps know it in some intuitive sense.

Selbst über unterschiedliche Arbeiten und Personen hinweg beschreiben die meisten Lernkurven eine negative Beschleunigung – anfangs sind große Fortschritte für wenig Aufwand zu erwarten, wobei dieser Fluss immer weiter stagniert, wie Ritter festhält [36]:

> It [the learning curve] suggests that practice always helps improve performance, but that the most dramatic improvements happen first. Another implication is that with sufficient practice people can achieve comparable levels of performance.

Laut Ritters „full power law formula“,

$$Time = MinTime + (TrialNumber + PrevPractice)^{-(constant for Task)}$$

, welche eine der beiden möglichen mathematischen Beschreibungen einer Lernkurve darstellt, gibt es eine bestimmte Minimumzeit, für eine Aufgabe, die nie erreicht werden, sich ihrer aber angenähert werden kann.

Diese Lernkurve ist aufgrund mehrere komplexer Vorgänge für jede Person individuell. Ritter [36] fasst mehrere Studien zusammen:

> Even when problems are of comparable difficulty, subjects may use different strategies. [...] The power law applies across strategies, but the fit is better to each strategy [...], or even an individual's strategies [...]

Es ist anzunehmen, dass es optimiertere Strategien gibt, ein Problem zu lösen. Daher kann man annehmen, dass ein Spieler, welcher in einem neuen Spiel ähnlichen Aufgaben gegenübersteht wie jenen, die er bereits gemeistert hat, in der Regel eine bessere Lernstrategie entwickelt hat, um diese Aufgabe zu bewältigen. Man kann einem erfahrenen Spieler in einem bestimmten Genre daher bessere Lerneigenschaften zussprechen, als einem Anfänger ohne Vorkenntnisse.

2.2 Grundlagen zum Spielerlebnis

Die Essenz des Spieles ist das Erlebnis, das den Spieler in seinen Bann zieht, ihn festhält und nicht mehr loslässt. Nicht ganz uneigennützig, da dies auch der Grund ist, warum Spiele gekauft und gespielt werden, haben bereits einige die Quintessenz dieser Spielerfahrung gesucht.

Thomas Malones Model beschreibt drei wesentliche Aspekte vom Spielvergnügen, *fantasy*, *curiosity* und *challenge*, wobei letzterer direkt mit Spielschwierigkeit in Verbindung stehen soll [30].

Einen wichtigen Teil des Spielvergnügens macht laut Ryan [38] das Gefühl von *Kompetenz* aus. Ob sich ein Spieler kompetent fühlt oder nicht, hängt wieder von der Spielschwierigkeit ab, mit der er konfrontiert wird. Auch andere Arbeiten referenzieren auf Schwierigkeit als Schlüsselelement zum Spielvergnügen.

Es steht also fest, dass die Spielschwierigkeit einen nicht unwesentlichen Beitrag zum Spielspaß beiträgt. Eines der wichtigsten Konzepte, das dieses Phänomen auf psychologischer Ebene zu erklären sucht, ist *Flow* [4, 14, 21].

2.2.1 Flow

Flow stellt eine Größe dar, die das Spielempfinden des Spielers beschreiben soll und die bisher, aufgrund der direkten Abhängigkeit von dem nicht eindeutig definierten und nicht messbaren Parameter „Schwierigkeit“ [4], ebenfalls nicht eindeutig messbar ist. Im Feld der Spielentwicklung und auch der dynamischen Schwierigkeitsanpassung ist *Flow* ein Konzept, das im Mittelpunkt steht. Es ist die Antwort auf die Frage, *wofür brauchen wir dynamische Schwierigkeitsanpassung?*

Bereits aus der Definition des Wortes *Spiel* geht hervor, dass ein hohes Maß an persönlichem Engagement von Seiten des Spielers entscheidend ist, um eine Tätigkeit als Spielen zu interpretieren. Dieses persönliche Einlassen auf das Spielgeschehen geht mit bestimmten Erwartungen (siehe Abschnitt 2.1.1) an das Spiel einher, deren Erfüllung durch die Interaktion mit diesem Spiel erreicht werden kann. Ausgewogene Herausforderungen und positive Rückmeldung stärken die emotionale Beteiligung des Spieler, wohingegen durch falsch gesetzte Schwierigkeit Frustration, Angst oder Langeweile ausgelöst werden können, die das Engagement des Spielers negativ beeinflussen.

Dieser empfindliche Zustand zwischen allen unerwünschten Stresszuständen, der das Optimum der emotionalen Beteiligung des Spielers am Spielgeschehen darstellt, wird im Englischen als *Flow* oder auch als *being in „the Zone“* [12] bezeichnet. Anthony Youssef schreibt in [45]:

> Many game designers face the challenge of providing their players with a balance between challenge and experience.

Starke emotionale Beteiligung tritt dann auf, wenn die Fähigkeiten eines Spielers die Anforderungen einer Herausforderung erfüllen und die Herausforderung diese Fähigkeiten ausreizt, so Chanel in [10]. *Flow* erhält dadurch den Status des zentralen Elements der Spielentwicklung.

Komponenten des Flows

Flow ist laut Csikszentmihalyi aus mehreren Hauptkomponenten zusammengestellt [14]. Einige dieser Hauptkomponenten hängen hauptsächlich vom Spieler ab und sind nicht direkt zu beeinflussen, u. a. Konzentration auf die aktuelle Aufgabe, Verlust der Selbstwahrnehmung oder eine veränderte Zeitwahrnehmung.

Andere von Csikszentmihalyi definierte Elemente sind besser zu manipulieren und können von Spielentwicklern gezielt eingesetzt werden. Sie sind die Hauptkriterien, um *Flow* zu kreieren [9]:

- Die herausfordernde Aktivität, die bestimmte Fertigkeiten erfordert,
- Direkte und augenblickliche Rückmeldung,
- Das Gefühl von Kontrolle.

Kontrolle selbst ist ein wieder sehr weit gefasster Begriff, der im Kapitel 5 weiter zur Verwendung kommen wird. Die kurze Definition in [35] „autonomy, power, freedom“ ist nur ein kurzer Einblick in die Vielfalt der Bedeutung, weiterführend behandeln [17, 40] das Thema.

Herausforderungen: Die essentiellste Anforderung ist, Aufgaben für den Spieler zu erstellen, die ausgewogen Herausforderung und Fähigkeiten des Spielers kombinieren. Das Gleichgewicht, welches diese Aufgaben representiert, definiert die Qualität der Spielerfahrung, die schlussendlich zur emotionalen Bindung an das Spiel beiträgt. Auch andere Arbeiten vermerken, dass die ausgeglichene Kombination von Schwierigkeit und Fähigkeit die wesentliche Aufgabe des Spieldesigners ist ([11, 12, 20, 42] und viele mehr).

Kontrolle: Das Gefühl der Kontrolle steht in Verbindung mit der passenden Schwierigkeit, aber auch mit der Möglichkeit der Selbstbestimmung, beispielsweise durch diverse Einstellungsmöglichkeiten, und auch mit den Reaktionen des Spiels.

Unerwünschte Zustände

Neben dem *Flow* werden zwei weitere wichtige Emotionszustände definiert, die, im Gegensatz zum Hauptzustand *Flow*, nur mit Maß eingesetzt werden sollten und, in der Regel, als Dauerzustand unerwünscht sind. Die Bereiche von Flow werden in Abbildung 2.1 grafisch dargestellt.

Angst, Stress: engl. als *Anxiety* bezeichnet – Unangenehme *Stresszustände* oder *Angst* treten häufig auf, denn der Spieler mit Herausforderungen konfrontiert wird, in denen die Wahrscheinlichkeit, mit den ihm zur Verfügung stehenden Fähigkeiten zu gewinnen, äusserst gering sind. So schreiben Yun et al. [46]:

> [...] some players will become anxious if challenge exceeds their game-play abilities.

Stresszustände können dabei aus vielerlei Gründen entstehen, u. a. wenn der Spieler das Gefühl hat, die Kontrolle zu verlieren oder mit „unfairen“ Situationen umgehen muss. Grundsätzlich gilt, dass Spannung und ein begrenzter Stresszustand erwünscht, aber die Grenze zur *Frustration* bei jedem Spieler unterschiedlich ist und in der Regel nur selten überschritten werden sollte – mit Ausnahme jener Spiele (und -genres) in denen diese Stresszustände als Stilmittel eingesetzt werden.

Langeweile: engl. *Boredom* – Der emotionale Zustand der *Langeweile* entsteht häufig durch Unterforderung, beispielsweise eine „durch lange Gewohn-

heit langweilig gewordene, ohne innere Beteiligung ausgeführte Tätigkeit“[4] (nach C. Fisher in [15]):

> [...] an unpleasant, transient affective state in which the individual fells a pervasive lack of interest in and difficulty concentrating on the current activity.

oder aber auch durch nicht ausreichend schwierige Herausforderung und Rätsel (Yun in [46]):

> [...], other players will lose interest if challenge levels are not sufficient.

Langeweile ist einer der möglichen Ursachen für eine schwache emotionale Bindung an das Spielgeschehen und kann der entscheidende Grund sein, dass ein Spiel nicht vollständig oder wiederholt gespielt wird.

2.3 Grundlagen zur Spielschwierigkeit

Wie in Abschnitt 2.2 erläutert, ist Spielschwierigkeit eines der Schlüsselelemente bei der Erstellung von Spielen.

2.3.1 Schwierigkeit und Leveldesign

Ursprünglich war es die Aufgabe von Spiel- und Level-Designern, die Schwierigkeit des Spiels durch das Gestalten des entsprechenden Levels zu definieren. Durch Hindernisse, Gegner und Rätsel lag es am Spieldesigner, sowohl die Emotionen des Spielers zu formen, als auch das Spielerlebnis und den Spannungsbogen, den er durchleben soll.

Es lässt sich streiten, ob sich Level-Designer hauptsächlich auf ihre Intuition verlassen oder auf Feldstudien zurückgegreifen [21]. Fest steht, dass beim Erstellen der Schwierigkeit eines Levels ein Referenzspieler verfügbar sein muss. Dieser Referenzspieler ist eine Idee, stellvertretend für die entsprechende Zielgruppe, und enthält beispielsweise Alter und Geschlecht sowie eine gewisse Vorstellung über seine Erfahrung und seine Lernkurve (siehe Abschnitt 2.1.2). Anhand dieser kreirt der Leveldesigner die Schwierigkeitskurve des Levels. Die Referenzspieler werden im Test-Stadium des Spiels verifiziert, indem viele Probespieler das Spiel spielen, ihre Meinung dazu in einer vordefinierten Form abgeben und diese Evaluierung in die weitere Entwicklung und Veränderung des Levels und dessen Schwierigkeit einfließen. Aponte schreibt in [5]:

> For now, game difficulty adjustment is a subjective and iterative process. Level and game designers create a sequence of challenges and seit their parameters to match their chosen difficulty curve.

[4] http://services.langenscheidt.de/fremdwb/fremdwb.html, Stichwort „Routine“

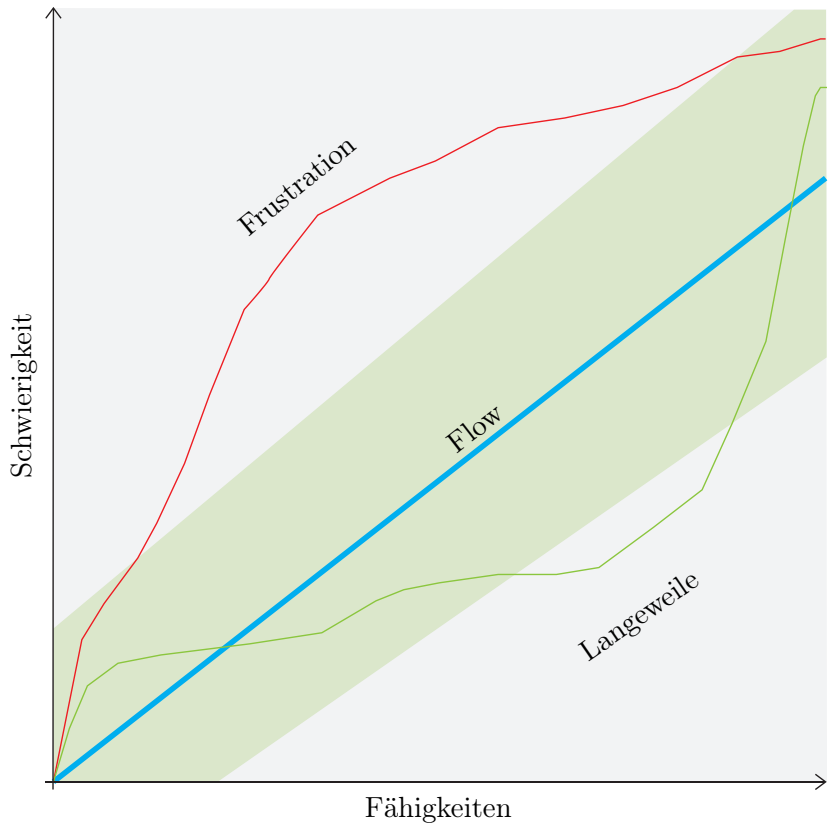

Abbildung 2.1: Flow ist das ausgewogene Verhältnis von Schwierigkeit zu Fähigkeit (grüner Bereich). Frustration entsteht bei zu hoher Schwierigkeit, Langeweile bei Unterforderung der Fähigkeiten. Die dicke blaue Linie ist der ideale Verlauf mit einer gleichmäßiger Steigerung der Schwierigkeit im Verhältnis zu den Fähigkeiten. Der Spannungsbogen erzeugt allerdings eine ungleichmäßige Steigerung der Schwierigkeit und kann u. a. zu den rot und grün gezeichneten Verläufen führen. Grafik nach [11, Fig. 1] und [12, Fig. 1].

Unter dem Begriff *Level* kann man daher die vom Level-Designer vordefinierte Schwierigkeitkurve verstehen, ebenso wie unter *Schwierigkeit* die feste Abhängigkeit der *Schwierigkeit* vom sie umgebenden Level.

2.3.2 Statische Schwierigkeit

> At the beginning, most existing game difficulty increases steadily along the course of the game (either in a smooth linear fashion, or through steps represented by the levels). [42]

Zu Beginn der Geschichte der digitalen Spiele wurden hauptsächlich statische Level erstellt, beispielsweise *Super Mario Land 1* (Screenshots in Abb. 2.2) oder *Pong* boten lediglich ein Level. Wie bereits Bin in [42] schreibt,

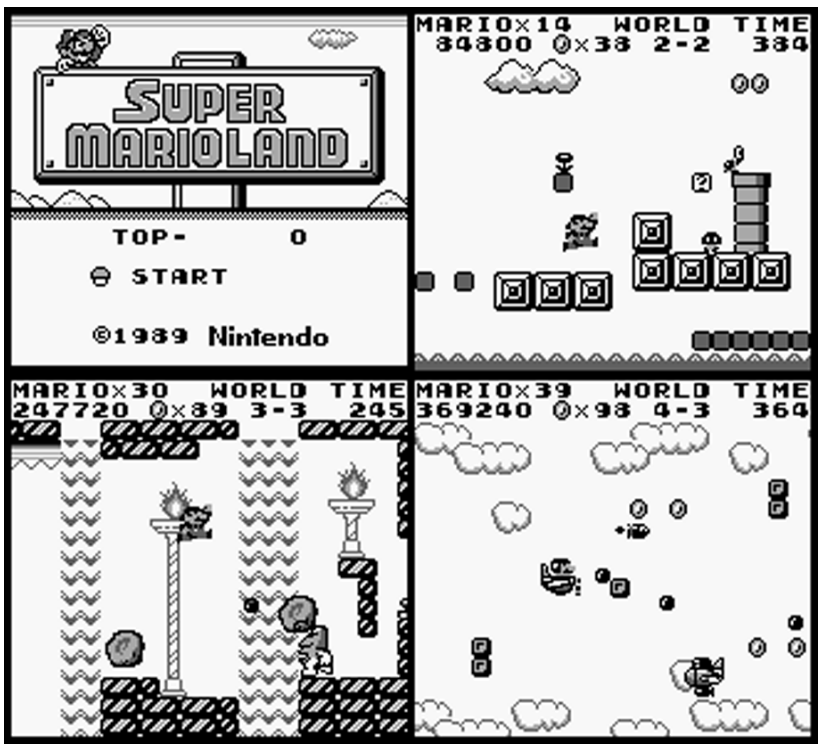

Abbildung 2.2: *Super Mario Land* hat nur ein Schwierigkeitslevel, der das gesamte Spiel definiert. Quelle: http://3dfxon.files.wordpress.com/2011/06/supermarioland_1.png.

steigt die Schwierigkeit dabei in einer vordefinierten Kurve an. Sie ist angepasst an den vorher angesprochenen Referenzspieler, den der Spieldesigner als Vorlage hatte.

Weicht ein Spieler nun von dieser vorgegebenen Lernkurve ab, verlässt er den Bereich des idealen Spielvergnügens und beginnt in eine der unerwünschten Zonen des Flow-Modells (siehe Abschnitt 2.2.1) abzudriften, erklärt Bin [42]:

> But this can lead to a frustrating experience for both experienced and inexperienced gamers, as they attempt to follow a preselected learning or difficulty curve.

Damals gab es für den Spieler keine Möglichkeit innerhalb des Spiels, diesen Umstand zu umgehen – aber auch nicht so viele Alternativen, wie heute am Markt sind.

2.3.3 Diskrete Schwierigkeitsstufen

Auf die Problematik, dass viele Spieler mit der angebotenen Schwierigkeitsstufe nicht zufrieden waren, wurde mit der Erstellung mehrerer Schwierigkeitsstufen reagiert, wie auch Xinyu et al. in [29] anmerken:

> The difficulty level can be adjusted only at the beginning of the game by selecting a difficulty level (Easy, Medium, Hard, and Insane) [...]

Das Spiel wird dann in der gewählten Schwierigkeit durchgespielt, ohne dass die Schwierigkeit währenddessen gewechselt werden könnte.

Ein Fortschritt ist die dem Spieler gebotene Möglichkeit, seine Präferenzen zu wählen, wie die Abbildung 2.3 aus dem Android-Spiel *Birds on a Wire* beispielhaft zeigt. Er kann selbst entscheiden, ob er sich dem *hard*-Modus stellen oder lieber eine gemütliche Runde auf *easy* spielen möchte. Dies ermöglicht es dem Spieler erstmals, seinen persönlichen Ehrgeiz auszuleben (siehe Abschnitt 2.1.2).

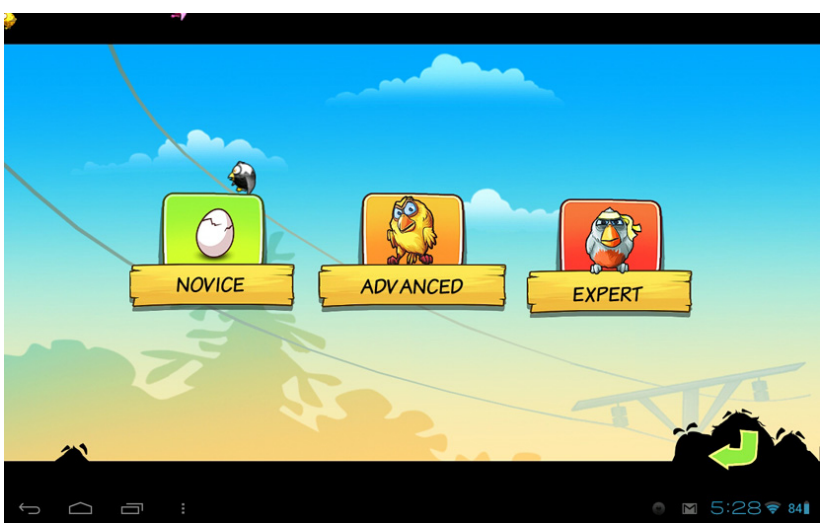

Abbildung 2.3: Viele Spiele stellen mehrere Schwierigkeitsstufen zur Wahl. Beispiel: *Birds on a Wire*. Quelle: http://cdn.androidtapp.com/wp-content/uploads/2012/07/Birds-Difficulty.jpeg.

Nachteilig ist vor allem das scheinbar zufällige Benennungsverfahren der Spielhersteller und die ungenormten Bezeichnungen. Obwohl *easy*, *medium* und *hard* vermeintlich eindeutig zu verstehen sind, hat jeder Hersteller und Spieler unterschiedliche Vorstellungen der dahinter versteckten Schwierigkeit. Oftmals kommt es vor, dass für einen Spieler die *medium*-Schwierigkeit zweier unterschiedlicher Spiele zu völlig unterschiedlichen Ergebnissen führt: eines ist ihm weiterhin viel zu leicht während das andere ihn vor unüberwindbare Herausforderungen stellt.

Es ergibt sich wieder das Problem, dass für jede Schwierigkeit ein Referenzspieler mit vordefinierter Lernkurve und Erfahrung herangezogen wird und alle Spieler, die davon abweichen, in der einen oder anderen Form Probleme haben werden, die erwartete Spielempfindung zu erleben. Dennoch, durch die Wahlmöglichkeit wird dieses Problem ein wenig abgeschwächt. Der Spieler erhält das Gefühl der Kontrolle über das Spielgeschehen, was laut Csikszentmihalyi für den optimalen Gefühlszustand während des Spiels mitverantwortlich ist (siehe Abschnitt 2.2.1).

2.3.4 Halbautomatische Schwierigkeitsgenerierung

Unter *halbautomatischer Schwierigkeitsgenerierung* kann man die Generierung mehrerer Levels mittels parametrisierbarer Algorithmen verstehen. Jennings schreibt in [23]:

> These games typically work by fitting together hand-authored level chunks into random combinations.

Dabei stellen Parameter wichtige Regler für die zu produzierende Schwierigkeit des Levels dar. Der Algorithmus erstellt dann Level, die alle notwendigen Informationen, Herausforderungen und Wege in einer fertigen, vordefinierten Form enthalten. Wichtig zu wissen ist, dass sich diese Level nicht während der Spielzeit verändern bzw. nach der Veröffentlichung nicht mehr verändert werden.

Durchaus möglich ist die Nachbearbeitung eines generierten Levels bei der *offline Generierung* (siehe nächster Abschnitt) durch einen Level-Designer, um die Spielbarkeit des Levels zu garantieren und die emotionale Erfahrung während des Levels zu optimieren. Vor allem weniger komplexe, nicht für alle Spezialfälle durchdachte Algorithmen könnten in diesen Bereichen Probleme verursachen und häufiger die Editierung durch einen Level-Designer benötigen.

Die genauen Algorithmen sind üblicherweise ein gut gehütetes Firmengeheimnis, sodass keine konkreten Vergleiche mehrerer Algorithmen durchgeführt werden können. Dennoch gibt es unterschiedliche Arbeitsweisen von Algorithmen, die in dieser Arbeit grob in *Offline-Generierung* und *Online-Generierung* kategorisiert werden und im Folgenden thematisiert werden sollen.

Offline Generierung

Die Offline Generierung von Levels dient hauptsächlich der Arbeitsoptimierung bei der Erstellung von Level. Bei anfänglich höherem Aufwand für die Erstellung des passenden Algorithmus sinkt die verwendete Zeit für die Erstellung eines neuen Levels drastisch, auch wenn es bei weniger ausgereiften Algorithmen zu Fehlern im Level kommen und ein nicht-spielbares Level

entstehen kann. Da das erstellte Level jedoch noch nicht veröffentlicht ist, kann der Algorithmus verbessert oder die bestehende Möglichkeit der Nachbearbeitung durch einen Level-Designer genutzt werden – anders als bei der *Online Generierung*. Trotz dieser Nachteile kann der Aufwand pro erstelltem Level schon durch einfache Algorithmen drastisch verringert werden, wobei die Einsparungen mit steigender Levelanzahl immer größer wird. Gerne verwendet werden derartige Algorithmen bei Rätsel- und *Jump and Run*-Spielen, wie das *iPhone*-Spiel *Unblock Me*, oder aber auch bei Spielen, die wenig Speicher zur Verfügung haben, z. B. das 1984 veröffentlichte Videospiel *The Elite*.

Die vielfache Generierung verschiedener Levels, d. h. in der Regel auch unterschiedliche Schwierigkeitsgrade, bietet den Spielern eine größere Auswahl und dem Entwickler eine Erleichterung. Die einzelnen Schwierigkeitsgrade ziehen sich allerdings nicht über das gesamte Spiel fort, sondern stellen nur ein Level, d. h. einen Teilbereich des Spiels, dar. Die durch die Level-Reihenfolge vorgegebene Schwierigkeitskurve gleicht daher wieder dem einzelnen statischen Schwierigkeitsgrad aus vergangenen Tagen (siehe Abschnitt 2.3.2). Die optimale Schwierigkeit für den Spieler ist nur in einem geringen Abschnitt des Spiels zu finden, wodurch auch durch diese Methode nicht gewährleistet ist, dass sich der Spieler während dem Großteil des Spielverlaufs im optimalen Flow-Status befindet.

Online Generierung

Als online generierte Level werden in dieser Arbeit jene bezeichnet, die zu Laufzeit des Spiels nach der Auswahl der Schwierigkeit nach einem Muster generiert werden. Die Schwierigkeit liegt in der perfekten Ausarbeitung des Algorithmus, sodass er jedes Mal ein spielbares Level produziert, und in der Optimierung der Laufzeit eben jenes Algorithmus, da die Erstellung in annehmbarer Zeit auf unterschiedlichsten, leistungsschwachen Rechnern der Spieler vonstatten gehen muss. Diese beiden Eigenschaften stehen sich zum Teil gegenseitig im Weg.

Der Vorteil von prozeduraler Levelgenerierung online wird von Jennings in [23] hervorgehoben:

> [...] a primary motivation for procedural level generation has been to create improved replayability for a game which can be accomplished by giving the player a new level each time through.

So gibt es einige Beispiele für Spiele, die bei der Veröffentlichung statt statischer Level Algorithmen verwenden, um die Umgebung und Herausforderungen zu generieren, u. a. die *Diablo*-Reihe, *Spore* und *Civilization*.

Diese Algorithmen erstellen jedes Mal eine „neue Welt“, wenn der Spieler diesen Spielbereich lädt. Solange er sich in diesem befindet, bleibt das Gebiet genau gleich, die Gegner greifen von den gleichen Stellen aus an und

die Wege verändern sich nicht. Wird der selbe Bereich verlassen und erneut betreten, erstellt der Algorithmus eine neue Kombination der Bausteine dieses Bereiches, versteckt Schätze und Gegner an anderen Stellen und erreicht dadurch einen höheren Wiederspielwert.

Kapitel 3

Dynamische Schwierigkeitsanpassung – DDA

Obgleich die *Online Generierung* von Levels bereits viel Wiederspielwert und Dynamik bewirkt, werden die Levels immer nach dem selben Muster erstellt. Sobald sie generiert sind, sind sie in ihrer Schwierigkeit fixiert und unveränderbar.

Dynamische Schwierigkeitsgenerierung (kurz: *DDA* für *dynamic difficulty adjustment*) ist sehr ähnlich zur online Generierung von Levels, mit dem Unterschied, dass sich die Level und die Schwierigkeit auch während der Laufzeit verändern können, so Missura [31]:

> [...] the game should possess an ability to change the difficulty of it's challenges on the fly, in an online fashion.

Eigentlich ist es die Aufgabe des Level-Designers, für einen einzelnen Spieler die Schwierigkeit und daher Spielerfahrung zu optimieren, d. h. die Herausforderungen so anzupassen, dass die Fähigkeiten des Spielers nach dessen Wünschen ausgereizt werden. Doch diese Aufgabe wird mit der zunehmenden Zahl an Spielern immer schwieriger und eine Individualisierung der Spielerfahrung immer wichtiger. So ist es in der Branche der unterhaltenden Computerspiele zwar von Vorteil, den Flow (siehe Abschnitt 2.2.1)des Spielers, wie in Abb. 3.1 gezeigt, zu optimieren. In sogenannten *serious games*, das heißt Spielen, die einen seriösen Zweck haben, z. B. Lern- und Therapiespiele, ist es hingegen von essentieller Bedeutung, den Spieler mit bewältigbaren, aber herausfordernden Aufgaben zu konfrontieren, um die Resultate zu maximieren – und immer einen Überblick über seine aktuellen Fähigkeiten zu haben.

Obwohl dieses Konzept enorm zukunftsträchtig ist und hohes Potential hat, wird es bisher nur vereinzelt eingesetzt. Es gibt bereits einige wenige

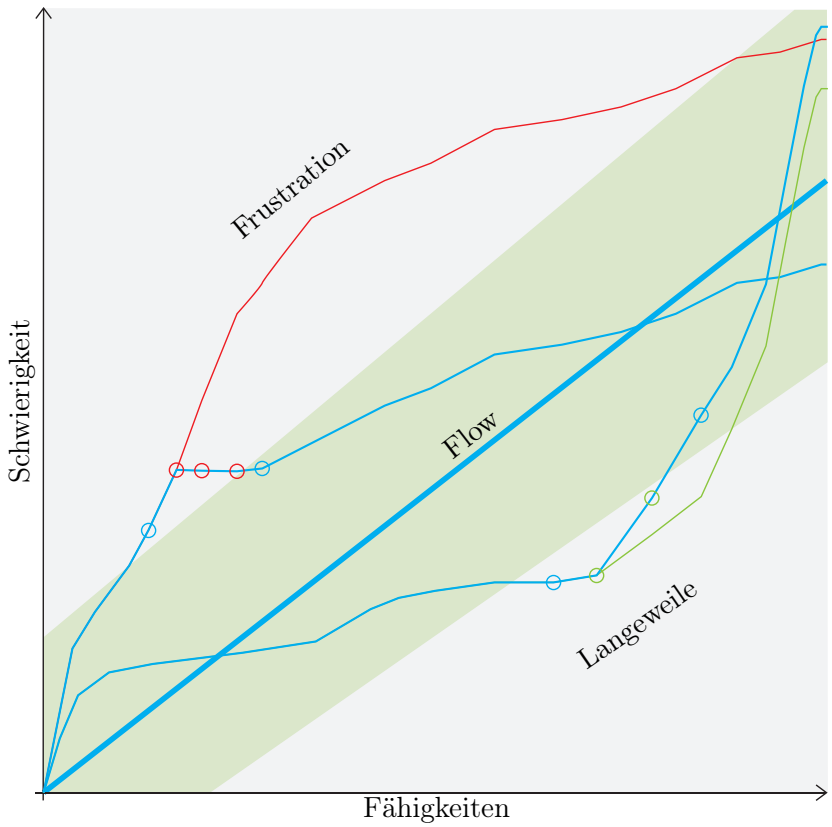

Abbildung 3.1: Unterschiedliche Spannungsbögen und Lernkurven können zum Ausbrechen aus dem Flow führen (rote und grüne Kurve). Kleinere Ausreisser, die erkannt werden, müssen nicht sofort korrigiert werden (blaue Kreise entsprechen der Definierung „Flow gewährleistet“), allerdings sollte ein gutes DDA-System zu große Abweichungen erkennen und solange korrigieren, bis der Flow-Bereich wieder erreicht ist (rote und grüne Kreise stellen Änderungsbedarf dar, der letzte blaue Kreis zeigt an, dass das System erkannt hat, dass der Spieler sich wieder im Flow-Bereich befindet). Ab diesem Zeitpunkt kann das System wieder mit der regulären Spannungskurve fortfahren. Grafik nach [12, Fig. 1].

Spiele, die versucht haben, die Schwierigkeit dynamisch zu gestalten und an den Spieler anzupassen, z. B. *Left 4 Dead*[1] oder *The Elder Scrolls IV: Oblivion*[2].

Es mangelt bisher an Erfahrung, Methoden und Richtlinien, sodass die

[1] *Left 4 Dead*'s DDA-System modifiziert beispielsweise die Anzahl der Gegner und die Stellen, an denen sie erscheinen je nach erwarteter Spielerperformance.

[2] *Oblivion* skaliert die Gegnerschwierigkeit direkt proportional zum Spielerlevel. *Oblivion* versetzt damit Spieler, die Skillpunkte in sekundäre Charakterfähigkeiten investieren, in die unangenehme Situation, dass die Gegner stärker als der eigene Charakter sind und erzwingt daher eine Skillpunkteverteilung auf primäre Fähigkeiten.

Verwendung von DDA eher experimenteller Natur ist, jedoch haben sich bereits einige Forscher dem Bereich DDA gewidmet und versucht, die unterschiedlichen nötigen Elemente genauer zu definieren und geeignete Methoden zu für

- das Erfassen der Spielerfähigkeit, und
- die Veränderung des Levels,

zu finden. In den folgenden Abschnitten werden unterschiedlichste Ansätze verschiedenster Perspektiven zum Thema DDA vorgestellt, die viele Bereiche des DDA-Systems abdecken.

3.1 Physiologische Indikatoren für Schwierigkeit

Yannakakis et al. [43, 44], Nacke und Lindley [34], Chanel et al. [10], Kuikkaniemi et al. [27] erforschen die physiologischen Eigenschaften des Spielers und die Möglichkeit über Rückschlüsse auf seine Wahrnehmung von Schwierigkeit in Computerspielen.

3.1.1 Können psychophysiologische Daten beim Spielen entstehende Emotionen widerspiegeln?

Nacke beschäftigt sich mit der Übereinstimmung von psychophysiologischen Daten mit den tatsächlichen Gefühlen der Spieler.

In einem kleinen Experiment, in dem Spieler drei verschiedene, vordefinierte Level nacheinander durchspielen und anschließend bewerten sollten, wurden sie beim Spielen beobachtet (Gesichtsmuskelstimuli, Hautwiderstand und Video). Er stellte fest, dass die während einer Spielsitzung ermittelten psychophysiologischen Daten direkt vom Spiel beeinflusst werden und daher der Umkehrschluss von den psychophysiologischen Daten auf das Empfinden des Spielers zulässig erscheint.

3.1.2 Direkte und indirekte Rückführung von psychophysiologischen Signalen

Kuikkaniemi et al. haben die biologische Reaktion von Spielern und die Auswirkung deren Rückführung in das Spiel auf deren Spielerlebnis in einem *First Person Shooter* erforscht.

Sie unterscheiden zwei Arten des Gebrauchs von biologischen Messdaten: jene Systeme mit indirekten Rückmeldungen und jene mit direkten Rückmeldungen. Bei ersteren sind sich die Testpersonen nicht darüber bewusst, dass ihre gemessenen physiologischen Daten dazu verwendet werden, das Spiel zu beeinflussen. Laut Kuikkaniemi hatte dieses System daher nur geringen Einfluss auf das Erlebnis des Spielers, eine Adaptierung des Spiels an die Eigenschaften des Spielers wurde vom Spieler kaum festgestellt.

Weitaus interessanter ist das System, in dem sich die Spieler bewusst ist, dass die Daten dazu benutzt werden, um das Spiel zu beeinflussen. Die Spieler lernten, die adaptierbaren Eigenschaften ihres Avatars mittels der biologischen Signale zu steuern und waren, laut Kuikkaniemi, wesentlich mehr in das Spiel eingebunden. Kuikkaniemi führt dies darauf zurück, dass die Identifikation mit dem Avatar dadurch stärker wird, ihn mittels biologischen Signalen zu steuern (z. B. die Avatar-Laufgeschwindigkeit).

Die Praxistauglichkeit seiner Methodik wird – zumindest für den aktuellen Zeitpunkt und unter den gerade vorherrschenden technischen Gegebenheiten – stark bezweifelt. Da sein System auf Messungen der Herz- und Atemfrequenz und andere, nur mit speziellen, medizinischen Geräten messbare Daten aufbaut, kann davon ausgegangen werden, dass diese Grundlage in naher Zukunft nicht von der breiten Masse zu Verfügung gestellt werden kann. In [27] wird festgehalten:

> Psychophysiological signals can be effectively used in systems built for automatic recognition of human emotion.

und daher, sollten die erforderlichen Daten früher oder später durch technische Neuerungen auch von der breiten Masse wie alltäglich zur Verfügung gestellt werden können, wird dies eine enorme Verbesserung der Leistung von digitalen, adaptiven Systemen bringen können.

3.1.3 Klassifizierung anhand periphärer Daten und EEG

Genauigkeit der Klassifizierung anhand periphärer Daten und EEG Auch Chanel baut in [10] zusätzlich auf die Daten von Elektroenzephalografien (EEG[3]) auf, um die Genauigkeit der Bestimmung von Emotionen des Spielers zu erhöhen.

Chanel extrahiert viele Merkmale aus den sogenannten „periphären“ Daten, die u. a. folgenden Gruppierungen zuzuordnen sind:

- Hautwiderstand,
- Blutdruck,
- Herzrate,
- Brusthöhlenerweiterung,
- Hauttemperatur.

Zusätzlich dazu benutzte er EEG-Daten, um die Genauigkeit der Klassifizierung zu vergleichen.

Chanel schreibt, dass bei einem Zeitraum von 300 Sekunden die erreichte Genauigkeit aus periphären Daten mit einen Wert von maximal 58% höher war als die Genauigkeit aus EEG-Daten. Beim Unterteilen der Daten in

[3] „Mittels EEG wird die vom Gehirn ausgehende elektrische Aktivität von der Kopfhaut abgeleitet, man bezeichnet EEG deswegen auch als Hirnstromkurve.“ http://www.neurologie.uni-goettingen.de/index.php/elektroenzephalographie-eeg.html

kürzere Zeiträume verschlechterten sich alle Ergebnisse bei weitem, wobei die Daten aus dem EEG dann eine bessere Genauigkeit vorwiesen als die der periphären Messungen.

Alles in Allem fasst Chanel zusammen, dass anhand einiger Merkmale die drei definierten Stadi (siehe Tab. 3.1) definiert werden können. Beim Erkennen der Stadi sieht er noch viel Verbesserung, jedoch merkt er an, dass es reichen würde, *Langeweile* und *Angst* zu erkennen und dass diese beiden Stati ausreichen, um das Spiel korrekt zu adaptieren. Er geht noch weiter und behauptet, dass es sogar ausreichen könnte, nur *Langeweile* zu verhindern (d. h. zu erkennen), da ein zu schwieriger Schwierigkeitslevel das Lernen und die Kompetenz des Spielers fördert und nicht unbedingt korrigiert werden muss.

	pressure	*pleasure*
Langeweile	low pressure	low pleasure
Flow	higher pressure	high pleasure
Angst	high pressure	low pleasure

Tabelle 3.1: Die Definition der drei Stadi des Empfindens nach [10].

3.1.4 Physikalische Daten

Yannakakis et al. verwenden in ihrem Ansatz ein Spiel namens *Bug Crasher*, in dem die Spieler – in diesem Fall Kinder – auf jene Spielfelder steigen/treten müssen, auf denen Käfer auftauchen. Aus den Daten des Experiments, die gesammelt wurden, versuchte Yannakakis die kleinstmögliche Kombination an Eigenschaften herauszufinden, die gesammelt werden müssen, um mit einer trainierten ANN eine möglichst akkurate Klassifikation der Spieler zu gewährleisten.

Mit gerademal 4 der 9 gemessenen Daten, nämlich dem Durchschnitt der Reaktionszeit, der Varianz des Drucks (durch das Auftreten auf das Spielfeld), die Anzahl der Interaktionen und die Informationen über die besuchten Spielfelder konnten die Vergleichsdaten am besten zugeordnet werden.

Anzumerken ist hierbei, dass das System auf die physikalische Komponente, d. h. den Druck auf das Spielfeld, angewiesen war, um gute Ergebnisse zu erzielen, andernfalls hätte es eine andere Kombination der zur Verfügung stehenden Daten wählen können, in der weder der Durchschnitt noch die Varianz des Drucks verwendet würden. Dies zeigt, dass die Genauigkeit der Einschätzung eines Spielers mittels physikalischen Informationen erhöht werden kann.

Daraus lässt sich die Hypothese aufstellen, dass mit entsprechendem phy-

+	−
Psychophysiologische und physikalische Informationen können die Präzision der Einstufung der Emotionen des Spielers erhöhen.	Aktuell besitzen kaum herkömmliche Haushalte Gerätschaften, um die notwendigen Daten zur Verfügung zu stellen. Das Ausliefern der Spiele mit notwendigem Equippment könnte teuer werden.
Diese Zusatzinformationen könnten vor allem bei therapeutischen Spielen zur Verwendung kommen, da es meistens in einem kontrollierten Umfeld stattfindet und medizine Geräte zur Verfügung gestellt werden könnten.	Das Aufbereiten der Daten und Extrahieren relevanter Informationen sowie das Ziehen der Rückschlüsse daraus ist noch nicht sehr ausgereift.

Tabelle 3.2: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [10, 27, 34, 43, 44], Abschnitt 3.1 „Physiologische Indikatoren für Schwierigkeit“.

sikalischen Messdaten die Genauigkeit einer Einschätzung von Spielern und ihrem aktuellen Immersionsgrad wesentlich verbessern lässt. Da die meisten Spiele auf Konsolen und PCs gespielt werden, wo es in den meisten Fällen keine Messung über Intensität der Eingabe gibt, stellt sich die Erhebung der notwendigen Daten als sehr schwieriges Unterfangen heraus, da zusätzliche Geräte nötig wären.

3.2 Eine Definition von Schwierigkeit

Aponte versucht, die Schwierigkeit in Videospielen, die sie als nicht präzise definierbaren und nicht messbaren Parameter sieht, in ihren Arbeiten [4–6] zu evaluieren. So schreibt sie in [4]:

> The currently widely adopted approach is to calculate a score, based on a heuristic function [...]

und sie merkt an, dass die entsprechenden Heuristiken oftmals nicht verifizierbar sind und sie daher ein falsches und verzerrtes Bild von „Schwierigkeit“ liefern können.

Ihr Vorschlag ist, die Schwierigkeit einer Herausforderung als die Wahrscheinlichkeit zu definieren, mit der diese Aufgabe schaffbar ist. Dabei ist sie sich dessen bewusst, dass auch die Wahrscheinlichkeit erst berechnet werden muss. Dabei stellt sie fest, dass es nicht einfach sein wird, die Schwierigkeit aus statistischen Beobachtungen zu extrahieren, da sich der Wert

Schwierigkeit fortlaufend ändert und daher nicht über längere statistische Beobachtung gewonnen werden kann.

Um das Problem zu lösen, definiert sie zweierlei Arten an Herausforderungen: jene, die die Spielmechanik ausmachen, die *core challenges* (später von ihr *abilities* des Spielers genannt), und jene, die sich daraus zusammensetzen, genannt *composite challenges* (von ihr später auch als *challenge* bezeichnet).

Core challenges sind z. B. das Zielen im *First Person Shooter* oder der Sprung im `Platformer` und wird so oft in so kurzen Zeiträumen wiederholt, dass hier davon ausgegangen wird, dass das Level der Schwierigkeit für den Spieler konstant ist und daher die Wahrscheinlichkeit, eine *core challenge* zu schaffen (oder eben nicht), auf statistischem Wege erhoben werden kann.

Auf diese Weise ist es möglich, ein Spieler-Modell zu erstellen, in dem die Fähigkeiten des Spielers für jede *core challenge* (d. h. *ability*) eingetragen ist. Daraus lässt sich eine Wahrscheinlichkeit ableiten, eine *(composite) challenge* erfolgreich zu beenden.

In [5] erweitert Aponte diese Theorie:

> [...] a video game challenge is a dynamic rule based system with two outcomes *WIN* or *LOSE*. At a given time t a challenge can be in one of the four possible states *NOT STARTED*, *IN PROGRESS*, *WIN*, *LOSE* [...]

Zusammen mit der Definition zweier Herausforderungssets (den *core challenges* und den daraus zusammengesetzten Herausforderungen) können mittels der in [5] beschriebenen Regeln die Erfolgswahrscheinlichkeit bei zusammengesetzten Herausforderungen berechnet werden.

Es ergibt sich der Nachteil der Datenflut, wenn der statistische Ansatz gewählt wird. Aponte merkt an, dass für die detaillierteren Einstufungen (detaillierter als „gut, mittel schlecht“) einer Fähigkeit (*abilty*) auch wesentlich mehr Daten notwendig sind, um statistisch relevant zu sein. Ob Vergleiche unter den Spielern leicht möglich sind, ist nicht genau abzuschätzen, sodass Achievement-Systeme oder High-Scores zum Problemfall werden könnten. Dennoch muss gesagt werden, dass der Ansatz, die Schwierigkeit relativ zu den Spielerfähigkeiten zu definieren, einige Erleichterungen bringen kann.

3.3 Die Schwierigkeit in Real Time Strategy Spielen messen

Hagelbäck verwendet in [18] ein *Real Time Strategy* Spiel, um seinen Ansatz der Spielschwierigkeitsmessung zu evaluieren. Dazu verwendete er mehrere verschiedene Bots mit unterschiedlichen Verhaltensweisen und lies die Tester im Anschluss an ein Spiel gegen den Bot einen Fragebogen ausfüllen.

+	−
Die Schwierigkeit als *Wahrscheinlichkeit zu Gewinnen* zu definieren, bringt den Vorteil, dass dieser Wert immer im Verhältnis zum Spieler steht und leicht reguliert werden kann.	Die Datensammlung für statistische Auswertung könnte ausufern.
Es wäre anhand der *core challenges* und einiger Regeln möglich, das System eigene Herausforderungen und Sequenzen generieren zu lassen.	
Vergleiche zwischen Spielern könnten auf Ebene der *core challenges* stattfinden.	Der Versuch, mehrere Spieler miteinander zu vergleichen, wird komplexer (schwieriger/unübersichtlicher) sein als ein einfacher Score.

Tabelle 3.3: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [4–6], Abschnitt 3.2 „Eine Definition von *Schwierigkeit*“.

Im gesamten Spielverlauf wird regelmäßig eine Variable, von Hagelbäck $isScore$ genannt, berechnet, die die aktuelle Schwierigkeit, oder besser formuliert Dominanz, beschreibt, wobei das Gleichgewicht der Kontrahenten (Bot–Bot oder Bot–Spieler) durch eine 0 dargestellt würde, egal, wie die Berechnung dahinter aussieht. In Hagelbäcks Experiment wurden die Einheiten und die Hälfte ihrer verbleibenden Stärke für beide Seiten aufsummiert – wer mehr hatte, war im Vorteil.

Aus der aktuellen Punktezahl wurde eine Variable $s \in [0, 1]$ aktualisiert, die die Aktivität der gegnerischen Einheiten steuerte (dies war die Schwierigkeitsregulierung während des Experiments). Sollte also der Spieler dominieren, war s größer, als wenn der Gegner dominierte, sodass für jede von gegnerischen Einheiten erzeugte Zufallszahl r mit höherer Wahrscheinlichkeit gilt: $r \leq s$ und sich die Einheit daher bewegen durfte.

In seinem Experiment testete Hagelbäck ausserdem unterschiedliche Verhaltensweisen der AI. Zum einen verwendete er zwei Bots mit statischer Schwierigkeit zum Vergleich. Die anderen drei unterscheideten sich in der Lernrate (Geschwindigkeit, mit der sich s veränderte) und in der voreingestellten Schwierigkeit. Herausgestochen ist allerdings der dynamische Bot, welcher zwar während des Spiels eine normale Lernrate und Schwierigkeit hatte, sich aber dumm stellte sobald der Spieler nur mehr 5 Einheiten besaß und den Spieler immer gewinnen ließ.

+	−
Der Punktestand für das aktuelle Kräfte-Gleichgewicht ist auch für eine komplexere Spielsituation anwendbar.	Da in einer komplexeren Spielsituation mehrere Faktoren berücksichtigt und ausbalanciert werden müssen, könnte die Formelerstellung einige Zeit in Anspruch nehmen.
Spieler mögen es (meist) nicht, wenn man sie kurz vor dem Ende des Spiels absichtlich und ganz offensichtlich gewinnen lässt.	

Tabelle 3.4: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [18], Abschnitt 3.3 „Die Schwierigkeit in Real Time Strategy Spielen messen“.

Der anschließende Fragebogen stellte klar, dass gerade letzterer Bot als besonders einfach und langweilig empfunden wurde. Die beiden dynamischen Bots, bei denen nicht immer gewonnen wurde, wurden definitiv bevorzugt, wodurch angenommen werden kann, dass dies allgemeine Gültigkeit hat.

3.4 Dynamische Schwierigkeitsanpassung durch Motivationserkennung

Hocine et al. beschäftigen sich mit der Entwicklung von sogenannten *serious games*, worunter meist Spiele mit ernstem bzw. sinnvollem Hintergrund verstanden werden. In diesem Fall wird an therapeutischen Spielen gearbeitet, um Schlaganfallspatienten beim Rehabilitationsprozess zu helfen.

Gerade im therapeutischen Bereich ist die Einstellung des Spiels auf den Patienten besonders wichtig, betont Hocine in [19]. Hocine sieht in den therapeutischen Spielen viel Potential, die Patienten zum Üben motivieren zu können und den Therapeuten teilweise zu ersetzen.

Die wichtige Rolle des Therapeuten muss also das Spiel übernehmen können [19].

> Therapist changes for instance the objective difficulty according to the patient's performances, tiredness, health conditions and motivation.

Der Therapeut muss für den Patienten so arbeiten, wie ein optimales DDA-Model für den Spieler funktionieren muss. Lediglich ist die Notwendigkeit für

DDA und vor allem die Notwendigkeit dessen Genauigkeit und Zuverlässigkeit in *serious games*, wozu auch Lernspiele für SchülerInnen zählen, weitaus höher als im Unterhaltungssektor, um nicht nur den Motivationsfaktor zu erhalten, sondern auch Gerechtigkeit zwischen den Schülern zu gewährleisten.

Um den Therapeuten zu ersetzen, setzt Hocine auf die Erkennung von Motivation. Als Motivation definiert sie in [19]:

> [...] motivation can depend on several factors such as: success and stimulation, feedback on activities, autonomy, self-esteem, meaningfulness and variety of tasks.

wovon einige Punkte bereits im Kapitel 2.2.1 „Flow“ zu finden sind.

Motivation und Activation

Hocine's Ansatz ist, die Motivation durch den richtigen Grad der Stimulation zu erreichen, d. h. den Patienten ausreichend und dennoch nicht zuviel zu fordern. Als Maß dient, welches *Activation*, das das Verhältnis der Wahrscheinlichkeiten, eine Aktivität erfolgreich zu beenden oder zu scheitern, ist. Es findet wie folgt Verwendung:

$$Activation(n) = \begin{cases} log\frac{Pr_n(success)}{1-Pr_{n-1}(success)} & \text{if } Pr_n(success) \neq 1 \\ 1 & \text{otherwise} \end{cases} \tag{3.1}$$

$$\sigma(n) = \sum_{i=1}^{n} Activation(i) \tag{3.2}$$

$$\delta(n) = Activation(n) - Activation(n-1) \tag{3.3}$$

wobei gilt:

$n \ldots$ Zeitpunkt
$Pr_{n-1}(success) \ldots$ Wahrscheinlichkeit aller bisherigen Spiele
$\sigma \ldots$ Summe aller *Activation*
$\delta \ldots$ Tendenz zu einem bestimmten Zeitpunkt

Wahrscheinlichkeitsmatrix

Weiters führt sie eine Matrix für das Spiel-Level ein, in dem an jedem Punkt für das Spiellevel die Wahrscheinlichkeit, eine Aktivität auszuführen, gespeichert ist. Diese Matrix wird aus den Patientendaten und den Herausforderungsgraden des Levels generiert, sodass der Patient bei hoher Herausforderung viele Aktivitäten auszuführen hat, bei niedriger Herausforderung die

Wahrscheinlichkeit, eine Aktivität ausführen zu müssen, eher gering ist.

$$D(i,j) = \begin{cases} \frac{1}{\#\{A(x,y) \geq d\}} & \text{if } (A(i,j) > d) \\ 0 & \text{if } (A(i,j) < d)) \end{cases} \tag{3.4}$$

Die Matrix wird dann gefüllt und an allen Positionen, wo die Wahrscheinlichkeit erfolgreich zu sein, mindestens d ist, wird eine gleichmäßige Wahrscheinlichkeit, eine Aktivität auszuführen, eingetragen.

Adaptierungsbedingung

Die Anpassung der Schwierigkeit erfolgt bei Hocine nach jeder Spielrunde. Die Instabilität der Motivation (Formeln 3.5 und 3.6 mit binärer Ausgabe (`true`, `false`)) im Zeitfenster ω wird errechnet, sodass diese gemeinsam mit der globalen Tendenz (Formel 3.7) innerhalb des Zeitraumes zu einer Entscheidung der Schwierigkeitsveränderung führt.

$$S -_{local}(n) = \frac{\#\delta(i) < 0, i \in [n-\omega, n]}{\omega} < \omega^{+}_{local} \tag{3.5}$$

$$S +_{local}(n) = \frac{\#\delta(i) \geq 0, i \in [n-\omega, n]}{\omega} > \omega^{-}_{local} \tag{3.6}$$

$$T_{global}(n) = Trend(\sigma(1), \sigma(n)) \geq \omega_{trend} \tag{3.7}$$

3.5 Schwierigkeitsauswahl anhand des Partially Observable Markov Decision Process

Goetschalckx et al. benutzten in [16] POMDP , um ein therapeutisches Spiel für Demenzkranke dynamisch anpassen zu können.

Wie schon am Anfang dieses Kapitels erklärt, müssen die Systeme, die DDA benutzen, die Fähigkeiten des Spielers (er-)kennen, um darauf reagieren zu können. Goetschalckxs Methodik nimmt an, dass dies nicht direkt möglich ist, sondern die Fähigkeiten des Benutzers, seine Frustrationsschwellen und seine Erfahrung bzw. Lerncharakteristik nur indirekt beobachtet werden könnten, als schätzbare Werte, gewonnen aus den beobachtbaren Interaktionen des Benutzers mit dem System. Das System kennt daher nur einen ungefähren, geschätzten Wert der Eigenschaften und Fähigkeiten des Spielers und reagiert, sobald dieser geschätzte Wert in ein kritisches Verhältnis zu Optimalwerten gelangt.

Goetschalckx nennt die nicht direkt beobachtbaren Variablen, „unsichtbare Variablen“ :

> [...] the hidden variables are the actual abilities of the player (which will change as the player learns to play the game better),

+	−
Pro Position im Spiel die Wahrscheinlichkeit, eine Aktion auszuführen, als Matrix	
Der Schwellwert für die Erfolgswahrscheinlichkeit definiert individuellen Schwierigkeitsgrad	Technik schließt nur Aktivitäten aus, die den Schwellwert nicht erreichen – keine Veränderung im Spielgeschehen selbst
Level-Designer behält viel Kontrolle über das Level	
Bisheriges Verhalten wird mit einberechnet	Änderungen werden nur ründlich vorgenommen

Tabelle 3.5: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [19], Abschnitt 3.4 „Dynamische Schwierigkeitsanpassung durch Motivationserkennung“.

> the enjoyment (including boredom and frustration levels), and the player type, as well as the stretch which indicates how hard the previous level was, given the estimated abilities of the player.

Sie sind jene Werte, die notwendig zu wissen sind, um die Individualisierung eines Spieles zu ermöglichen. Für die beobachtbaren Interaktionen werden folgende Beispiele genannt:

- Zeit, die für eine Aufgabe benötigt wurde;
- Leben, die bei einer Aufgabe verloren wurden;
- Punkte, die durch eine Aufgabe erarbeitet wurden.

Goetschalckx nimmt also an, dass bereits eine so hohe Abstraktion der Interaktion des Spielers mit dem Spiel einen Anhaltspunkt für die Fähigkeiten des Spielers liefern kann. Vorrauszusetzen sind dabei Referenzen, um die Leistung des Spielers in ein Verhältnis zu stellen, beispielsweise durchschnittliche Rundenzeiten aus der Testphase des Spiels.

Wie viele andere merkt Goetschalckx an, dass die Fähigkeiten des Benutzers schwanken können und das System jederzeit darauf zu reagieren hat. Zwar ist dies kein neues Konzept, in seinem Fall war es dennoch besonders wichtig, da er auf jede Veränderung reagieren sollte, da bei therapeutischen Prozessen und Spielen andere Motivatoren vorherrschen als bei unterhaltsamen Spielen. Für die digitale Spielindustrie wird es notwendig sein, angemessen auf die Schwankungen zu reagieren, wobei „angemessen“ noch nicht eindeutig definiert ist.

+	−
Fähigkeiten und Eigenschaften des Spielers können nicht direkt beobachtet werden	
Interaktionen mit dem Spiel sind beobachtbar und dienen als hervorragende Basis um Rückschlüsse auf Fähigkeiten des Spielers zu ziehen	Die enorme Abstraktion der Interaktion ermöglicht keinen genauen Rückschluss auf Stärken und Schwächen des Spielers
Erkenntnis, dass Lernkurve durch zu hohe Komplexität in keiner direkten Korrelation zu Beobachtungen steht	

Tabelle 3.6: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [16], Abschnitt 3.5 „Schwierigkeitsauswahl anhand Partially Observable Markov Decision Process“.

Goetschalckx et al. zeigen, dass die Lernkurve bzw. Lernrate des Spielers durch das Beobachten seiner Performance geschätzt werden kann. Dennoch wird angemerkt:

> [...] due to the complexity of the dynamics of the system, a clear correspondence between the ability improvement frequency and the learning rate is not readily available.

Es kann angenommen werden, dass die meisten Spiele ein Level an Komplexität aufweisen, dass die selbe Problematik auftreten und keine Übereinstimmung zwischen der geschätzten und tatsächlichen Lernrate vorliegen würde. Daher ist weiters anzunehmen, dass ein System, welches sich selbst nur anhand einer geschätzten Lernkurve verändern würde, keine zufriedenstellenden Ergebnisse liefert.

3.6 Dynamische Schwierigkeitsanpassung mittels Artificial Neural Networks

Xinyu et al. erläutern in [29] die Möglichkeit, Artificial Neural Networks (kurz ANN) mittels UCT[4] zu trainieren.

UCT sehen sie für sehr wenig komplexe Spiele als Möglichkeit an, dynamisch den Schwierigkeitsgrad des Spieles zu verändern. Da diese Methode

[4]Upper Confidence bound for Trees

+	−
ANN können unterschiedlich gut und möglicherweise auch auf unterschiedliche Strategien trainiert werden.	Als Beispiel diente eine nochmals vereinfachte Form des Pac-Man, dessen Spielprinzip an und für sich schon wesentlich weniger Komplexität aufweist als die meisten heute gängigen Spiele.
Das Laden unterschiedlicher ANN kann den Schwierigkeitsgrad während des Spieles anpassen.	Der Spieler springt in diesem Falle immer von einem fix vorgegebenen Schwierigkeitsgrad zum anderen.

Tabelle 3.7: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [29], Abschnitt 3.6 „Dynamische Schwierigkeitsanpassung mittels Artificial Neural Networks“.

jedoch immens rechenintensiv ist, würde es bei komplexeren Spielen sehr schnell zu Problemen mit der Berechnungszeit führen. Sie haben daher anhand eines vereinfachten Pac-Man-Spiels gezeigt, dass es möglich ist, in der Test-Phase eines Spieles mittels UCT-Verfahren Daten zu erzeugen, mit denen ein oder mehrere ANN trainiert werden können.

Die unterschiedlich gut trainierten ANN wurden mit unterschiedlich guten Daten zum Lernen erzeugt. So erhielt ein schlecht trainiertes ANN Daten von einem UCT, welches nur eine geringe Berechnungszeit zur Verfügung gestellt bekommen hatte und Daten produzierte, die nur in seltenen Fällen zu einem Sieg geführt hätten. Längere Berechnungszeiten des UCT führten zu besseren Daten und daher geringeren Gewinnchancen des Spielers, d. h. einer höheren Schwierigkeit. Wurde ein ANN mit den Daten dieses UCT trainiert, wird auch die Wahrscheinlichkeit des ANN sehr hoch sein, zu gewinnen und eine hohe Herausforderung für den Spieler darzustellen.

Xinyu et al. schlagen in [29] vor, mehrere ANN für ein Spiel zu generieren und diese online gegeneinander auszutauschen, je nachdem innerhalb welches Schwierigkeitsbereiches man sich gerade bewegt.

3.7 DDA durch prozedurale Levelgenerierung

Jennings-Teats hat das Konzept der prozeduralen Levelgenerierung angepasst: statt den bisher von ihm in [24] sogenannten

> [...] offline full-level generation tequniques, meaning that they create an entire playable level as a whole – usually ahead of time [...]

+	−
Der Wiederspielwert wird durch ständig neue Levels erhöht, Flow durch die Anpassung der hinzugefügten Bauteile.	Es ist fraglich, ob dieser Ansatz bei anderen als *Jump and Run* gut funktioniert.

Tabelle 3.8: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [24], Abschnitt 3.7 „DDA durch prozedurale Levelgenerierung“.

entwickelt er einen Algorithmus, der vorgenerierte Teile eines *Jump and Runs*-Spiels zur Laufzeit an das bisher gespielte Level anfügt.

Alle zur Verfügung stehenden Levelbauteile wurden von Spielern in unterschiedlichen Kombinationen gespielt und bei einem Fehler oder am Ende des Levels bewertet. Ein *machine learning* Algorithmus hat anhand der gespielten Teile und der Bewertungen durch die Spieler eine Klassifizierung jedes Bausteins erstellt.

Nach der Klassifizierung war es möglich, für eine bestimmte Schwierigkeit entsprechende Bauteile auszuwählen, sodass ein dafür entwickelter Algorithmus die Leistung des Spielers beobachtet und sein Level mit seiner Schwierigkeit entsprechenden Bauteilen erweitert.

3.8 Ein Versuch zum allgemeinen Ansatz

Missura und Gärtner arbeiten seit 2009 [31–33] an einer allgmeinen Lösung des Problems von dynamischer Schwierigkeitsanpassung, welcher zwei Eingabeparameter annimmt und die richtigen Adaptierungen für einen Spielertypus zurückliefern soll.

Der Spielertypus stellt hier eine Gruppe an Spielern dar, die ein ähnliches Spielverhalten aufweisen [32]. Die Einteilung des Spielers erfolgt via Analyse gesammelter Daten, jene aus den Alpha- und Beta-Tests, und neuen, wenn das Spiel schon veröffentlicht wurde, aus den Daten der Spieler. Die Daten aus den Test-Phasen werden von den Testern kategorisiert, wodurch das Einteilen der Test-Daten in bestimmte Spielertypen (z. B. *easy*, *medium* und *hard*) und das Zuweisen eines neuen Spielers zu einer Spielergruppe einfacher fällt. Dieser Schritt, d. h. die Zuweisung eines neuen Spielers zu einer Gruppe, sollte online stattfinden.

In [32] haben Missura und Gärtner folgende, wichtige Erkenntnisse über die Spielerprofilierung gewonnen:

– Die dynamische Schwierigkeitsanpassung hat, bei bei bestmöglicher Einteilung des Spielers zu einem Spielertypus, immer besser funktioniert, als statische Schwierigkeit.

- Die Ergebnisse nach der bestmöglichen Einteilung sind immer besser als die schlechtmöglichsten Einteilungen.
- Je mehr Spielertypen es gibt, desto schlechter werden die Ergebnisse für sowohl die bestmögliche als auch die schlechtestmögliche Einteilung, wobei die Adaptierung zu lediglich einem Spielertyp nicht das Ziel ist.
- Selbst die Spielerkategorisierung aus wenig Daten (zu Anfang des Spieles) findet, unabhängig von der Anzahl der Spielertypen, immer ein im folgenden definiertes, akzeptables Ergebnis [32]:

> [...] the best we can hope for is that in some settings the performance of the predicted cluster is close to the best cluster while always being much better than the worst cluster.

In [31] befassen sie sich mit der Verallgemeinerung des Spiels, sodass es möglich ist, das selbe Schwierigkeitskonzept auf alle Spiele anzuwenden. So wird ein Graph definiert, dessen Knoten Vektoren sind, die Spielsituationen definieren und nach Schwierigkeit sortiert aneinander gereiht sind.

Der Spieler bewegt sich durch diesen Graphen anhand seiner Entscheidungen[5] und wird vom Schwierigkeitsalgorithmus durch den Graphen geführt, d. h. in eine folgende, leichtere, gleichschwierige oder schwierigere Situation gebracht.

Dabei treten mehrere Probleme auf:

- Der Graph und die Ordnung des Graphen sind bisher vollständig unbekannt.
- Die Wahrnehmung der Schwierigkeit ändert sich. Missura und Gärtner nehmen an, dass sich diese Wahrnehmung nur in eine Richtung bewegt, nämlich, dass Situationen mit jedem Zeitschritt ein wenig einfacher eingeschätzt werden, als bisher.
- Das Problem einer möglichen, endlosen Schleife.

Weitergehend haben sich Missura und Gärtner mit dem Algorithmus zur Knotenbestimmung beschäftigt, dessen erste Regel sein sollte, so selten wie möglich einen Knoten mit falscher Schwierigkeit auszuwählen. Der von ihnen entwickelte Algorithmus verzeichnet eine hohe Performanz, wenn es darum ging, die Schwierigkeit für einen einzelnen Spieler zu definieren (da die Schwierigkeit laut ihren Definitionen nicht sprunghaft wechseln würde), und zeigt auch beim sehr problematischen Spielerwechsel-Szenario verhältnismäßig gute Ergebnisse.

[5] Die Spielsituationen beinhalten z. B. auch die Beschreibung der Waffe, daher würde sich die Position im Graphen wesentlich ändern, sobald der Spieler die Waffe wechselt.

+	−
Ein nach Schwierigkeit sortierter Graph mit kategorisierten Knoten würde die Bereitstellung von adäquaten Herausforderungen vereinfachen.	Größe des Graphen für komplexere Spiele fragwürdig (7x7 Spielfeld mit 50 Knoten (siehe [33]).
	Keine Regeln, nach denen die Knoten verbunden werden können/sollen, keine Hinweise über die Reihung nach Schwierigkeitsgrad.
Die Einteilung der Spieler in Kategorien kann die Analyse für langfristige Vorhersagen seiner Performance vereinfachen.	Das Einteilen in Kategorien anhand von Messdaten wird nicht ausreichend sein, um die optimale Schwierigkeit zu wählen.

Tabelle 3.9: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [31–33], Abschnitt 3.8 „Ein Versuch zum allgemeinen Ansatz“.

3.9 Hamlet

Hunicke war eine der Vorreiterinnen der dynamischen Schwierigkeitsanpassung. In [22], teilt Hunicke *Spiel* in drei essentielle Elemente – MDA – auf:

Mechanics Die einzelnen Komponenten und Regeln, auf dem Level der Datenrepräsentation und Algorithmen,

Dynamics Das Laufzeitverhalten der *Mechanics* sowie deren Interaktionen untereinander und Reaktionen auf Spielereingaben, sowie

Aesthetics Die emotionale Reaktion, die beim Spieler durch die Interaktionen hervorgerufen werden.

Sie argumentiert, dass, wenn die emotionale Reaktion des Spieler verändert werden soll, aufgrund der Abhängigkeiten der drei Elemente zwingend Änderungen in den *Mechanics* vorgenommen werden müssen.

Zeitgleich entwickelte sie *Hamlet* [20, 21], eine Bibliothek innerhalb der *Half Life* Game-Engine. Diese Bibliothek bietet u. a. die Funktionalitäten für das Beobachten der Spielstatistik, das Definieren von Anpassungen und -reaktionen, das Ausführen dieser Aktionen und Reaktionen, das Anzeigen der Daten und Einstellungen sowie das Erzeugen von Spielsitzungsaufzeichnungen.

Anhand eines FPS testet sie ihren Ansatz: Spieler über Statistiken beobachten, vorhersagen, wie die nächste Herausforderung ausgeht und unerwünschte Situationen möglichst unauffällig vermeiden. Sie definiert die Akti-

onsschleifen eines *FPS* als *searching*, *retrieving*, *solving* und *fighting* und lässt ihr DDA-System bestimmte Situationen bevorzugen und benachteiligen, je nachdem, in welcher Verfassung sich der Spieler befindet. Dabei betont sie allerdings in [21]:

> We want the system to *support* the player – without eliminating negative feedback and making everything very easy and predictable. We want *Hamlet* to intervene *just enough* – so that the system is very stable and predictable over time.

Ihre Bewertungsmethode bezieht sich auf das *Inventar*, welches alle dem Spieler zur Verfügung stehenden Ressourcen beinhaltet[6]. Sie interessiert sich vor allem für die Bewegungen innerhalb des Inventars, so z. B. für die Lebenspunkte oder Munition, die im Kampf verloren wurden, die Lebenspunkte, die durch Sprünge verloren werden, u. a.. Aus den Bewegungen und statistischen Analysen berechnet sie sogenannte *shortfalls*, d. h. sie versucht vorherzusagen, wann Elemente im Inventar zu wenig sind, um die nächste Herausforderung zu bestreiten. Den *shortfall* des Spielers, P_s, erreichnet sie aus der Summe aller Wahrscheinlichkeiten, dass der Schaden die anfänglichen Lebenspunkte überschreitet:

$$P_s = 1 - F(z)$$

wobei $F(z)$ das Fehlerintegral ist, mit den anfänglichen Lebenspunkten und dem durschnittlichen Wert sowie die Standardabweichung des Lebensverlustes über die Zeit.

Sollte ein derartiger *shortfall* bevorstehen, beurteilt Hamlet einen bevorstehenden Eingriff ins System z. B. über

- die Zeit, die der Spieler bereits spielt,
- die Zahl der Tode – sowohl für diese Position, als auch das aktuelle Level und über das gesamte Spiel,
- die Anzahl der bisherigen Versuche für diese Herausforderung,
- wie oft bereits eingegriffen wurde,
- u. a..

Sie nennt es die Kosten eines Eingriffs und entscheidet anhand dieser Kosten und einer definierten *Policy*, ob und wie eingegriffen wird. Sie gibt zwei Beispiele für *Policies* in [21]:

> **Comfort Zone:** This policy will keep players within a mean range of health points over time. Entities will be tuned to shoot less often and less accurately, and heals is readily available. The goal is to keep the player at about 50% health, occasionally dipping near 25% or cresting to 75%.
>
> [...]

[6] Das Inventar beinhaltet z. B. die Lebenspunkte oder Anzahl der Munition.

+	−
Die Entwicklung des State-Graphen, dessen Zustände vom DDA-System je nach Situation gefördert oder unterdrückt werden können, macht das System genreunabhängig.	Bisherige Experimente hatten keine guten Resultate. Trotz versprochenem hohen Potential gibt es keine weiteren Publikationen.
Es wird betont, dass Adaptierungen unauffällig vonstatten gehen müssen und logisch in die Spielwelt passen sollen.	
Das Konzept der *Policies* setzt die Spielerwünsche gut um.	Der genaue Aufbau einer *Policy* ist allerdings unbeschrieben.

Tabelle 3.10: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [20–22], Abschnitt 3.9 „Hamlet“.

> **Discomfort Zone:** This policy is designed for more experienced players. The entities in a DZ game are incresingly accurate, ammo and health relatively scarce. The goal here is to keep the player „on the edge of her seat“ – constantly on the alert for enemies, and fighting back from 15 or 20% health most of the time.

In [20] veröffentlicht Hunicke die Ergebnisse ihres Experiments mit Hamlet und der *Comfort Zone Policy*. Sie ließ zwei Gruppen von Personen jeweils 15 Minuten spielen – entweder die adaptierte oder normale Version des Spiels. Während die normale Version durchschnittliche 6.4 Tode verzeichnet, hat die Adaptierung bei der anderen Gruppe für nur 4 Tode gesorgt. Sie vermerkt, dass die spätere Angabe des Spielers darüber, ob sein Spiel adaptiert wurde, häufig nicht mit der Wirklichkeit übereinstimmt. Ausserdem kam dabei heraus, dass sich viele der Spieler falsch eingeschätzt haben und andere Leistungen erbrachten, als sie von sich erwarteten.

Trotz der nicht überragenden Ergebnisse traut Hunicke ihrem Ansatz großes Potential zu [20]:

> As such, we believe that the techniques used in Hamlet will scale beyond simple FPS games to any system with a clear economy of items – including RPG and strategy games as well as simpler arcade-style interactions. We are currently extending the DDA techniques expressed in this paper for use in Neverwinter Nights.

+	−
Es wird argumentiert, dass spielerzentrierte Anpassungen auch die persönlichen Wünsche des Spielers mit berücksichtigen müssen.	

Tabelle 3.11: Zusammenfassung der wichtigen Argumentationen und Erkenntnisse in [46], Abschnitt 3.10 „Profilbasierte Anpassung“.

3.10 Profilbasierte Anpassung

Yun et al. [46] bauen ihr DDA-System auf die Arbeit von Bartle [8] auf, der die Spielertypen und -charakteristiken definiert hatte. Ausgehend von Bartles Definitionen argumentieren Yun et al., dass diese Informationen verwendet werden können, um die Spielerfahrung zu optimieren.

Das *PADS*, engl. *profile-based adaptive difficulty system*, definiert für das vorliegende Spiel (und -genre) einige Profile, denen die Spieler zugeordnet werden können. Am Beispiel *FPS* stellt Yun dem Spieler die Frage nach ihrer Erfahrung und nach ihrem Ziel beim Spielen.

Anhand der Leistungsdaten – in Yun et al. werden im *FPS* die Anzahl der getöteten Gegner, der gesehenen Gegner, der Lebensverlust/-zuwachs und die tot verbrachte Zeit seit der letzten Änderung als Leistungsdaten verwendet – wird das System angepasst. In einem Konvertierer werden die Leistungsdaten in eine Punkteskala übersetzt, wobei der Konverter Schwellwerte für die Daten verwendet, die aufgrund des vordefinierten Spielerprofils, dem der Spieler zugeordnet ist, variieren.

Durch die Abhängigkeit der Schwierigkeitsanpassungen von dem aktuellen Spielerprofil definieren Yun et al. ihren Ansatz als *player centered* [46]. Die Anmerkung, dass ihre drei definierten Schwierigkeitsstufen zu wenig und für die Spieler in den Experimenten deutlich zu spüren war, zeigt, dass in diesem Fall eine bessere Abstufung nötig wäre, um eine durchgängigere Schwierigkeitswahrnehmung zu gewährleisten.

Kapitel 4

Eigener Ansatz

Ausgehend von diesen vielen Beispielen, wie man das Thema „dynamische Schwierigkeitsanpassung“ verstehen und wie man es behandeln kann, sollte eine Lösung entwickelt werden, welches Thema der nächsten Abschnitte sein wird.

4.1 Anforderungen

Die Hauptanforderung an den neuen Ansatz waren mit Fokus auf die schnelle und billige Verwendbarkeit und eine einfache Wiederverwendbarkeit gehalten. Ziel sollte sein, dass dieser Ansatz es möglich macht, dynamische Schwierigkeitsanpassungen auch für kleinere und vor allem mehr Spielproduzenten populärer zu machen.

Daraus entstanden folgende Anforderungen:

Allgemeingültigkeit: Die zu entwickelnde Lösung soll kaum konkrete Annahmen treffen und daher auf möglichst viele Situationen, Menschen und Spiele zutreffen.

Entwicklerfreiheit: Die zu entwickelnde Lösung sollte dem Entwickler eines Spiels, ungeachtet der Nachteile wie möglichen Missbrauch oder falscher Verwendung der Funktionalitäten, nur wenige Grenzen – eher Leitlinien – setzen, da es im Bereich der Spielentwicklung möglich sein soll, Neues zu schaffen – und auf diese neuen Situationen und Spielmechaniken zum jetzigen Zeitpunkt nicht Rücksicht genommen werden kann.

So soll die Lösung vom Spiel *verwendet werden* anstatt das Spiel *zu verwenden*.

Entwicklerfreundlichkeit: Die Einlernphase in die Lösung soll niedrig bleiben, d. h. die Komplexität des Themas sollte nicht kompliziert zu ver-

wenden sein, um keine großen finanziellen, zeitlichen oder mentalen Hürden zu erzeugen.

Genreunabhängigkeit: Um möglichst hohe Wiederverwendbarkeit zu gewährleisten, sollte die Lösung genreunabhängig sein, d. h. jegliche genrespezifischen Anpassungen sollen durchgeführt werden können, aber nicht vorgegeben und daher aufgezwungen sein.

4.2 Skizzierung des Frameworks

Aus den bisherigen Ansätzen lassen sich klar mehrere große Teilgebiete erkennen, die die Grundsätze des Systems der dynamischen Schwierigkeitsanpassung ausmachen (siehe Abb. 4.1). Jeder Teilbereich interagiert regelmäßig mit der Game-Engine, die u. a. die Logik und alle Spielmechaniken enthält. Diese Architektur wird auch in der theoretischen Arbeit [7] von C. Bailey vorgeschlagen. Obwohl Bailey diese Architektur für Experimente zum Verständnis von Spielerfahrung und neuen Algorithmen zur Veränderung der Spielschwierigkeit entworfen hat, kann sie auch als praktisches Mittel ausserhalb von Experimenten eingesetzt werden.

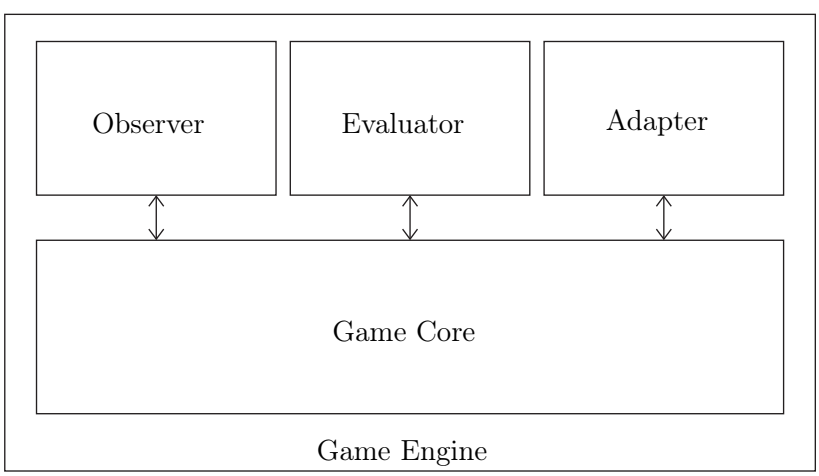

Abbildung 4.1: Das Framwork wird bei Verwendung Teil der Game-Engine werden und regelmäßig mit den Game-Core Elementen interagieren.

Die Aufgabenbereiche der einzelnen Elemente, die nicht klar abgegrenzt sind, werden in den nächsten Abschnitten näher erläutert. Eine prototypische Implementierung wird in Abschnitt 4.3 vorgestellt, bei der auf einige Besonderheiten im Bezug auf die Game-Engine *Cogaen* eingegangen wird. Ziel des Prototyps war, sowohl den Aufwand für die Implementierung für eine bestimmte Game-Engine zu testen, als auch die Konzepte zu überprüfen und

mögliche Anforderungen zu finden, die das Framework an die Game-Engine stellt.

4.2.1 Beobachtung, Observation

Die Beobachtung des Spielers stellt sich als wesentliches Element der Schwierigkeitsanpassung heraus. Ohne einer zuverlässigen Entscheidungsgrundlage ist es nicht möglich, akkurate Entscheidungen zu treffen, sodass die Auswahl der zu beobachtenden Parameter genau überlegt sein muss. Der Observer, in [7] auch „Monitoring“ genannt, dient zum Überwachen dieser Parameter und Objekte.

Auswahl der Objekte

Es wurde bereits erwähnt, dass eine Vielzahl an Auswahlmöglichkeiten für überwachbare Parameter existiert, so werden z. B. die Interaktionen mit dem Spiel aufgezeichnet [4–6, 16], die Werte bestimmter Attribute während des Spielverlaufs verfolgt [18], oder die Endergebnisse der bisher bestrittenen Herausforderungen beobachtet [4–6]. Die Möglichkeit, psychophysiologische Daten zu verwenden, wird hier kategorisch ausgeschlossen, da die Annahme getroffen wird, dass die notwendigen Gerätschaften nur unter Laborbedingungen zur Verfügung stehen.

Wichtig ist, dass die Objekte einen Bezug zum Spiel und zu der Spielerleistung haben, sodass ihr Wert selbst, ihre Entwicklung oder die Entwicklung im Zusammenhang mit anderen Objektwerten oder -entwicklungen aussagekräftig sind.

Einbinden der Parameter ins Framework

Da das Framework für mehrere Spiel-Genres und Game-Engines[1] verwendbar sein soll, ist die Spielmechanik für den Observer des Frameworks unbekannt. Die Game-Engine wie eine Blackbox, d. h. der Observer weiß nicht, wie diese arbeitet und welche Schnittstellen sie anbietet.

Daher wird, wie in Abb. 4.4 schematisch dargestellt, der Game-Engine ein Interface zur Verfügung gestellt, über das es dem Observer mitteilen kann, welche Attribute zu beobachten sind. Der Entwickler des Spiels kann aufgrund der individuellen Spielmechanik und Game-Engine entscheiden, welche Objekte er beobachten lassen möchte.

[1] Fachbegriff für die wiederverwendbare Basisarchitektur eines Spiels, die die nötigen Grundelemente wie Graphics-, Sound-Engine oder Ressourcenverwaltung zur Verfügung stellt, und zu dem die speziellen Spielmechaniken des zu entwickelnden Spiels hinzugefügt werden.

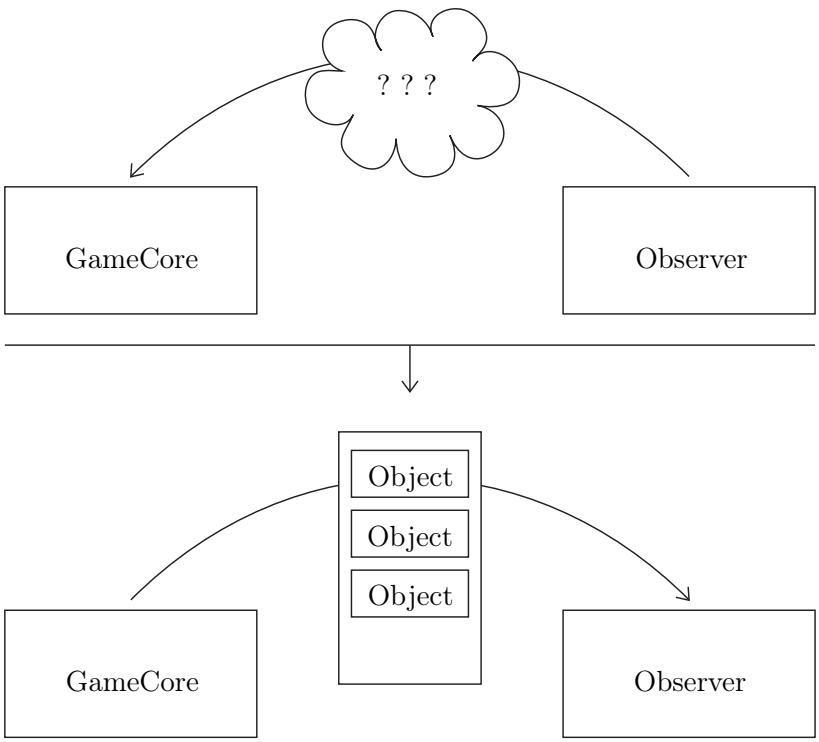

Abbildung 4.2: Da die Game-Engine für das Framework eine Blackbox ist, weiß der Observer weder, wie es die notwendigen Daten extrahieren kann noch, was die notwendigen Daten sind. Daher ist es Aufgabe der Game-Engine, dem Observer mitzuteilen, welche Daten relevant sind.

4.2.2 Analyse, Evaluation

Der Bereich der gesamtheitlichen Analyse ist der zentralste Stelle des Systems. Hier geht es darum, alle miteinander in Beziehung stehenden, beobachtbaren Objekte (`IObservableGameObject`) auch logisch miteinander in Bezug zu bringen. Die Effektivität der Analyse hängt wesentlich davon ab, wie gut die Wechselwirkungen und Beziehungen der einzelnen Objekte abgebildet werden können. Dabei sollten alle relevanten Einheiten, Herausforderungen, Rätsel, Spielmechaniken, Prioritäten, u. v. m. berücksichtigt werden.

Ziel der Analyse

Mit seinem allumfassenden Wissen soll das Modul zur Analyse folgende für das System wesentliche Aufgaben erfüllen:

- Evaluation des Spielers, bestehend aus
  - Evaluation der Situationen, und
  - Evaluation der Interaktionen;

- Definition der Schwierigkeit,
- Anweisung zur Adaptierung, wenn nötig.

Situationen evaluieren

Beim Bewerten der Situation evaluiert das System alle relevanten Informationen auf drei Ebenen:

Ebene – Attribut: Attribute sollten eine Bewertung ihres aktuellen Status zur Verfügung stellen, welcher mit einem anderen Status dieses Attributs vergleichbar sein sollte. Es ist daher eine Momentaufnahme oder Betrachtung der Entwicklung eines einzelnen Parameters ohne Zusammenhänge mit anderen Objekten.

Ebene – Objekt: Innerhalb eines Objektes können die einzelnen Bewertungen der Attribute im Kontext des Objekts miteinander verbunden werden und so die Bewertung des Objekts ermöglichen. In den meisten Fällen ist dies die Evaluation ohne Wissen zur Umgebung, das heißt eine Momentaufnahme des Objekts. Zwar sollte auch die Objektbewertung mit vorangegangenen Objektbewertungen vergleichbar sein, dennoch steht die Bewertung nicht in Relation zu z. B. anderen Objekten.

Ebene – Welt: Für die übergreifende Bewertung ist schlussendlich der Evaluator zuständig. Er kann aufgrund einer einzelnen Attributbewertung, -veränderung oder jene Bewertungen und Veränderungen eines Objekts bereits Entscheidungen treffen oder andere Faktoren miteinbeziehen, z. B. Anzahl und Entfernung von Gegnern (aktive und inaktive), die Entfernung und Anzahl umliegender Verstecke vor den Gegnern, umliegende Tränke, Munition, u. v. m..

Interaktionen evaluieren

Eine Bewertung der Spielerfertigkeiten stellt dies allerdings nicht direkt dar. Die Fähigkeit, mit den Herausforderungen umzugehen, setzt sich üblicherweise aus mehreren kleinen Herausforderungen zusammen – z. B. dem Schießen auf Gegner und dem Ausweichen einer gegnerischen Attacke. Diese einzelnen Interaktionen mit dem Spiel sollten vom System festgehalten werden. Sie sind ein guter Indikator für das Können des Spielers in den essentiellen Spielmechaniken und können in Kombination einen Richtwert für eine zusammengesetzte Herausforderung sein basierend auf den Annahmen in [4, 16]. Da in einer komplexen Herausforderung oft keine lineare Abhängigkeit zu den einzelnen benötigten Fähigkeiten erwartet werden kann, könnte man sich sowohl in der Testphase des Spiels als auch in der veröffentlichten Version auf eine vorberechnete, wahrscheinliche Standard-Leistung zu der komple-

xen Herausforderung aufbauen und die tatsächliche Leistung als Abweichung dieser Standardisierung verwenden, sofern man sich an das „quanifizierbares Ergebnis“-Prinzip aus [39] hält.

Schwierigkeitsdefinition

Die Definition der Schwierigkeit selbst erfüllt mehrere Zwecke. Zum einen ist sie wichtig, um den Flow-Status zu bestimmen. Flow ist die bereits im Abschnitt 2.2.1 beschriebene, wichtige Komponente der Spielerfahrung, die durch Spielschwierigkeit maßgeblich beeinflusst wird. Durch das Definieren des Flow-Status eines Spielers kann dieser durch Adaptieren der Schwierigkeit optimiert werden, sodass sich die für den Spieler gewünschte Erfahrung einstellt.

Der zweite Vorteil einer definierten, d. h. quantisierten Schwierigkeit, würde die Möglichkeit zum Vergleich der einzelnen Spielerleistungen sein (siehe Abb. 4.3(c)). Mit dieser Quantisierung steht es dem Spielentwickler auch weiterhin offen, Achievements und High-Scores anzubieten. Sollte die Schwierigkeit nur relativ zum Spieler gemessen werden und kann keine Übersetzung in eine vergleichbare Skala (z. B. in Form einer nach oben hin offenen Zahl, einem Prozentsatz der gemeisterten Fähigkeiten oder mittels einem gänzlich neuen, noch nicht gefundenem Ansatz) gefunden werden, wird der Vergleich unter Spielern schwer. Im Kapitel 5 wird eingehender über die Möglichkeiten diskutiert.

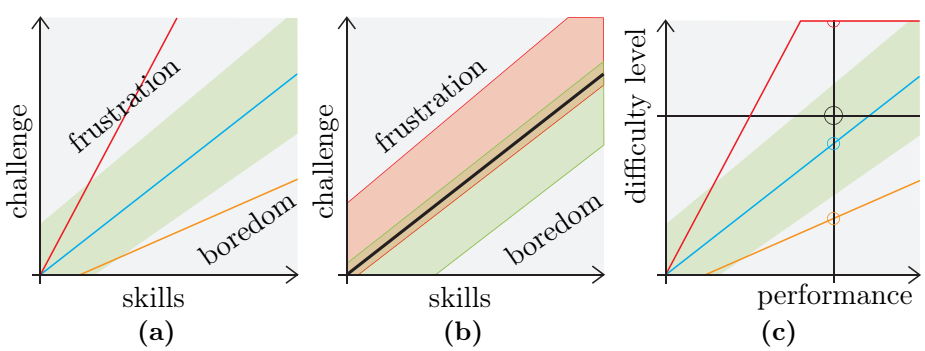

Abbildung 4.3: Die individuelle Flow-Kurve (a) und die unterschiedlichen Wünsche des Spielers ((a) grün: standard, (b) rot: eher spannend, grün: eher sehr einfach) können Basis für den Impuls zur Änderung sein, wenn der durch die aktuellen Werte *performance* und *difficulty level* definierte Status ((c) schwarzer Kreis) ausserhalb des grünen Bereichs liegen würde, d. h. zu weit von der idealen Flow-Kurve ((c) Kreise ausserhalb des grünen Bereichs) des Spielers entfernt wäre.

Die Hauptaufgabe des Analyse-Moduls stellt die umfangreiche Weltüberwachung dar, d. h., auch das Analyse-Modul beobachtet, es wertet jedoch im

gesamtheitlichen Kontext.

Das Inbezugsetzen der einzelnen Objekte kann dabei auf mehrere Arten geschehen. So ist die direkte Implementierung eines Algorithmus möglich, sofern die komplexen Beziehungen damit abgebildet werden können. Andernfalls kann auf andere Techniken, wie Clustering oder ANN zurückgegriffen werden. Die erforderlichen Daten für diese Methoden können als gegeben angenommen werden, da auch ein Spiel mit DDA einer Testphase unterzogen werden sollte.

Die Analyse endet schlussendlich in der Entscheidung, die Schwierigkeit anzupassen oder nicht, der Information, wie intensiv diese Anpassung ausfallen soll und welcher Art die Anpassung sein wird.

4.2.3 Anpassung, Adaption

Der letzte Schritt in der Kette ist die Adaption selbst. Auch sie muss mehrere Aufgaben erfüllen können:

- Sofortige Eingriffe in das Spielgeschehen,
- Langfristige Änderungen der Spielschwierigkeit,
- Erstellung neuer Herausforderungen.

Änderungsmöglichkeiten

Dabei kann es in einem Spiel aufgrund vielseitiger Spielmechanik viele Möglichkeiten geben, das Spiel zu adaptieren. In [7] werden einige Möglichkeiten der groben Adaptionsbereiche genannt:

- Spieler Character-Attribute,
- Nicht-Spieler Character-Attribute,
- Welt- und Level-Attribute,
- Rätsel- und Hindernis-Attribute.

Je besser die gesamtheitliche Analyse der Spielerleistungen, d. h. detailliertere Bewertung unterschiedlicher Fähigkeiten, desto genauer kann mittels präziser Anweisungen das Spiel vom Adapter angepasst werden. So kann, ausgehend von den vorgeschlagenen Veränderungen in [7] aufgrund der beobachteten Fähigkeit des Spielers, sich in einem FPS hinter Gegenständen vor Gegnern zu verstecken, die Anzahl der Verfügbaren Gegenstände erhöht (einfacher) oder verringert (schwerer) werden. Oder, wie in [7] vorgeschlagen, die Puzzle-Teile eines bestimmten Rätsels näher (einfacher) oder weiter weg (schwerer) vom Rätsel selbst platziert werden, ausgehend von der beobachteten Leistung bei bisherigen Rätseln dieser Art.

Im Kapitel 5 wird näher darauf eingegangen, welche Änderungen bevorzugt und welche gemieden werden sollten.

Einbindung der Änderungsmöglichkeiten

Da diese Änderungen wieder stark von der Spielmechanik abhängen, steht das Framework vor einem ähnlichen Problem wie beim Observer, daher wird auch eine ähnliche Lösung angeboten: die Game-Engine informiert das Framework über alle adaptierbaren Objekte und welche Adaptierungen sie ausführen können, sodass der Adaptionsmechanismus einem Event-System ähnelt, mit dem Unterschied, dass bei einer Adaptierung *nicht alle* dafür geeigneten Objekte darüber informiert werden (siehe Abb **??**).

Der Adaptionsmanager trifft die Entscheidung, wieviele und welche Objekte die Anweisung zur Anpassung weitergeleitet bekommen. Die Adaptierung selbst muss das dazu aufgeforderte Objekt selbst übernehmen. So bleibt die relevante Logik bei dem Objekt, zu dem sie gehört.

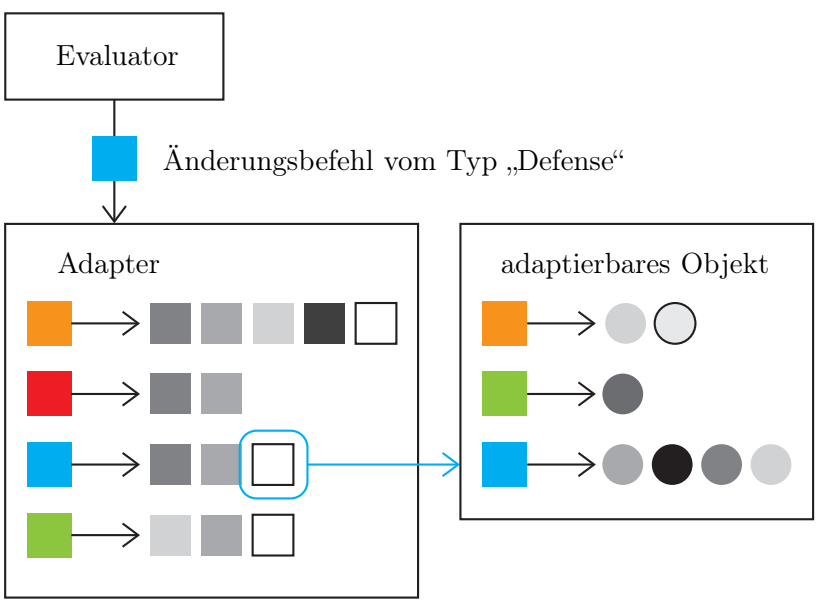

Abbildung 4.4: Der Evaluator setzt einen Befehl zur Änderung einer bestimmten Änderungsgruppe an. Im Adapter sind die adaptierbaren Objekte (graue Quadrate) vermerkt und entscheidet, an welches Objekt der Befehl weitergegeben wird. Das Objekt selbst kann auf mehrere Befehlstypen hören und führt dann eine Liste an Aktionen aus (graue Kreise).

Erschaffung neuer Herausforderungen

Neben der Möglichkeit, Attribute vorgegebener Herausforderungen zu verändern, kann es dem Adapter auch zukommen, maßgeschneiderte Herausforderungen zu erstellen.

Adaptieren, aber richtig!

Ideales DDA funktioniert unauffällig [21], d. h. es beansprucht kaum Rechenleistung und die Adaptierungen sind nur unterbewusst bemerkbar. Hunicke weist in [20] zwar auf die *change blindness* hin, was die geringe Wahrnehmung gegenüber Veränderungen im periphären Sichtfeld beschreibt, dennoch sollten Adaptierungen vermieden werden, die auffällige sind. So sollte keine unerwartete Lebensregeneration einsetzen, da sich der Spieler fragen würde, woher das kommt (ausser es gehört zu den Spielmechanismen) – und auf die Frage, wie es zu diesem Verhalten kommt, folgt oftmals das gezielte Ausnutzen des Mechanismus, das sogenannte *cheaten*. Um unauffällig zu bleiben und *cheating* zu vermeiden, lohnt es sich, unberechenbar zu sein. Auf eine bestimmte Situation sollte das System mehrere Möglichkeiten haben, zum zu reagieren.

Ein weiterer wichtiger Punkt zur Kontrolle der Anpassungen sind die Wünsche des Spielers, denn es gibt auch jene Gruppe, die erwartet, dass das System nicht eingreift um zu helfen, sondern lediglich um die Herausforderung stets auf einem anspruchsvollen Niveau zu halten. Auch dies ist ein guter Grund, die Präferenzen des Spielers in einem Profil festzuhalten.

Weitere Diskussionspunkte werden in Kapitel 5 angesprochen.

4.2.4 Weitere Elemente

Neben der Verwendung des Frameworks gibt es weitere wichtige Elemente des DDA, die bei diesem Ansatz berücksichtigt werden sollten.

Spielerprofil

Da heutzutage bereits sehr viele Plattformen für Spielerprofile vorhanden sind (z. B. *Steam*- oder *XBOX*-Profile), und bereits einige Spiele eine Verbindung zu derartigen Platformen aus unterschiedlichsten Gründen voraussetzen (z. B. *Dota2* setzt einen *Steam*-Account voraus, um das Spiel anfordern zu können), kann angenommen werden, dass die Akzeptanz eines – möglicherweise zentralen, spielübergreifenden – Profils hoch ist. Daher haben Spiele mit DDA-Systemen, die ein Spielerprofil verlangen (um die Leistungen des Spielers zu speichern), keine Ablehnung zu erwarten.

Anfangswert-Problem: Zu Beginn des Projektes wurde davon ausgegangen, dass die Evaluation des Spielers zu einem sogenannten „Anfangswert-Problem“ führt, d. h. anfangs zu wenige Daten zur Verfügung stehen, um eine korrekte Bewertung durchzuführen. So könnte der Spieler aus Glück eine sehr gute erste Leistung bringen, das DDA-System daraus schlussfolgern, dass der Spieler sehr gut ist und ihn mit Herausforderungen konfrontieren, die für den Spieler nicht bewältigbar sind. Das DDA-System würde in dieser

Situation versuchen, die Herausforderung erneut anzupassen und der Spieler könnte die resultierenden Schwankungen als unangenehm empfinden.

Um dieses Anfangswert-Problem zu umgehen, wurde angedacht, allgemeine Fähigkeiten des Spielers als Grundlage für eine akkurate Bewertung zu verwenden. Reaktionsvermögen und Genauigkeit mit den Eingabemedien (Tastatur, Maus, Controller, u. a.) werden in allen Spielgenres auf die eine oder andere Art benötigt und hätten als derartige Basis dienen können.

Bei näherer Betrachtung des Aufbaus eines Spiels und Klimmt's et al. Annahme, dass Spieler zu Beginn eines Spiels niedrigere Schwierigkeiten bevorzugen [25], kann jedoch davon ausgegangen werden, dass der Spieler beim Neustart eines Spiels langsam in die Spielmechanik eingeführt und daher vorerst mit einfacheren Herausforderungen konfrontiert wird, sodass der Spieler an eine tendenziell niedrige Schwierigkeit zu Beginn des Spiels gewöhnt ist und dies möglicherweise sogar erwartet.

Das Anfangswert-Problem bei mehreren Spielsitzungen, d. h. wenn der Spieler an einem gespeicherten Spielstand mitten im Spielverlauf startet, kann durch das Vorhandensein eines Spielerprofils mit den bisherigen Leistungen in dem Spiel gelöst werden.

Daraus lässt sich schlussfolgern, dass die Vorraussetzung eines lokal oder zentral verwalteten Spielerprofils für DDA-fähigen Spiele nicht abgelehnt wird, wenn spielbezogene Daten gespeichert werden müssen.

Spielerwünsche: Ein bisher noch wenig beachteter Aspekt des DDA-Prinzips ist die Anpassung an die Spielereigenschaften und -wünsche. Wie bereits in Abschnitt 2.1 beschrieben, gibt es unterschiedliche Eigenschafen der Spieler, die mit berücksichtigt werden sollen.

Allen voran steht der gewünschte Herausforderungsgrad des Spielers: Möchte der Spieler immer gewinnen, möchte der Spieler meistens gewinnen oder wünscht er sich Herausforderungen, an denen er lange kämpft, bevor er sie überwinden kann? Ein erfordertes Spielerprofil würde die Beantwortung dieser einmalig zu stellenden Frage speichern und wiederverwendbar machen können. Ein während dem Spielverlauf regelmäßig gestellter Fragebogen zum aktuellen Spielspaß des Benutzers ist lediglich in der Testphase zulässig, da die Immersion und der Flow dadurch enorm gestört werden.

Auch andere Eigenschaften des Spielers, wie beispielsweise sein Spieltyp, könnten im Spielerprofil gespeichert werden und eine weitere Dimension der Individualisierung eröffnen.

Levelgestaltung

Die herkömlichen Techniken der Level-Gestaltung könnten sich durch den Einsatz eines DDA-System grundlegend verändern, denn der Level-Designer ist nun nicht mehr mit dem klassischen Design eines Levels, der Erstellung der Herausforderungen, sondern eher mit der Gestaltung eines Spannungs-

bogens betraut.

Der Spannungsbogen setzt sich aus den spannenden, herausfordernden und den entspannenden, einfachen Sequenzen zusammen. Der Level-Designer baut diesen Bogen in ein Level ein, d. h. er definiert die herausfordernden und die einfachen Gebiete in einem Level, muss sich aber nicht um das konkrete Balancing der Herausforderungen gegenüber dem Spieler kümmern.

Eine mögliche Technik wird in [19] vorgestellt, in der das Level-Design, d. h. alle im Spiel befindlichen Herausforderungen, letztendlich in eine Matrix umgewandelt wird, in der jene Positionen im Level definiert sind, an denen zu einer bestimmten Wahrscheinlichkeit eine Herausforderung bestritten werden soll. Diese Matrix wird in dieser Technik zwar aus einem fertigen Beispiel-Level generiert, der Ansatz, eine Matrix mit Wahrscheinlichkeiten zu verwenden, kann aber auch durch sogenannte *heat maps* bzw. graustufige Texturen umgesetzt werden.

Unter der Annahme, ein *FPS* sollte mit einem DDA-System entwickelt werden, bleiben dem Level-Designer z. B. folgende Möglichkeiten:

- Der Level-Designer stellt dem Spiel ein vorgegebenes Level mit statischer Struktur und vordefinierten Herausforderungen zur Verfügung, die in ihren Attributen angepasst werden dürfen. Parkour sowie Position der Gegner ändern sich nicht. Es könnten beispielsweise ihre Lebenspunkte oder ihr Schaden verändert werden, möglicherweise auch ihre Anzahl.
- Der Level-Designer baut einen fixen Parkour und zeichnet die Wahrscheinlichkeit für Gegner in eine Textur, woraus das DDA-System Herausforderungen generiert. Der Parkour ändert sich nie, jedoch könnten unterschiedliche Herausforderungen an verschiedenen Stellen erschaffen werden.
- Der Level-Designer lässt sowohl die Hindernisse als auch die Herausforderungen dynamisch erstellen und baut dafür jeweils eine Textur mit Wahrscheinlichkeiten. Die Art des Levels ändert sich nie und auch eine gewisse Ähnlichkeit wird unter den einzelnen Levels erkannt werden können, die Vielfältigkeit der Level wird es dem Spieler erschweren, sich an ein Level zu gewöhnen.

Die Möglichkeiten der Kombination von dynamischen und vorgegebenen, unveränderlichen Eigenschaften eines Levels sind unbegrenzt und werden den Charakter eines Spiels maßgeblich beeinflussen. Mit Graustufen- und *heat map*-Texturen bietet eine für den Level-Designer einfach Technik, um dem DDA-System die Grenzen und Beschaffenheiten der Levels mitzuteilen.

4.2.5 Fazit

Der ursprünglich für Debug- und Forschungszwecke definierte Systemaufbau von Bailey [7] eignet sich auch als Grundlage für einen allgemein gül-

tigen Ansatz für DDA-Systeme. Die Vielseitigkeit und Abkapselung dieser Architektur lassen den Spielentwickler viel Freiheit in der Gestaltung einzelnen Komponenten. Dabei muss aber auch deutlich angemerkt werden, dass durch die enorme Freiheit des Entwicklers viele essentielle Fehler beim Erstellen von guter DDA nicht verhindert werden können. Selbst eine perfekte technische Umsetzung von DDA kann mit den falschen Adaptierungen vom Spieler vollständig abgelehnt werden!

Ein wichtiger Bereich, der noch sehr viel Erfahrungswerte benötigt ist die Evaluierung. Verschiedene Evaluationsmechanismen müssen in Zukunft weiter auf Effizient und Präzision geprüft werden, sodass es eventuell auch hier möglich ist, ein bessere Hilfestellung durch das Framework zur Verfügung zu stellen. Aktuell müssen die Entwickler ihre eigenen Methoden erstellen und auf Tauglichkeit prüfen.

Das Hinzufügen von Spielerprofilen würde den bisher wenig beachteten Einfluss der Spielerwünsche integrieren, wodurch besser gewährleistet werden kann, dass tatsächlich die Präferenzen der Spieler, ihr Spieltyp und ihre bisherigen Erfahrungen berücksichtigt werden. Besonders sinnvoll wären spielübergreifende Profile, in denen z. B. auch genreübergreifende Fähigkeiten gespeichert und in verschiedenen Spielen verwendet werden können.

Ob Level-Designer tatsächlich zukünftig nur noch mit *heat maps* arbeiten werden, ist fraglich, ihre bisherige Arbeitsweise wird sich allerdings stark verändern. Flexiblere Gestaltung wird das Spieldesign von morgen sein, wie genau das aber aussehen wird bedarf mehr Forschung.

4.3 Prototyp

Die vorgeschlagenen Konzepte sollten in einer prototypischen Implementierung innerhalb einer bestehenden Game-Engine auf ihre Praxistauglichkeit getestet und eine Evaluation der formulierten Anforderungen und Annahmen abgegeben werden.

Game-Engine

Cogaen ist eine in Java geschriebene Game-Engine, die genreunabhängig entwickelt worden ist. Für den Prototypen wurde die Version 3[2], sowie die aktuelle Version der *Cogaen LWJGL* verwendet.

Aufgrund der komponentenbasierten Architektur eignet sich dieses Framework besonders zum Testen auf einfache Einbindung und geringen zusätzlichen Aufwand beim Anpassen der Komponenten zur Verwendung mit dem Framework.

4.3.1 Observer

In *Cogaen* können Entitäten komponentenbasiert sein, d. h. ihre Eigenschaften, Attribute und Verhaltensweisen sind in eigene Klassen, die sogenannten Komponenten, gekapselt. Sie setzen sich dann je nach Bedarf aus den entsprechenden Komponenten zusammen.

Als Beispiel soll die komponentenbasierte Entität Spieler dienen. Eine derartige Komponente könnte `Health` sein, die in sehr vielen Spielgenres von großer Bedeutung ist, weshalb es hier als Beispiel für ein zu beobachtendes Attribut gewählt wurde.

Objekte

`IObservableGameObject` ist ein Interface für jene in *Cogaen* u. a. aus Komponenten zusammengesetzte Entitäten im Spiel, die für die Schwierigkeit von Bedeutung sind, und als beobachtbares Objekt ausgezeichnet und dem `Observer` mitgeteilt werden sollten.

```
1 public interface IObservableGameObject {
2   String getName();
3   void SetupObservables();
4   <T extends IObservableAttribute<?>> T GetObservable(Class<T> c);
5   ArrayList<IObservableAttribute<?>> GetObservables();
6 }
```

Der `Observer` benötigt eine Schnittstelle zu allen zu beobachtenden Attributen des Objekts. Er kann sie dann auf eine beliebige Art intern speichern, exportieren oder anders verarbeiten. Durch den Überblick, den der Observer hat, eignet er sich auch hervorragend als universaler Monitor zum

[2]Git: https://github.com/divotkey/cogaen3-java

Debuggen – das zur Verfügung stellen lesbarer Namen des Objekts und dessen zugehörige Attribute ist für diesen Zweck sinnvoll.

```
1 public class Player extends ComponentEntity implements
      IObservableGameObject {
2   private Map<Class<?>, IObservableAttribute<?>> observables;
3   public Player(Core core, CogaenId id, double x, double y) {
4     super(core, id, TYPE_ID);
5     ...
6     addComponent(new HealthComponent(100));
7     ...
8     SetupObservables();
9   }
10  @Override
11  public void SetupObservables() {
12    observables = new HashMap<Class<?>, IObservableAttribute<?>>();
13    observables.put(HealthComponent.class, (IObservableAttribute<?>)
      getAttribute(HealthComponent.ATTR_ID));
14    ...
15  }
16  @Override
17  public String getName() {
18    return "Player";
19  }
20  @Override
21  public <T extends IObservableAttributeReadonly<?>> T GetObservable(
      Class<T> c) {
22    if (observables.containsKey(c)) {
23      return (T) observables.get(c);
24    }
25    return null;
26  }
27  @Override
28  public ArrayList<IObservableAttribute<?>> GetObservables() {
29    return (new ArrayList(observables.values()));
30  }
31  ...
32 }
```

Programm 4.1: Der `Player` als beobachtbares Objekt. Die komponentenbasierte Entität bietet sich hervorragend für die Verwendung mit dem vorgeschlagenen Framework an.

Beispiel: Die folgende Liste an Beispielen für beobachtbare Objekte eines Spiels kann abhängig vom Spielgenre variieren und erweitert werden:

- gegnerische, freundliche oder neutral gesinnte Non-Player-Character,
- Gegenstände, die verwendet werden könnnen (z. B. Rätselbausteine, Munition, Power-Ups, Quest-Gegenstände), ...

Attribute

In Java wurde ein entsprechendes Interface für die beobachtbaren Attribute wie folgt definiert:

```
 1 public interface IObservableAttributeReadonly<T> {
 2   public String getName();
 3   public T getValue();
 4   public double getNumericValue();
 5   public double getImportance();
 6   public double getPercentage();
 7   public double getEvaluation();
 8 }
 9 public interface IObservableAttribute<T> extends
       IObservableAttributeReadonly<T> {
10   public IObservableAttributeReadonly<T>
11          getRepresentation(T value, T initValue, double importance);
12   public void setValue(T value);
13   public void setMin(T value);
14   public void setMax(T value);
15   public void setInit(T value);
16   public void setImportance(double importance);
17   public void reset();
18 }
```

`IObservableAttributeReadonly` stellt das Interface zur reinen Datensammlung zur Verfügung, daher sollte das Interface überall dort, wo die Attribute nicht zwingend verändert werden müssen, auf diese Schnittstellen reduziert werden. Die Sub-Klasse `IObservableAttribute` stellt auch schreibende Methoden zur Verfügung, sodass dieses Interface für alle aktuellen, sich laufend ändernden Attribute eines Objekts verwendet wird.

Bis auf die in späteren Abschnitten näher beleuchteten Variablen `double importance` und die Methoden `double getPercentage()` könnte die in Programm 4.2 eine, auch ohne der Intention DDA zu verwenden, gebräuchliche Implementierung der Komponente `Health` darstellen, da lediglich die `Getter` und `Setter`, sowie die Identifizierung über einen Namen standardisiert und die Sichtbarkeit durch die Hierachie der Interfaces einschränkbar gemacht wurden.

Beispiel: Als Beispiele für jene Attribute, die interessant für DDA sein könnten, seien folgende genannt:

- Lebenspunkte (und -regeneration) `<Integer>`,
- Mana (und -regeneration) `<Integer>`,
- Waffe (evtl. mit näheren Waffeninformationen (Schaden pro Sekunde) und Waffenressourcen (Munition, Zauberkosten)) `<Object>`,
- Zustand (z. B. im Kampf, Erholung, u. a.) `<String>` oder `<Enum>`,
- Liste der Gegner, mit denen man sich im Kampf befindet `<List>`,
- Inventar-Gegenstände (z. B. Heiltränke, Munitionsreserven, u. v. m.)

```
1 public class HealthComponent implements IObservableAttribute<Integer> {
2   Integer value, initValue,
3           min = Integer.MIN_VALUE, max = Integer.MAX_VALUE;
4   double importance;
5   public Health(int initValue, double importance) {
6     setInit(initValue);
7     setImportance(importance);
8     reset();
9   }
10  public String getName() { return "HealthAsExample"; }
11  public Integer getValue() { return this.value; }
12  public double getNumericValue() { return (double) this.getValue(); }
13  public double getImportance() { return this.importance; }
14  public double getPercentage() { return this.value/this.initValue; }
15  public double getEvaluation() {
16    return getPercentage() * getImportance();
17  }
18  public void setValue(Integer value) {
19    this.value = Math.min(max, Math.max(min, value));
20  }
21  public void setMin(Integer value) {
22    if(value > this.max) {
23      int temp = this.max; this.max = value; this.min = temp;
24      return;
25    }
26    this.min = value;
27  }
28  public void setMax(Integer value) {
29    if(value < this.min) {
30      int temp = this.min; this.min = value; this.max = temp;
31      return;
32    }
33    this.max = value;
34  }
35  public void setInit(Integer value) { this.initValue = value; }
36  public void setImportance(double importance) {
37    this.importance = Math.min(1, Math.max(0, importance))
38  }
39  public void reset() { this.value = this.initValue; }
40 }
```

Programm 4.2: `Health` als beobachtbares Attribut eines Objektes könnte das Interface `IObservableAttribute` ohne größere Umstände implementieren.

Obgleich die Evaluation schon vom Attribut berechnet werden kann, muss beachtet werden, dass ein Attribut im Idealfall keine weiteren Referenzen auf andere Attribute hat, mit denen es im Kontext steht. Die Aufgabe, die Evaluation im Kontext mit anderen, kontextnahen Parametern durchzuführen, übernimmt das `IObservableGameObject`. Eine gute Evalu-

ierungsmethode erfüllt dabei mehrere Vorraussetzungen:

- sie sollte den Zustand des Objekts in einer Variable zusammenfassen,
- die berechneten Werte sollten untereinander zumindest einen groben Vergleich zweier Situationen des Objekts ermöglichen.

Einen genauen Vergleich zweier Zustände eines Objekt, der mehr Informationen als einee einzelnen Wert zur Verfügung hat, sollte ebenfalls zur Verfügung gestellt sein.

4.3.2 Evaluation

Für die Evaluation im Prototypen wurde eine einfache numerische Representation gewählt. Das Interface:

```
1 public interface IEvaluateable
2   public double getEvaluation();
3   public double[] compareEvaluations(double[] evaluations);
4 }
```

ermöglicht es sicherzugehen, dass immer einen numerischen Wert zurückgeliefert wird. Auch andere Repräsentationsarten der Evaluation sind möglich und bei komplexeren Objekten sogar sinnvoll. Ausserdem sollen die Evaluationswerte eines Attributes oder eines Objekts untereinander vergleichbar sein.

Ebene: Attribut

Auf der Ebene der Attribute ist es üblicherweise leicht möglich, einen numerischen Wert als Bewertung des aktuellen Status zu berechnen. Am Beispiel `Health` wurde im Programm 4.2 bereits die Methode `double getPercentage()` vorgestellt. Sie eignet sich gut, um den aktuellen Status des Attributs zu representieren und ist ausserdem leicht vergleichbar.

$$status_{attribut} = \frac{current}{max}$$

Ebene: Objekt

Die Evaluation eines Objektes beruft sich im Gegensatz zu einem Attribut auf mehrere Werte, die bevorzugt aus den beobachtbaren Attributen des Objekts zusammengesetzt sind.

Auch in diesem Fall wurde eine numerische Darstellung gewählt. Beim Hinzufügen der Komponenten zu einem Objekt wurde ein Parameter übergeben, der die Gewichtung (ω) des Attributs (i) für das Objekt darstellen sollte. Der Status des Objekts errechnet sich dann wie folgt:

$$status_{object} = \sum_{i=1}^{n} status_i * \omega_i$$

wobei alle Gewichtungen (`getImportance()`) summiert 1 ergeben sollten.

Ebene: Welt

Die Evaluierung im Evaluator basiert auf ganzheitlichem Wissen und Zugriff auf alle verfügbaren Variablen. Im Prototypen wurde, aufgrund der gering gehaltenen Komplexität, ein Entscheidungs-Algorithmus auf Basis der Spieler-Werte entwickelt. Für komplexere Spiele sollte auf andere Techniken zurückgegriffen werden.

Anhand der aktuellen Spieler-Werte werden unterschiedliche Aktionen vom `Evaluator` provoziert. So würde der Algorithmus bei 30% Leben und dem Spielerstatus „im Kampf“ dem Spieler eine Soforthilfe anfordern, in diesem Fall einige Lebenspunkte – aber nicht öfter als einmal alle 100 Sekunden.

Egal, wie die Entscheidungsfindung vonstatten geht, Ziel ist es, wie bereits im Abschnitt 4.2.2

4.3.3 Adapter

Der Adapter erfüllt in der Game-Engine den Zweck der sofortigen Hilfeleistung (eben jene wenige Lebenspunkte) oder erstellt neue Herausforderungen ausgehend von den Werten des Spielers, sodass eine zugeschnittene Schwierigkeit ermöglicht werden sollte. Wie im Evaluator ist auch hier ein Algorithmus verwendet worden, der in der Entscheidung über die Parameter für eine Gegnergenerierung geendet ist. Für komplexere Spiele sind möglicherweise andere Techniken erforderlich, um eine angepasste Spielsituation zu ermöglichen.

4.3.4 Fazit

Die Einbindung des Frameworks stellte kein gröberes Problem dar, gerade beim komponentenbasierten Ansatz waren die zu beobachtenden Attribute bereits als eigene Komponente gekapselt und stellten daher ideale Bedingungen zur Verfügung. In Game-Engines, die anders arbeiten, würde es notwendig sein, einzelne Attribute, wie die Lebenspunkte, als eigene Klasse zu kapseln, um die benötigten Funktionalitäten zur Verfügung zu stellen. Auch für die Adaption erwies sich diese Struktur als vorteilhaft. Andere Game-Engines sollten noch auf Kompatibilität getestet werden um notwendige Verbesserungen am Framework vornehmen zu können.

Beim Evaluator wurden, wie erwartet, keine klaren Strukturen und Muster für andere Spiele entdeckt, sodass dem Framework keine weiteren Schnittstellen zur Verfügung gestellt werden konnten. Da dies aber der umfangreichste Part des Systems darstellt, sind weitere Tests mit unterschiedlichen Spielen erforderlich. Unterschiedliche Methoden (z. B. ANN oder Clustering) sollten auf ihre Tauglichkeit und einfache Verwendung getestet werden. Eventuell ergeben sich aus folgenden Tests dieser Richtung Richtlinien, die das Framework erweitern können und Spielentwicklern mehr Hilfestellung geben. Alles in Allem ist der Ansatz praxistauglich und sollte weiterverfolgt werden.

Kapitel 5

Offene Fragen zur Verwendung von dynamischer Schwierigkeitsanpassung

Wie bereits im Abschnitt 4.2.5 angedeutet, gibt es viele Fehler abseits der technischen Komponenten, die bei dynamischer Schwierigkeitsanpassung begangen werden können. Folgendes Kapitel wird einige essentielle Fragen aufgreifen, um zu veranschaulichen, welche Aspekte berücksichtigt werden müssen, welche Vielfalt geboten ist und um das Denken der Entwickler auf die Probleme zu sensibilisieren.

Welche Spiele sind DDA tauglich?

Grundsätzlich kann davon ausgegangen werden, dass jede Art von Spiel aufgrund der allgemeinen Definition *Spiel* („quantifyable outcome“) für dynamische Schwierigkeitsanpassung geeignet ist. Die unterschiedlichen Spielmechaniken entscheiden darüber, wie der Spieler evaluiert und das Spiel angepasst wird.

Die Form des DDA-Frameworks sollte für viele Game-Engines tauglich sein, dies ist aber bisher nicht getestet worden.

Wie wird ein Spiel DDA tauglich entworfen?

Das wichtigste bei der Entwicklung eines DDA tauglichen Spiels ist die Adaptierbarkeit – so sollten alle wichtigen Eigenschaften im Spiel adaptierbar und nicht statisch in die Game-Engine eingebaut sein.

Wie oft soll evaluiert werden?

Für die Intervalle der Evaluationen gibt es keinen Richtwert, aber allgemein kann davon ausgegangen werden, dass nicht jeden Durchlauf der Game-

Engine auch die gesamte Spielsitation bewertet werden muss – bei üblichen 60 *frames per second* wäre eine Evaluierung alle $\frac{1}{60}$ Sekunden zuviel unnötiger Aufwand. Yannakis Aussage in [44]:

> The goal is to determine the shortest time window for which the model can still predict reported entertainment with acceptable accuracy.

unterstützt diese Annahme. Je größer die Zeitfenster und höher die zu analysierende Datenmenge, desto genauer könenn die Leistungen und Fähigkeiten des Spielers bewertet werden. Kürzere Intervalle lassen eine schnellere Adaption zu.

Was kann zur Evaluierung des Spielers verwendet werden?

Zur Evaluierung der Spielerfähigkeit eignet sich die Entwicklung von Werten des Spielers im Verhältnis zur gestellten Herausforderung, und alle Interaktionen mit dem Spiel besonders [44]:

> [...] the player-opponent interaction [...] is the property that contributes the majority of the quality features of entertainment in a computer game

Dabei können bei den Interaktionen die Frequenz (*apm* – actions per minute), die Qualität der Ausführung und die Effektivität der Aktion gute Indikatoren für die Leistung des Spielers sein.

Wie oft soll adaptiert werden?

Das Adaptionsintervall ist von großer Bedeutung. Der Spieler soll die Anpassungen nicht spüren, das heißt zum Einen, dass die Veränderungen kein merkbares Ausmaß annehmen und zum Anderen, dass sich Objekte und Gegebenheiten des Spiels nicht fortlaufend ändern. Die Vorhersehbarkeit des Spiels und dadurch das Gefühl der Kontrolle des Spielers würden dadurch untergraben und können die Immersion beeinträchtigen.

Wann soll adaptiert werden?

Nicht jedes Mal, wenn der Spieler in Bedrängnis kommt, soll adaptiert werden! Es ist Teil der meisten Spielmechaniken, dass ein Spieler auch scheitern kann, und dieses Schlüsselelement darf durch die Eingriffe nicht verhindert werden. Die Adaptierungen sollen dem Spieler Herausforderungen bieten, die ihn fordern und durchaus ein gewisses Maß an *regret* erlauben [4]:

> Indeed, if the player has to deal with both success and failure and is not able to predict whether he will succeed or not, then he's on a partial reinforcement schedule, which induces the strongest motivation.

Es ist mit ein Grund, warum eine ständige Evaluierung nicht notwendig ist und erst nach mehreren Versuchen an einer Herausforderung eben jene vereinfacht werden sollte. Oder die Unterstützung, die das Spiel dem Spieler sofort bieten kann, wie einige Lebenspunkte, nicht einfach hinzugefügt werden, sondern neben dem Spieler ein „Erste-Hilfe-Set“ platziert wird, das er sich holen könnte.

Was kann adaptiert werden?

Grundsätzlich kann alles sinnvoll adaptiert werden, u. a. kann in manchen Fällen sogar sinnvoll sein, Eigenschaften der Spielmechanik (z. B. die Sprungweite des Avatars in einem *Jump and Run*) zu verändern. Meistens jedoch sind Objekte, z. B. Gegner oder das Level, am Besten zum Adaptieren geeignet.

Hier allerdings fangen Diskussionen über die Sinnhaftigkeit diverser veränderbarer Attribute an – vor allem die Frage „Sollen sich die Werte des Spielers verändern, d. h. die Rüstung, der Schaden, u. a.?“ und die Meinungen gehen auseinander: während die meisten wissenschaftlichen Arbeiten zu dem Thema die Spielereigenschaften verändern (Sprungweite des *Jump and Run*-Avatars, Laufgeschwindigkeit des Pac-Man, Schaden in *FPS*), erarbeiten sich Spieler gerne ihre Ausrüstung und sind stolz auf die erlangten Attribute. Sollten sich diese nun aufgrund ihrer Fähigkeiten verändern, z. B. die Werte einer besonders gute Rüstung würden für einen guten Spieler auf die Rüstungswerte einer kleinen Rüstung für schwache Spieler gesenkt werden, würde dies der Immersion beträchtlichen Schaden zufügen.

Eben deswegen wird der Ansatz empfohlen, nur Veränderungen der Umgebung und Nicht-Spieler-Objekten vorzunehmen. Aber auch hier können schwere Fehler gemacht werden. Ein generiertes Level mit einer bestimmten Anzahl an Objekten kann während der Laufzeit nicht einfach verändert werden (zumindest nicht, wenn keine gute Erklärung zur Veränderung der Umgebung vorhanden ist), da es unnatürlich wirkt, wenn plötzlich z. B. eine Kiste neben dem Spieler auftauchen würde. Und der Typ eines Gegners soll es dem Spieler ermöglichen, ihn einzuschätzen, d. h. ein bestimmter Gegner soll durch z. B. sein Aussehen als Typ `X` erkannt werden und bedeutet beispielsweise geringen Schaden und hohe Wendigkeit. Sollte dieser Gegnertyp `X` auf einmal hohen Schaden verursachen, wird sich der Spieler über diesen Sachverhalt wundern.

Für wen welche Einstellungen?

Die Essenz des DDA, die Individualisierung, stellt den Spieler in den Mittelpunkt. Es gibt vielerlei Möglichkeiten, die Individualisierung noch weiter zu treiben, als lediglich die Spielschwierigkeit anzupassen.

Hauptsächlich sind die Spielerintentionen zu berücksichtigen, d. h. sein

Wunsch nach Herausforderung oder Beschäftigung und seine bisherigen Erfahrungen. In großen, offenen Spielwelten wie *Minecraft* könnte einem Spieler abhängig von seinem Spielertyp anderen Aufgaben und Abenteuer zur Verfügung gestellt werden. Auch *Adventures* wie *Indiana Jones and the Fate of Atlantis* ermöglichen dem Spieler einen eigenen Pfad durch das Spiel mit Schwerpunkt auf Rätseln oder Kämpfen, je nachdem, wofür sich der Spieler entscheidet. Diese vormals entscheidungsbasierte Entwicklung des Spiels könnte in Zukunft auf der Evaluierung des Spielers basieren.

Welche Schwierigkeit soll grundsätzlich eingestellt werden?

Welche Schwierigkeit in welcher Situation sinnvoll ist, hängt zuerst von den Spielerwünschen ab, allerdings auch vom Zeitpunkt im Spiel. Wie bereits erwähnt, bevorzugen Spieler anfangs lieber leichtere Schwierigkeiten und im Laufe des Spiels immer schwierigere. Grundsätzlich sollte bei richtiger Schwierigkeitsanpassung die Schwierigkeit lediglich in kleinen Schritten nach oben korrigiert werden, und nur selten der Fall eintreten, in dem das Spiel den Spieler unterstützen muss. Dies wäre der Fall, wenn die Adaption die Schwierigkeit zu rasant hätte steigen lassen. Bereits Chanel hat in [10] die Überlegung angeregt, dass es ausreichend sein könnte, lediglich *Langeweile* zu erkennen und den Spieler dann mehr zu fordern.

Wie kann zwischen Spielern verglichen werden, wenn die Schwierigkeit vieler Herausforderungen individuell angepasst ist?

Die Frage könnte auch „Wird es noch Achievements und High-Scores geben?“ lauten und mit „Ja.“ beantwortet werden, sofern die Spielentwickler einen Weg gefunden haben, die Schwierigkeit in ihrem Spiel zu quantisieren, d. h. in irgend einer Form zu *bezeichnen* – sei dies nun mit Nummern oder benannten Bereichen (wie ehemals *easy*, *medium* und *hard*).

High-Scores und Achievements können auch in der Schwierigkeit festgelegte Levels haben, sodass z. B. ein Spieler, der in der weltweite Rangliste eingetragen sein möchte, ein fixiertes und damit für alle gleiches Level spielen muss und seine Leistung innerhalb dieses Levels in der Rangliste veröffentlicht wird. Spieler hingegen, die ihre Immersion in einer Geschichte maximieren wollen, könnten den Ansatz der dynamischen Schwierigkeitsgenerierung genießen.

Wie wird der Spieler fair belohnt?

In Einzel-Modus spielen muss man sich um eine „faire“ Belohnung weniger Gedanken machen als in Multiplayer-Spielen oder gar *MMO*s. Die Notwendigkeit einer Skalierung der Belohnung ist sehr hoch und kann anhand des Schwierigkeitslevels durchgeführt werden.

Würden zwei unterschiedliche Spieler für die selbe Aufgabe (in unterschiedlicher Schwierigkeit) die selbe Belohnung bekommen und anschließend gegeneinander antreten oder miteinander verglichen werden, so wäre der Spieler mit niedrigerem Schwierigkeitsgrad bevorzugt. Dieses Problem kann leider nicht spielübergreifend beurteilt und gelöst werden, da die Lösung von der Spielmechanik abhängt. Ein Vorschlag wäre jedoch, die Attribute der Belohnung (bis zu einem Maximum) zu verändern, je nachdem, auf welchem Schwierigkeitsgrad die Herausforderung bestanden worden ist. So ist der Erhalt der Belohnung der Beweis für das Bestehen der Prüfung, die Attribute der Belohnung der Nachweis über die Schwierigkeit.

Beim Gestalten der Belohnungssysteme ist ausserdem zu beachten, welche Auswirkung unterschiedlich starke Belohnungen auf die Spieler gegen Spieler Elemente des Spiels haben.

Kapitel 6

Potential, Ausblick

Dynamische Schwierigkeitsanpassung steckt zwar noch in der Entwicklungsphase, kann aber zu der künftig wichtigsten Schwierigkeitsregulierung in Spielen werden. Obwohl sich Spieler in vielen Diskussionen vorerst gegen die Bevormundung durch ein System – meist durch Unwissenheit oder Erfahrungen mit Spielen, die DDA nicht richtig einsetzen – sträuben, in [3] wörtlich:

> Some players hate it.

kann die Technik des DDA die Bedürfnisse jedes Einzelnen besser abdecken und in Zukunft breite Zustimmung finden.

Dennoch sind noch viele Bereiche genauer zu erforschen, vor allem die Evaluierungstechniken zeigen viel Potential. Mittels *ANN* und *machine learning* könnten bislang unentdeckte Zusammenhänge von Spielereigenschaften und -verhalten zu einer genaueren Einschätzung der Spielerfähigkeiten führen.

Auch im Bereich Team-Spiel wird zukünftig viel Arbeit notwendig sein, um zu erforschen, wie mehrere Spieler miteinander verglichen und zu einem ausgewogenen Team formiert werden können, um gemeinsam in Multiplayer-Spielen größere Herausforderungen bestreiten zu können.

Die Möglichkeiten sind endlos.

Quellenverzeichnis

Literatur

[4] Maria-Virginia Aponte, Guillaume Levieux und Stéphane Natkin. „Difficulty in videogames: an experimental validation of a formal definition“. In: *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*. (Lisbon, Portugal). ACE '11. New York, NY, USA: ACM, 2011, 49:1–49:8. ISBN: 978-1-4503-0827-4. DOI: 10.1145/2071423.2071484. URL: http://doi.acm.org/10.1145/2071423.2071484.

[5] Maria-Virginia Aponte, Guillaume Levieux und Stephane Natkin. „Measuring the level of difficulty in single player video games“. In: *Entertainment Computing* 2.4 (2011). <ce:title>Special Section: International Conference on Entertainment Computing and Special Section: Entertainment Interfaces</ce:title>, S. 205 –213. ISSN: 1875-9521. DOI: 10.1016/j.entcom.2011.04.001. URL: http://www.sciencedirect.com/science/article/pii/S1875952111000231.

[6] Maria-Virginia Aponte, Guillaume Levieux und Stéphane Natkin. „Scaling the Level of Difficulty in Single Player Video Games“. In: *Proceedings of the 8th International Conference on Entertainment Computing*. (Paris, France). ICEC '09. Berlin, Heidelberg: Springer-Verlag, 2009, S. 24–35. ISBN: 978-3-642-04051-1. DOI: 10.1007/978-3-642-04052-8_3. URL: http://dx.doi.org/10.1007/978-3-642-04052-8_3.

[7] Christine Bailey und Michael Katchabaw. „Games with Dynamic Difficulty Adjustment using POMDPs“. In: *Proceedings of the 2005 GameOn North America Conference*. Montreal, Canada, Aug. 2005, S. 18–22. URL: http://www.cp.eng.chula.ac.th/~vishnu/gameResearch/AI_november_2005/AN\%20EXPERIMENTAL\%20TESTBED\%20TO\%20ENABLE\%20AUTO-DYNAMIC\%20DIFFICULTY\%20IN\%20MODERN\%20VIDEO\%20GAMES.pdf.

[8] Richard Bartle. „Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs“. In: *Developing Online Games An Insiders Guide*. Hrsg. von Jessica Mulligan und BridgetteEditors Patrovsky. Bd. 1. 1. New Riders

Games; illustrated edition, 2003, S. 1–25. URL: http://www.mud.co.uk/richard/hcds.htm.

[9] D.O. Broin. „Using a Criteria-Based User Model for Facilitating Flow in Serious Games“. In: *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2011 Third International Conference on*. Mai 2011, S. 63–69. DOI: 10.1109/VS-GAMES.2011.15.

[10] G. Chanel u. a. „Emotion Assessment From Physiological Signals for Adaptation of Game Difficulty“. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 41.6 (Nov. 2011), S. 1052–1063. ISSN: 1083-4427. DOI: 10.1109/TSMCA.2011.2116000.

[11] Guillaume Chanel u. a. „Boredom, engagement and anxiety as indicators for adaptation to difficulty in games“. In: *Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era*. (Tampere, Finland). MindTrek '08. New York, NY, USA: ACM, 2008, S. 13–17. ISBN: 978-1-60558-197-2. DOI: http://doi.acm.org/10.1145/1457199.1457203. URL: http://doi.acm.org/10.1145/1457199.1457203.

[12] Jenova Chen. „Flow in games (and everything else)“. In: *Commun. ACM* 50.4 (Apr. 2007), S. 31–34. ISSN: 0001-0782. DOI: 10.1145/1232743.1232769. URL: http://doi.acm.org/10.1145/1232743.1232769.

[13] Ben Cowley u. a. „Toward an understanding of flow in video games“. In: *Comput. Entertain.* 6 (2 Juli 2008), 20:1–20:27. ISSN: 1544-3574. DOI: http://doi.acm.org/10.1145/1371216.1371223. URL: http://doi.acm.org/10.1145/1371216.1371223.

[14] M. Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper & Row, 1990. ISBN: 9780060162535. URL: http://books.google.at/books?id=V9KrQgAACAAJ.

[15] Cynthia D. Fisher. „Boredom at work: a neglected concept“. In: *Human Relations (HR) No. 46* (1993), S. 395–417.

[16] Robby Goetschalckx u. a. „Games with Dynamic Difficulty Adjustment using POMDPs“. In: *Proceedings of the ICML Workshop on Machine Learning and Games*. 2010. URL: http://www-kd.iai.uni-bonn.de/icml2010mlg/papers/GoetschalckxEtAl.pdf.

[17] Torben Grodal. „Video Games and the Pleasures of Control“. In: *Media entertainment The psychology of its appeal*. Routledge, 2000, S. 197–213.

[18] J. Hagelbäck und S.J. Johansson. „Measuring player experience on runtime dynamic difficulty scaling in an RTS game“. In: *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*. IEEE. 2009, S. 46–52.

[19] Nadia Hocine u. a. „Motivation based difficulty adaptation for therapeutic games“. In: *Serious Games and Applications for Health (SeGAH), 2011 IEEE 1st International Conference on*. Nov. 2011, S. 1 –8. DOI: 10.1109/SeGAH.2011.6165459.

[20] Robin Hunicke. „The case for dynamic difficulty adjustment in games“. In: *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. (Valencia, Spain). ACE '05. New York, NY, USA: ACM, 2005, S. 429–433. ISBN: 1-59593-110-4. DOI: http://doi.acm.org/10.1145/1178477.1178573. URL: http://doi.acm.org/10.1145/1178477.1178573.

[21] Robin Hunicke und Vernell Chapman. „AI for Dynamic Difficulty Adjustment in Games“. In: (2004). URL: http://www.cs.northwestern.edu/\~{}hunicke/pubs/Hamlet.pdf.

[22] Robin Hunicke, Marc Leblanc und Robert Zubek. „MDA: A formal approach to game design and game research“. In: *In Proceedings of the Challenges in Games AI Workshop, Nineteenth National Conference of Artificial Intelligence*. Press, 2004, S. 1–5.

[23] 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*. PCGames '10. Monterey, California: ACM, 2010, 11:1–11:4. ISBN: 978-1-4503-0023-0. DOI: 10.1145/1814256.1814267. URL: http://doi.acm.org/10.1145/1814256.1814267.

[24] 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*. PCGames '10. Monterey, California: ACM, 2010, 11:1–11:4. ISBN: 978-1-4503-0023-0. DOI: 10.1145/1814256.1814267. URL: http://doi.acm.org/10.1145/1814256.1814267.

[25] Christoph Klimmt u. a. „Player Performance, Satisfaction, and Video Game Enjoyment“. In: *ICEC*. 2009, S. 1–12.

[26] Raph Koster und Will Wright. *A Theory of Fun for Game Design*. Paraglyph Press, 2004. ISBN: 1932111972.

[27] Kai Kuikkaniemi u. a. „The influence of implicit and explicit biofeedback in first-person shooter games“. In: *Proceedings of the 28th international conference on Human factors in computing systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, S. 859–868. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753453. URL: http://doi.acm.org/10.1145/1753326.1753453.

[28] Annakaisa Kultima und Jaakko Stenros. „Designing games for everyone: the expanded game experience model“. In: *Proceedings of the International Academic Conference on the Future of Game Design and Technology*. Futureplay '10. Vancouver, British Columbia, Canada: ACM, 2010, S. 66–73. ISBN: 978-1-4503-0235-7. DOI: 10.1145/1920778.1920788. URL: http://doi.acm.org/10.1145/1920778.1920788.

[29] Xinyu Li u. a. „To create DDA by the approach of ANN from UCT-created data“. In: *Computer Application and System Modeling (ICCASM), 2010 International Conference on*. Bd. 8. Okt. 2010, S. V8–475–V8–478. DOI: 10.1109/ICCASM.2010.5620008.

[30] Thomas W. Malone. „What makes things fun to learn? heuristics for designing instructional computer games“. In: *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*. SIGSMALL '80. Palo Alto, California, United States: ACM, 1980, S. 162–169. ISBN: 0-89791-024-9. DOI: 10.1145/800088.802839. URL: http://doi.acm.org/10.1145/800088.802839.

[31] Olana Missura und Thomas Gärtner. „Classification on Graphs for Dynamic Difficulty Adjustment“. In: *Mining and Learning with Graphs*. Hrsg. von H. Blockeel, K. Borgwardt und X. Yan. 2009.

[32] Olana Missura und Thomas Gärtner. „Player Modeling for Intelligent Difficulty Adjustment“. In: *Proceedings of the 12th International Conference on Discovery Science*. DS '09. Porto, Portugal: Springer-Verlag, 2009, S. 197–211. ISBN: 978-3-642-04746-6. DOI: 10.1007/978-3-642-04747-3_17. URL: http://dx.doi.org/10.1007/978-3-642-04747-3_17.

[33] Olana Missura und Thomas Gärtner. „Predicting Dynamic Difficulty“. In: *Advances in Neural Information Processing Systems 24*. 2011, 2007–2015.

[34] Lennart Nacke und Craig A. Lindley. „Flow and immersion in first-person shooters: measuring the player's gameplay experience“. In: *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*. Future Play '08. Toronto, Ontario, Canada: ACM, 2008, S. 81–88. ISBN: 978-1-60558-218-4. DOI: 10.1145/1496984.1496998. URL: http://doi.acm.org/10.1145/1496984.1496998.

[35] Karolien Poels, Yvonne de Kort und Wijnand Ijsselsteijn. „„It is always a lot of fun!“: exploring dimensions of digital game experience using focus group methodology“. In: *Proceedings of the 2007 conference on Future Play*. Future Play '07. Toronto, Canada: ACM, 2007, S. 83–89. ISBN: 978-1-59593-943-2. DOI: 10.1145/1328202.1328218. URL: http://doi.acm.org/10.1145/1328202.1328218.

[36] F. E. Ritter und L. J. Schooler. „The learning curve“. In: *International encyclopedia of the social and behavioral sciences*. Amsterdam: Pergamon, 2002, S. 8602–8605.

[37] Andrew Rollings und Ernest Adams. *Andrew Rollings and Ernest Adams on Game Design*. New Riders Publishing, Mai 2003. ISBN: 1592730019. URL: http://www.worldcat.org/isbn/1592730019.

[38] Richard Ryan, C. Rigby und Andrew Przybylski. „The Motivational Pull of Video Games: A Self-Determination Theory Approach“. In: *Motivation and Emotion* 30 (4 2006). 10.1007/s11031-006-9051-8, S. 344–360. ISSN: 0146-7239. URL: http://dx.doi.org/10.1007/s11031-006-9051-8.

[39] Katie Salen und Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. The MIT Press, Sep. 2003. ISBN: 0262240459. URL: http://www.worldcat.org/isbn/0262240459.

[40] Jesse Schell. *The Art of Game Design - A Book of Lenses*. 1. Aufl. Elsevier/Morgan Kaufmann, Aug. 2008. ISBN: 0123694965.

[41] Peter Vorderer, Tilo Hartmann und Christoph Klimmt. „Explaining the enjoyment of playing video games: the role of competition“. In: *Proceedings of the second international conference on Entertainment computing*. ICEC '03. Pittsburgh, Pennsylvania: Carnegie Mellon University, 2003, S. 1–9. URL: http://dl.acm.org/citation.cfm?id=958720.958735.

[42] Bin Wu u. a. „Dynamic difficulty adjustment based on an improved algorithm of UCT for the Pac-Man Game“. In: *Electronics, Communications and Control (ICECC), 2011 International Conference on*. Sep. 2011, S. 4255–4259. DOI: 10.1109/ICECC.2011.6066649.

[43] G.N. Yannakakis und J. Hallam. „Real-time adaptation of augmented-reality games for optimizing player satisfaction“. In: *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*. Dez. 2008, S. 103 –110. DOI: 10.1109/CIG.2008.5035627.

[44] G.N. Yannakakis und J. Hallam. „Real-Time Game Adaptation for Optimizing Player Satisfaction“. In: *Computational Intelligence and AI in Games, IEEE Transactions on* 1.2 (Juni 2009), S. 121 –133. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2009.2024533.

[45] Anthony Youssef und Stephen Cossell. „Thoughts on adjusting perceived difficulty in games“. In: *Proceedings of the Sixth Australasian Conference on Interactive Entertainment*. (Sydney, Australia). IE '09. New York, NY, USA: ACM, 2009, 14:1–14:4. ISBN: 978-1-4503-0010-0. DOI: http://doi.acm.org/10.1145/1746050.1746064. URL: http://doi.acm.org/10.1145/1746050.1746064.

[46] Chang Yun u. a. „PADS: enhancing gaming experience using profile-based adaptive difficulty system“. In: *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*. Sandbox '10. Los Angeles, California: ACM, 2010, S. 31–36. ISBN: 978-1-4503-0097-1. DOI: 10.1145/1836135.1836140. URL: http://doi.acm.org/10.1145/1836135.1836140.

Online-Quellen

[1] Jan. 2011. URL: http://eu.blizzard.com/de-de/company/press/pressreleases.html?id=2450471.

[2] Okt. 2012. URL: http://www.gamasutra.com/view/feature/6474/personality_and_play_styles_a_.php?print=1.

[3] Okt. 2012. URL: http://www.gamasutra.com/view/feature/132061/the_designers_notebook_.php?print=1.