FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Entwurf einer Android-Applikation zum Erlernen von Konzepten der Informatik
auf Basis von Ideen der Gamification

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Informatikdidaktik

eingereicht von

Bernhard Fischer
Matrikelnummer 9271466

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Futschek

Wien, 20.08.2014 _____ _____

(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung zur Verfassung der Arbeit

Bernhard Fischer, BSc
Blumberggasse 21/13
1160 Wien

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Bernhard Fischer, BSc Wien, am 20. August 2014

Kurzfassung

Problemstellung dieser Masterarbeit ist es, für die Aufgaben des Biber-Wettbewerbs (ein Wettbewerb dessen Ziel es ist, junge Leute für Konzepte der Informatik zu begeistern) eine spielerische Lernumgebung in Form einer mobilen App für das Betriebssystem Android zu entwerfen. Die wissenschaftliche Fragestellung lautet wie folgt:

„Wie können motivierende und beim Erlangen der Lernziele unterstützende Gamedesign-Elemente in eine mobile Applikation zum Erlernen von Konzepten der Informatik integriert werden?"

Zur Beantwortung dieser Frage findet zunächst eine theoretische Auseinandersetzung mit dem Konzept *Gamification* statt und es wird ein kurzer Überblick über die Initiative *Bebras* (in Österreich: „*Der Biber der Informatik*") und das Betriebssystem *Android* gegeben. Weiters werden im Rahmen der theoretischen Auseinandersetzung Gamedesign-Elemente identifiziert, die für ein gamifiziertes Designkonzept notwendig sind.

Es wird ein solches Designkonzept verfasst, die einzelnen Gamedesign-Elemente werden detailliert erklärt und ihr Zusammenspiel wird erläutert.

Zentrales Ergebnis dieser Arbeit ist ein Prototyp, welcher mit dem Android Software Development Kit (Android SDK) in der Entwicklungsumgebung Eclipse mit integriertem Android Developer Tools-Plugin (ADT) verfasst wurde.

Abschließend findet eine Evaluierung dieses Prototyps statt, deren Ziel es ist, einerseits qualitative Mängel aufzuzeigen, als auch die Frage zu beantworten, ob eine sinnvolle Integration der Gamedesign-Elemente in den Prototypen gelungen ist.

Folgende wissenschaftliche Methoden kommen in dieser Arbeit zum Einsatz:

- Umfassende **Literaturrecherche** zum lerntheoretischen Hintergrund von Gamification und zur Identifikation motivierender Gamedesign-Elemente, die beim Erlangen von Lernzielen unterstützen können
- Erstellung eines **Designkonzepts** basierend auf dieser Recherche
- Entwicklung eines **Prototyps** in Form einer mobilen Applikation für das Betriebssystem Android
- ***Summative* Evaluation** mittels Online-Fragenkatalog zur Bewertung von Usability und Qualität der Integration der Gamedesign-Elemente in den Prototypen

Abstract

The aim of this master thesis is to develop a playful learning environment for the tasks of the Bebras contest (a contest, the goal of which is to awaken young people‘s interest for computer science concepts) in the form of a mobile app for the Android operating system. The scientific question reads as follows:

How can motivating game design elements which support the attainment of learning goals be integrated into a mobile application for the learning of computer science concepts?

To answer this question, the concept of *gamification* is discussed theoretically and outlines of the *Bebras* initiative and the *Android* operating system are given. Moreover, game design elements necessary for a *gamified* design concept are identified in the context of theoretical discussion.

Following, such a design concept is composed and the game design elements and their interaction with each other are explained in-depth.

Central outcome of this thesis is a prototype which was developed with the Android Software Development Kit (Android SDK) and the development environment Eclipse having the Android Developer Tools Plugin (ADT) installed.

To conclude, an evaluation of this prototype takes place which aims to highlight qualitative flaws and errors and to answer the question if the game design elements are integrated meaningfully into the prototype.

The following scientific methods are used in this thesis:

- broad **literature research** about the theoretical background of gamification in the context of learning and instruction and about motivating game design elements that can support the attainment of learning goals
- creation of a **design concept** based on that research
- development of a **prototype** in the form of a mobile app for the Android operating system
- ***summative* evaluation** by means of an online questionnaire to assess the usability and quality of the game design elements‘ integration into the prototype

Inhalt

Einleitung 11
- **Problemstellung** 11
- **Wissenschaftliche Fragestellung** 12
- **Methodisches Vorgehen** 12
- **State of the Art** 13

Bebras - Der Biber der Informatik 15
- **Entstehungsgeschichte und Überblick** 15
- **Bebras-Tasks: Klassifikation und Kriterien** 16
- **Bebras-Tasks: Beispiele** 18

Das mobile Betriebssystem Android 21
- **Geschichte** 21
- **Technische Grundlagen** 21

Theoretische Grundlagen 25
- **Spiel** 25
- **Gamification** 25
- **Lerntheoretischer Hintergrund** 27
  - *Extrinsische Motivation* 27
  - *Intrinsische Motivation* 27
  - *Leppers Design-Grundsätze intrinsischer Motivation in der Lehre* 28
  - *Malones Theorie intrinsisch motivierender Lehre* 29
  - *Taxonomie intrinsischer Motivation* 29
  - *Selbstbestimmungstheorie der Motivation* 30
  - *Flow* 30
  - *ARCS-Modell* 31
  - *Scaffolding* 32
- **Gamedesign-Elemente** 33
  - *Zehn Zutaten großartiger Spiele* 33
  - *Abstraktion von Konzepten und der Wirklichkeit* 33
  - *Ziele* 34
  - *Regeln* 35
  - *Konflikt, Wettbewerb, Kooperation* 36
  - *Zeit* 37
  - *Belohnungsstrukturen (Reward structures)* 37
  - *Feedback* 38
  - *Levels* 40
  - *Story* 41
  - *Ästhetik* 41
  - *Wiederholung (Replay)* 42
  - *Kategorisierung* 42

Designkonzept 45
- **Design Document** 45
  - *Kurzdarstellung des Designkonzepts* 45

Erwartetes Resultat 46
Lernziele 46
Tasks 47
Module 49
Bonus-Tasks 49
Levels 50
Avatare 51
Erfahrungspunkte (XP) 51
Hints 52
Bebras-Coins (BC) 52
Timer 53
Overview (inkl. Avatar-Sammlung) 53
Leaderboard (Highscores) 53
Concept Map 54
User Interface Design 55
Mock-Ups 55
Beschreibung der Mock-Ups 58
Prototyp 63
Technische Hintergrundinfos 63
Entwicklungsumgebung und Zielplattformen 63
Implementierung der Layouts 64
Vererbung 65
Adapter 65
Bilder (Drawables) 66
Speicherung des Spielzustands mit SharedPreferences 67
Erweiterbarkeit 68
Dokumentation 71
Start 71
Hauptmenü 71
Overview-Ansicht 73
Module (geöffnet) 75
Bonus-Tasks (geöffnet) 76
Tasks 76
Evaluierung 79
Methodik 79
Ergebnisse 80
Teilnehmende 80
Usability des Prototyps 82
Gamedesign-Elemente 85
Kommentare und Anregungen 90
Conclusio 95
Zusammenfassung 97
Quellen 101
Abbildungen 103
Tabellen 105

Einleitung

Problemstellung

Informatik stellt eine Schlüsseltechnologie dar, die für die Wissensarbeitenden von morgen rasant an Bedeutung gewinnt. Um Kinder und Jugendliche an Denkweisen der Informatik heranzuführen, wurde auf Initiative von Prof. Valentina Dagienė 2004 in Litauen die internationale Initiative „Bebras“ (http://www.bebras.org) gegründet, welche seitdem einmal jährlich den „Bebras Contest“ [Futschek 2009] [Dagienė 2008] organisiert. In Österreich wird diese Initiative unter dem Namen „Der Biber der Informatik“ (http://www.ocg.at/de/biber) von der „Österreichischen Computer Gesellschaft“ (OCG) mitgetragen. Hierbei handelt es sich um einen Wettbewerb, welcher 10- bis 20-jährigen Schülerinnen und Schülern Konzepte der Informatik näher bringt (Algorithmisches Denken, Graphentheorie, Codierung, Kompression etc.) und an dem im Jahr 2013 international bereits mehr als 750.000 Jugendliche teilnahmen.

Das Erlernen informatischer und analytischer Denkweisen kann für so manchen eine trockene, komplizierte oder gar mühsame Angelegenheit sein. Auf der Suche nach der Beantwortung der Frage, wie man motivationshemmenden Faktoren entgegenwirken kann, stieß ich auf den Begriff der *Gamification* [Deterding 2011|1] [Raymer 2011] [Kapp 2012], der derzeit viel Aufmerksamkeit erhält. Hierbei handelt es sich um einen Sammelbegriff für den Einsatz von Spielmechanismen (wie sie beispielsweise in Videospielen eingesetzt werden) in einem prinzipiell spielfremden Kontext. An dieser Stelle sollte nicht unerwähnt bleiben, dass sich der Begriff vom *Game-Based Learning* [Prensky 2003], also dem Lernen basierend auf einem Spiel, zwar abgrenzt, diese Abgrenzung aber nicht immer scharf gezogen werden kann.

Problemstellung dieser Masterarbeit ist es, für die Aufgaben des Biber-Wettbewerbs - also Aufgaben, die informatisches Konzept- und Paradigmendenken ohne Vorkenntnisse fördern - eine spielerische Lernumgebung in Form einer Applikation (*App*) für das mobile Betriebssystem Android zu entwerfen. Hierzu findet zunächst eine theoretische Auseinandersetzung mit dem Konzept *Gamification* statt. Auf Basis dieser Auseinandersetzung werden daraufhin Möglichkeiten identifiziert Elemente dieses Konzepts in den Prototypen der App einfließen zu lassen. Die Aufgaben (Tasks) sollen dabei aus den aktuellen Aufgabenheften des Bebras-Wettbewerbs stammen. Die Creative-Commons-Lizenz *CC BY-NC-SA 3.0* erlaubt hier die Erstellung von Derivaten, bei Namensnennung der Autoren, nichtkommerzieller Nutzung und Weitergabe unter gleichen Lizenzbedingungen.

Wissenschaftliche Fragestellung

Aus vorangehender Problemstellung schließend lässt sich die wissenschaftliche Fragestellung der vorliegenden Arbeit wie folgt formulieren:

> ***Wie können motivierende und beim Erlangen der Lernziele unterstützende Gamedesign-Elemente in eine mobile Applikation zum Erlernen von Konzepten der Informatik integriert werden?***

Methodisches Vorgehen

Zur Beantwortung der ebengenannten Fragestellung werden im Abschnitt ***Theoretische Grundlagen*** Gamedesign-Elemente, wie sie der Literatur zum Thema Gamification zu entnehmen sind, identifiziert und es wird auf deren lerntheoretischen Hintergrund eingegangen.

Auf Basis dieser Literaturgrundlagen wird im Abschnitt ***Designkonzept*** ein „gamifiziertes" didaktisches Konzept (*Design Document*) entwickelt, dessen Aufbau sich auf die Struktur der Biber-Aufgaben stützt. Weiters wird ein Überblick über das geplante User-Interface-Design des Prototyps in Form von Mock-Ups gegeben.

Die Dokumentation des abgeschlossenen Prototyps sowie technische Hintergrundinformationen zum Entwicklungsprozess finden sich im Abschnitt ***Implementierung des Prototyps***.

Im letzten Abschnitt ***Evaluierung*** schließlich wird eine *summative Evaluation* [Klante 1999] des Prototyps durchgeführt. Zur Durchführung dieser Evaluierung wird ein Online-Fragenkatalog erstellt, dessen Fokus auf folgenden zwei Themen liegt:

- *Motivation* (Wie sinnvoll wurden die Gamedesign-Elemente eingesetzt?)
- *Usability* (Wie gut lässt sich die Software benutzen?)

Ziel dieser Umfrage ist es einerseits, Mängel am Prototypen zu identifizieren, deren Behebung in zukünftige Versionen einfließen kann, andererseits die Beantwortung der wissenschaftlichen Fragestellung, sprich: *ist eine sinnvolle Integration der Gamedesign-Elemente in den Prototypen gelungen?*

State of the Art

Die wissenschaftliche Auseinandersetzung mit dem Spiel als wesentlichem Motivationsfaktor des Menschen ist nicht neu. So stellt der niederländische Kulturhistoriker Johan Huizinga bereits im Jahre 1938 dem Homo faber (dem tätigen Menschen) und dem Homo sapiens (dem denkenden Menschen) den Homo ludens (den spielenden Menschen) an die Seite und stellt die Behauptung auf, dass sich ohne die Lust und Fähigkeit des Menschens zum Spielen wesentliche Bereiche der menschlichen Kultur nicht entwickelt hätten [Huizinga 1938]. Was hingegen neu ist, ist die intensive wissenschaftliche Auseinandersetzung mit Computerspielen und den daraus entstehenden Implikationen, insbesondere jenen, die Auswirkungen auf unsere Art zu lernen haben können [Gee 2007].

Zur Nutzung in der Informatik-Lehre haben O'Rourke et al. eine iPhone-Applikation entwickelt, die den Bubble Sort-Algorithmus sowie die binäre Suche in Form eines Spieles an den Nutzer bringt. [O'Rourke 2010]

Weiters existiert ein breites Angebot an Programmiersprachen für Kinder, deren Ziel es ist, grundsätzliche Programmierparadigmata und -konzepte wie beispielsweise jene des objektorientierten Programmierens spielerisch zu vermitteln. Exemplarisch seien hier das Projekt Alice (http://www.alice.org) der Carnegie Mellon University und die Programmiersprache Scratch (http://scratch.mit.edu) des Massachusetts Institute of Technology (MIT) genannt, die auch im Speziellen zur Erforschung einer Didaktik für Informatik-Konzepte dienen. Weiterführende Informationen hierzu finden sich bei [Meerbaum 2010], [Ward 2010], sowie [Werner 2012].

Die direkte Kombination von Spielmechanismen - wie sie aus Videospielen bekannt sind - im spielfremden Kontext des Programmieren-Lernens ist noch relativ neu. Ansätze davon finden sich beispielsweise auf den Seiten Codecademy (http://www.codecademy.com) und Code School (http://www.codeschool.com).

Viele erhoffen sich derzeit Vorteile durch den Einsatz von Gamification. So sollen Spielmechanismen nicht nur das Engagement und die Motivation von Studentinnen und Studenten verbessern [Fitz-Walter 2011], sondern auch die Lernerfolge beim E-Learning drastisch verbessert werden [Raymer 2011], auch bei der Gestaltung von mobilen Apps und Web Apps gewinnt dieser Begriff zunehmend an Bedeutung [Zichermann 2011].

Bebras - Der Biber der Informatik

Entstehungsgeschichte und Überblick

Bei „Bebras“ handelt es sich um eine im Jahre 2004 in Litauen gegründete internationale Initiative von Prof. Valentina Dagienė, aus der der international stattfindende „Bebras Contest“ (http://www.bebras.org) hervorging. Hierbei handelt es sich um einen Wettbewerb, dessen Ziel es ist, Kinder und Jugendliche für grundlegende Denkweisen, Konzepte und Paradigmen der Informatik (z.B. Algorithmisches Denken, Informationsrepräsentation, Codierung, Verschlüsselung, Anwendung von IT-Systemen, Datenstrukturen etc.) zu begeistern. „Bebras“ ist die litauische Übersetzung für den „Biber“, welcher als „hart arbeitendes, intelligentes, zielorientiertes und lebhaftes Tier“ [Futschek 2009] zum Maskottchen und Namensgeber der Initiative wurde. Seit 2006 schlossen sich viele Länder der Initiative an, darunter unter anderem Estland, Lettland, die Slowakei, Deutschland, die Niederlande, die Ukraine, Tschechien, Polen, Slowenien, Italien, Finnland, die Schweiz, Frankreich, Ungarn, Israel, Kanada, Spanien, Japan und Zypern. Die teilnehmenden Länder haben teils eigens auf ihr nationales Bildungssystem abgestimmte Wettbewerbe. [Bebras: History] In Österreich findet der „Bebras Contest“ unter dem Namen „Der Biber der Informatik“ (http://www.ocg.at/de/biber) an vielen unterschiedlichen Schulen statt und wird von der „Österreichischen Computer Gesellschaft“ (OCG) organisiert. Ziel des Wettbewerbs ist es, Schülerinnen und Schüler dazu zu bewegen, eigenständig und explorativ Probleme, die für die Informatik typisch sind, zu lösen. Die Teilnehmenden sollen mit Konzepten und Prinzipien vertraut gemacht werden, ein tiefgehendes Verständnis von ebendiesen erlangen und moderne Technologien kreativ und intensiv einsetzen können. [Haberman 2011]

Die Teilnehmenden von Biber-Wettbewerben werden in bis zu fünf Altersgruppen [Biber-Aufgaben 2013] unterteilt:

- **Little Beaver**: 3. - 4. Schulstufe (9 - 10 Jahre)
- **Benjamin**: 5. - 6. Schulstufe (11 - 12 Jahre)
- **Meteor**: 7. - 8. Schulstufe (13 - 14 Jahre)
- **Junior**: 9. - 10. Schulstufe (15 - 16 Jahre)
- **Senior**: 11. - 13. Schulstufe (17 - 19 Jahre)

Bebras-Aufgaben (Tasks) sind jeweils mindestens einer dieser Altersgruppen zugeordnet, wobei der dem Task zugewiesene Schwierigkeitsgrad (leicht - mittel - schwer) je nach Altersgruppe unterschiedlich sein kann (vgl. [Dagienė 2008]).

Bebras-Tasks: Klassifikation und Kriterien

Ein Bebras-Contest besteht üblicherweise aus 15 - 21 Tasks von unterschiedlichem Schwierigkeitsgrad, die innerhalb von 45 Minuten zu lösen sind [Haberman 2011]. Die grundlegende Idee der Bebras-Tasks ist es, durch eigenständiges Lösen interessanter und unterhaltsamer Problemstellungen jene kognitiven Fähigkeiten zu entwickeln, die für analytisches, abstraktes, logisches, kurz: informatisches Denken vonnöten sind. In [Dagienė 2008] präsentieren Futschek und Dagienė fünf vom „International Bebras Organizing Committee“ vorgeschlagene Kategorien, in welche sich Bebras-Tasks einordnen lassen:

- **INF - Informationsverständnis („Information comprehension“)**
  Repräsentation (symbolisch, numerisch, visuell), Kodierung, Verschlüsselung
- **ALG - Algorithmisches Denken („Algorithmic thinking“)**
  inkl. Aspekte der Programmierung
- **USE - Anwendung von Computersystemen („Using computer systems“)**
  z.B. E-Mail, Suchmaschinen, Tabellenkalkulation
  Allgemeine Prinzipien (keine *spezifischen* Systeme)
- **STRUC - Strukturen, Muster und Gliederungen („Structures, patterns and arrangements“)**
  Kombinatorik, Datenstrukturen (z.B. Graphen)
- **PUZ - Puzzles**
  Logische Puzzles und Spiele
- **SOC - IKT und Gesellschaft („ICT and Society“)**
  Soziale, ethische, kulturelle, internationale und rechtliche Grundlagen

Weiters stellen Futschek und Dagienė [ebd.] einen vom selben Komitee entwickelten Kriterienkatalog für gute Tasks vor. Demzufolge ***sollen*** gute Aufgaben...

- eine **Verbindung zum Fach Informatik** bzw. zum allgemeinen Umgang mit Computern („computer literacy“) aufweisen
- **Lernerlebnisse ermöglichen** (Erfolgserlebnisse beim Lernen)
- **in drei Minuten lösbar sein**
- einem **Schwierigkeitsgrad** (leicht - mittel - schwer) zuzuordnen sein
- für mindestens eine der **Altersklassen** (Little Beaver, Benjamin, Meteor, Junior, Senior) geeignet sein
- **unabhängig von spezifischen Lehrplänen** sein

- **unabhängig von spezifischen Computersystemen** sein (keine spezifischen Programmiersprachen, Betriebssysteme oder Anwendungssoftware)
- **leicht zu verstehende Aufgabenbeschreibungen** haben
- auf einer **einzigen Bildschirmseite** darzustellen sein
- **an einem Computer** ohne zusätzliche Hardware, Software und ohne Papier und Stift **lösbar** sein
- **politisch korrekt** sein (Verzicht auf Stereotypen hinsichtlich Rasse, Geschlecht, Religion etc.)

Die nun folgenden Kriterien werden als wünschenswert, jedoch als optional bezeichnet, da es nicht *immer* möglich ist, sie alle zu erfüllen.

Gute Aufgaben ***sollten***...

- **Spaß machen** (eine gute Aufgabe weckt Interesse am Thema und ist nicht langweilig)
- **Bilder** beinhalten (visuelle Darstellungen können das Verständnis enorm erleichtern)
- **Interaktive Elemente** beinhalten (Simulationen, direkte Manipulation etc.)
- **unmittelbares Feedback** geben (der Teilnehmende ist sich unmittelbar nach Abschluss einer Aufgabe sicher, diese korrekt gelöst zu haben)

Auf Fangfragen und trickreiche Fragestellungen sollte verzichtet werden, da diese den gegenteiligen Effekt bei den Teilnehmenden auslösen und in Folge statt zur Motivation zur Erhöhung der Frustration beitragen können.

Bebras-Tasks: Beispiele

Folgend nun zwei Beispiele für gute Bebras-Tasks aus dem Aufgabenkatalog zum „Biber der Informatik 2012“ [Biber-Aufgaben 2012]:

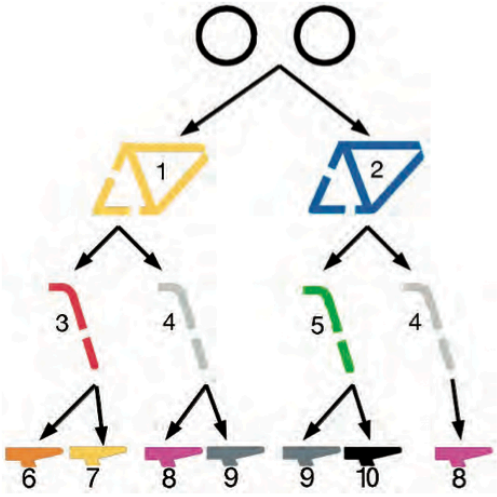
Fahrradkult

In Bebras-City ist es gerade Kult, die Fahrräder sehr bunt zu gestalten.

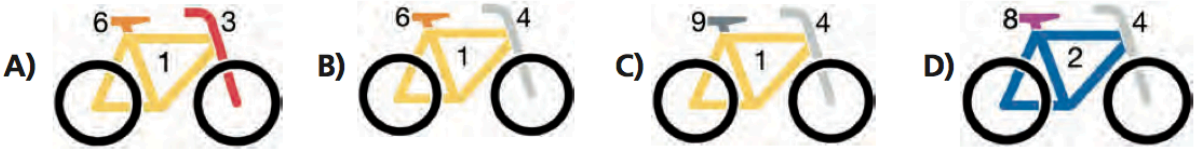
Allerdings hat das Stadtparlament die erlaubten Teile durchnummeriert und eine Vorschrift beschlossen, wie Fahrräder zusammengesetzt werden dürfen.

Im Bild ist festgelegt, aus welchen farbigen Teilen du ein Fahrrad zusammensetzen darfst.

Du beginnst oben mit den Rädern. Dann entscheidest du dich beim nächsten Teil, einem Pfeil folgend, für eine der Möglichkeiten. Und so weiter.

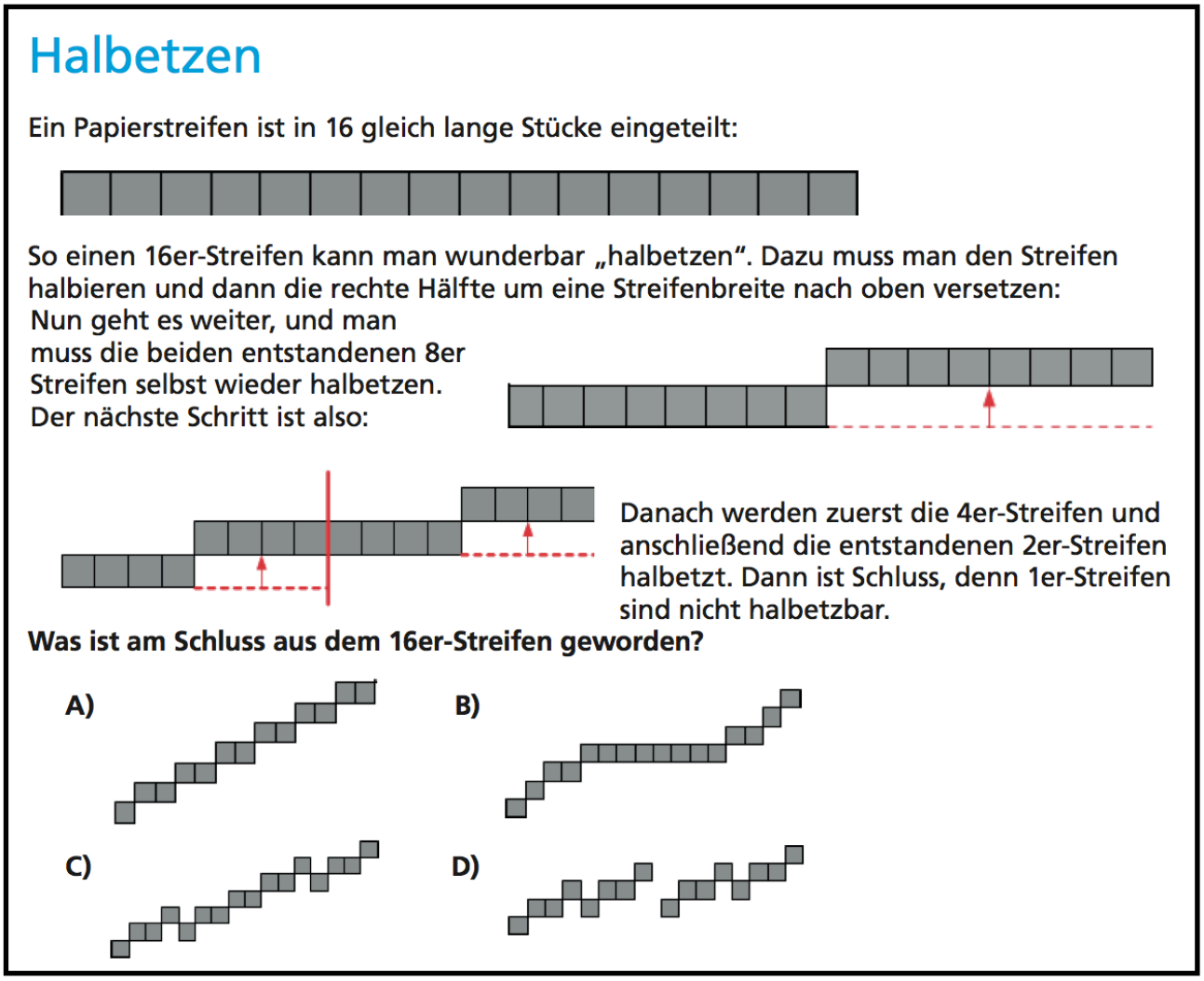


Welches dieser Fahrräder entspricht NICHT der Vorschrift des Stadtparlaments?



[Abb.BiberTask1] „Fahrradkult“ - Binärer Baum

Der Bebras-Task „Fahrradkult“ wurde für die 5. und 6. Schulstufe (Altersklasse „Benjamin“) und den Schwierigkeitsgrad „mittel“ konzipiert. Er ist der Kategorie **STRUC** (Strukturen, Muster und Gliederungen) zuzuordnen, da für seine Erfüllung ein Verständnis der für die Informatik wichtigen Datenstruktur des *binären Baumes* notwendig ist. Dieses Verständnis wird jedoch nicht vorausgesetzt, sondern kann durch Nachdenken bzw. geistiges Ausprobieren erlangt werden. Die Fahrräder A, C und D entsprechen gültigen Endzuständen (*Blättern*) des Baumes. Auf Fahrrad **B** trifft dies nicht zu, da Knoten 6 (oranger Sattel) nach vorangehender Auswahl von Knoten 1 (gelber Rahmen) und Knoten 4 (graue Lenkstange) nicht mehr gewählt werden kann.

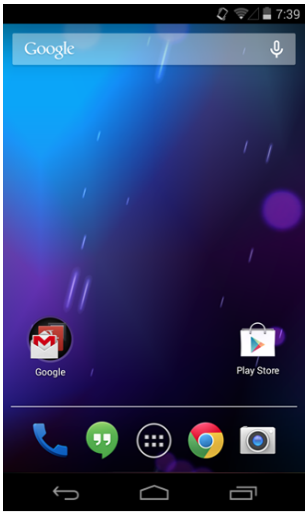


[Abb.BiberTask2] „Halbetzen" - Ein rekursiver Algorithmus

Der Task „Halbetzen" wurde für die 11. und 13. Schulstufe (Altersklasse „Senior") konzipiert und hat den Schwierigkeitsgrad „mittel". Er kann der Kategorie **ALG** (Algorithmisches Denken) zugeordnet werden, da er einen *rekursiven Algorithmus* und das in der Informatik vorkommende Prinzip des „Teile und herrsche" („Divide et impera") beschreibt. Wiederum setzt die Aufgabenstellung keinerlei Kenntnisse über Algorithmen bzw. Rekursion voraus. Durch geistiges Anwenden des in der Aufgabenstellung vorgestellten rekursiven Algorithmus (oder durch Ausprobieren mit Papier) lässt sich feststellen, dass **D** der Endzustand des Papierstreifens nach Anwendung des Algorithmus ist.

Das mobile Betriebssystem Android

Geschichte



Der Android-Homescreen [Abb.Android.Home]

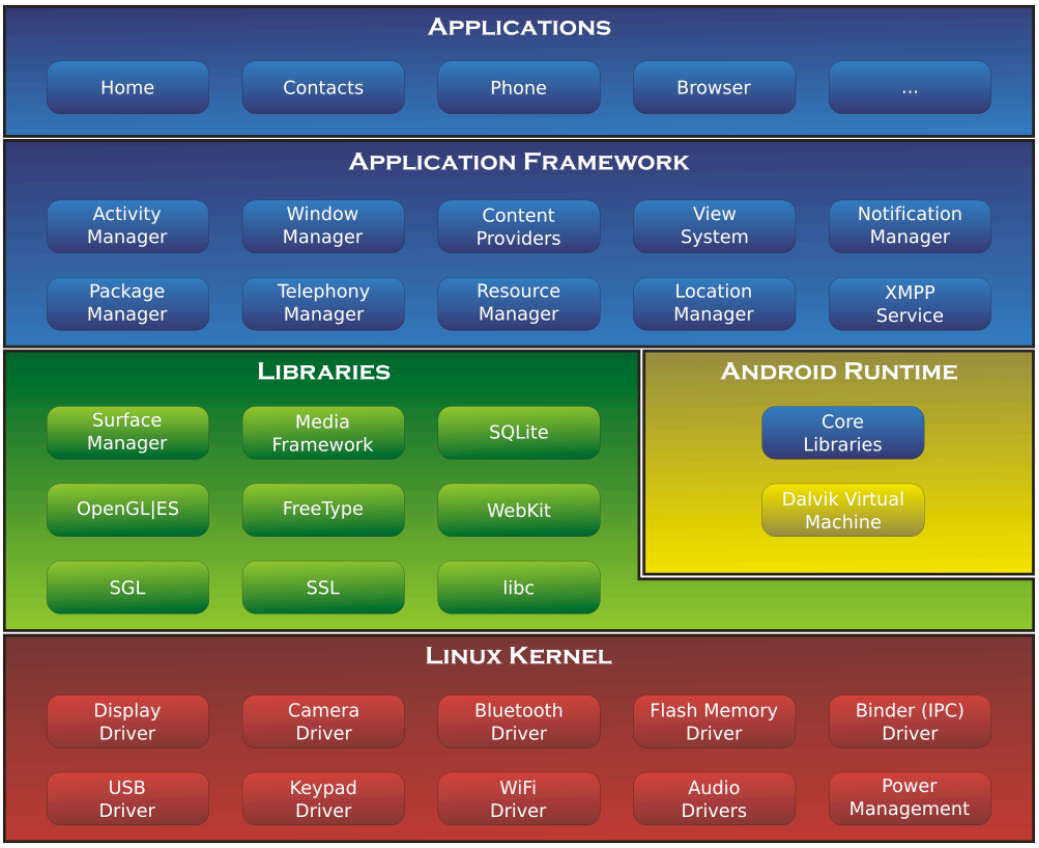
Android ist ein von der Open Handset Alliance, deren Hauptmitglied der IT-Konzern Google ist, entwickeltes Betriebssystem für mobile Geräte wie beispielsweise Smartphones und Tablet-PCs. Das Unternehmen Android wurde im Jahr 2003 von Andy Rubin gegründet und 2005 von Google übernommen. Android ist Open-Source bzw. freie Software, wird also quelloffen entwickelt. Jedoch sind wesentliche Bestandteile, die mit Android ausgeliefert werden, wie z.B. die Google Suche sowie andere Google-Produkte wie beispielsweise Google Maps proprietär, weshalb Android von der Free Software Foundation kritisch betrachtet wird. Android basiert auf dem Linux-Kernel, kann jedoch nicht als typische Linux-Distribution bezeichnet werden. [WP.de:Android 2013]

Seit Googles Übernahme hat Android eine rasante Entwicklung hinter sich. Im Jahr 2010 überholte Android das bis dato am weitesten verbreitete mobile Betriebssystem Symbian und wurde damit zur weltweit meistverbreiteten mobilen Plattform. Im Jahr 2013 lag der prozentuelle Anteil von Android-Smartphones (deren größter Hersteller der südkoreanische Elektronikkonzern Samsung war) weltweit bereits bei 80 Prozent. Programme für die Plattform werden als *Apps* bezeichnet und über den *Google Play Store* vertrieben. Im Mai 2013 lag die Anzahl der bis dahin aus dem Google Play Store heruntergeladenen und installierten Apps bei 48 Milliarden, die Anzahl aktivierter Android-Geräte lag im September 2013 bei einer Milliarde. [WP.en:Android 2014]

Technische Grundlagen

Die aktuelle Android-Version 4.4 „KitKat“ (Android-Versionen haben die Namen amerikanischer Süßspeisen und Desserts, so hießen die bisherigen Versionen Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich und Jelly Bean) basiert auf dem Linux-Kernel 3.4.10. Aufbauend auf den Linux-Kernel stellt die Systemarchitektur von Android [Abb.Android.Arch] Middleware, Bibliotheken (Libraries), Programmierschnittstellen (APIs) und ein Application Framework zur Verfügung, das mit Java kompatible Libraries beinhaltet. Als Laufzeitumgebung nutzt Android neben diversen Laufzeit-Bibliotheken die Dalvik Virtual Machine. Diese kompiliert üblicherweise Java-Bytecode in sogenannten „dex-code“ (Dalvik Executable Code)

und führt diesen aus. Weiters nutzt Android statt der unter Linux üblichen GNU C Bibliothek (glibc), die Eigenentwicklung „Bionic“, die speziell für Systeme mit geringer Prozessorleistung entwickelt wurde. [WP.en:Android 2014]



[Abb.Android.Arch] Android System-Architektur

Android Apps können mit Hilfe des *Android Software Development Kits* (SDK) entwickelt werden, das von der Android-Entwickler-Plattform (http://developer.android.com) frei bezogen werden kann. Das Android SDK enthält eine Reihe von für die App-Entwicklung hilfreichen Tools, darunter einen Debugger, Software-Bibliotheken, einen Emulator zum Simulieren unterschiedlicher Android-Geräte (Smartphones, Tablets), Code-Beispiele und Tutorials [WP.en:Android 2014].

Für die Implementierung der in dieser Arbeit konzeptionierten App kam die integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) *Eclipse* zum Einsatz, für welche Google das *Android Developer Tools* (ADT) *Plugin* zur Verfügung stellt.

Theoretische Grundlagen

Spiel

Bevor wir uns der Definition des Begriffes Gamification zuwenden, ist zunächst klarzustellen, wie der Begriff des Spieles in dieser Arbeit zu verstehen ist. Ich möchte mich hierbei an die Definition des niederländischen Kulturhistorikers Johan Huizinga halten:

„Spiel ist eine freiwillige Handlung oder Beschäftigung, die innerhalb gewisser festgesetzter Grenzen von Zeit und Raum nach freiwillig angenommenen, aber unbedingt bindenden Regeln verrichtet wird, ihr Ziel in sich selber hat und begleitet wird von einem Gefühl der Spannung und Freude und einem Bewusstsein des *Andersseins* als das *gewöhnliche Leben*." [Huizinga 1938:37]

Folgende weitere Definition zerlegt das Spiel in seine einzelnen Elemente:

„Ein Spiel ist ein System, in dem Spieler an einer abstrakten Herausforderung („challenge") arbeiten, definiert durch Regeln, Interaktivität und Feedback, die ein messbares Ergebnis („quantifiable outcome") hat und häufig eine emotionale Reaktion hervorruft." [Kapp 2012:7]

Gamification

„Bei Gamification handelt es sich um einen Sammelbegriff für den Einsatz von Videospiel-Elementen in Systemen, die per se keine Spiele sind, mit dem Ziel der Verbesserung der Benutzer-erfahrung (User Experience) und der Einbindung der Nutzer (User Engagement)." [Deterding 2011|1]

Weiters findet sich bei [Kapp 2012] folgende Definition:

„Gamification ist der Einsatz von spielbasierter Mechanik, Ästhetik und Spiel-Denkweisen, um Leute einzubinden, sie zum Handeln zu motivieren, sie beim Lernen zu fördern und Probleme zu lösen." [Kapp 2012:10]

Im Gegensatz zum *Digital Game-Based Learning* [Prensky 2003] geht es also nicht um den Einsatz vollständiger Spiele, sondern um die Betrachtung jener separaten Spielelemente, die *ein Spiel zum Spiel machen* und wie sich diese dazu nutzen lassen können, Services und Applikationen unterschiedlichster Art motivierender zu gestalten. Diese Spielelemente werden in [Raymer 2011] als Regelkonstrukte, die die Nutzer mit Hilfe von Feedback-Mechanismen dazu ermutigen, die Eigenschaften ihres „Raums an Möglichkeiten" („possibility spaces") zu erforschen und kennenzulernen, bezeichnet. So seien diese Möglichkeitsräume durch den Einsatz von Gamification mittlerweile auf andere Themengebiete wie z.B. Marketing, Bildung, Social Media, E-Learning und das Web ausgedehnt worden.

Die *Ziele* von Gamification können sehr unterschiedlich ausfallen und hängen von den Gestaltern ebendieser Services und Applikationen ab. So kann der Gestalter einer *gamifizierten* Webplattform es sich beispielsweise zum Ziel gesetzt haben, die Verweildauer der User auf der Webseite zu erhöhen, die Partizipation der User im Forum durch spieltypische Reputationssysteme (z.B. Belohnung in Form einer virtuellen Währung oder mit Auszeichnungen/*Badges*) zu steigern oder - im Marketingkontext - den Kauf eines bestimmten Produkts anzuregen bzw. die emotionale Bindung an eine Marke positiv zu beeinflussen.
Es sei an dieser Stelle auch darauf hingewiesen, dass es sich bei Gamification um einen aktuellen Trend handelt, an welchen *Erwartungen* gestellt werden, die dieser nicht in allen Belangen erfüllen wird können. Das bloße Hinzufügen spielerischer Elemente in einen per se nicht interessanten Kontext macht diesen nicht notwendigerweise interessanter. Auch sind beispielsweise einfache Belohnungssysteme (Punkte, Medaillen) nicht ausreichend, wenn diese keinen konkreten spielerischen Nutzen haben oder in keiner erkennbaren Relation zur erbrachten Leistung stehen. Sebastian Deterding bringt im Interview in [Bozarth 2011] folgendes Beispiel: Besiegt man ein Boss-Monster und erhält dafür 20 000 Punkte ist man nicht doppelt so froh wie über 10 000 Punkte, sondern wenn es *doppelt so schwierig* war wie ein 10 000-Punkte-Monster.
Gerade im Umfeld des Lernens und der Didaktik kann der Einsatz von Gamification jedoch durchaus sinnvoll sein. Durch die Vernetzung spielerischer Elemente - eingebettet in ein didaktisches Konzept - kann Gamification unsere intrinsische Motivation zu lernen steigern.

„Wir spielen Spiele, weil wir von Natur aus Vergnügen an dieser Aktivität finden."

- Sebastian Deterding in [Bozarth 2011]

Demzufolge kann ein abgeschlossener Gamification-Prozess nur dann als erfolgreich bewertet werden, wenn wir an der gamifizierten Aktivität auch tatsächlich Spaß haben und gerne bereit sind, diese freiwillig auszuüben.

Im folgenden Abschnitt wenden wir uns nun jenen **Lern- und Motivationstheorien** zu, die dem Konzept Gamification zu Grunde liegen.

Lerntheoretischer Hintergrund

Der folgende Abschnitt ist eine Auflistung von lernpsychologischen Theorien, die für den Einsatz von Gamification wesentliche theoretische Grundlagen bilden. An dieser Stelle sei darauf hingewiesen, dass diese Liste keineswegs vollständig ist, sondern eine Auswahl jener Theorien darstellt, die für das im nächsten Abschnitt folgende Designkonzept von Bedeutung sein können. Eine umfangreiche Liste findet sich bei [Kapp 2012:51ff.], die folgenden vorgestellten Theorien stellen eine Auswahl aus dieser Liste dar.

Extrinsische Motivation

Von extrinsischer Motivation ist dann die Rede, wenn das gewünschte Resultat nicht der Abschluss der Aktivität an sich ist, sondern von außen (external) kommt. Beispielsweise kann das Erreichen einer guten Note die eigentliche Motivation für die Arbeit an einer Lernaktivität sein, ohne dass man dafür notwendigerweise Freude am Ausüben dieser Lernaktivität empfindet. Ein anderes Beispiel ist das Ausüben einer Tätigkeit aufgrund der Bezahlung, die man für diese Tätigkeit erhält. Extrinsische Motivation ist also im weitesten Sinne das Streben nach Belohnung bzw. die Vermeidung von Bestrafung, wobei diese Belohnung bzw. Bestrafung in keiner direkten Relation zur Aktivität stehen muss. [Kapp 2012:52f.]

Intrinsische Motivation

Von intrinsischer Motivation sprechen wir, wenn eine Lernaktivität aus purer Freude an der Aktivität selbst (internal) bzw. an den Lern- und Erfolgserlebnissen, die diese bietet, ausgeübt wird. Diese Freude hängt wesentlich von den Interessen und der Psyche des Lernenden ab - sie kommt also von *innen* und ist unabhängig von äußeren Einflüssen (wie Belohnung oder Bestrafung). Da der Prozess des Ausführens der Aktivität (im Sinne von: *der Weg ist das Ziel*) bereits die Motivation darstellt, ist sie auch unabhängig vom Resultat.

„Intrinsisch motivierte Lernende tendieren dazu, ein weites Spektrum an Phänomenen bewusst wahrzunehmen, während sie gleichzeitig Komplexitäten, Inkonsistenzen, neuartigen Ereignissen und unerwarteten Möglichkeiten gegenüber aufgeschlossen sind.“ [Kapp 2012:52] Hierfür benötigen sie allerdings den notwendigen Zeit- und Gestaltungsspielraum, um ihre Kreativität frei und selbstbestimmt zu entfalten.

Die meisten Gamification-Konzepte kombinieren Elemente extrinsischer mit Elementen intrinsischer Motivation, wobei für ein nachhaltiges Konzept der Schwerpunkt auf die **intrinsische** Motivation gelegt werden sollte.

Leppers Design-Grundsätze intrinsischer Motivation in der Lehre

Für die Implementierung einer Didaktik, die intrinsische Motivation fördert, wurden von Mark Lepper, Forscher an der Stanford University, vier Grundsätze (Instructional Design Principles for Intrinsic Motivation) vorgestellt [Kapp 2012:57f.]:

- **Kontrolle (Control):** Lernende haben die Möglichkeit an ihren Lernaktivitäten durch eigene Entscheidungen, die möglichst unabhängig von extrinsischen Einflüssen (z.B. Belohnungen) sein sollten, mitzugestalten. Sie können beschließen, wann eine Aktivität gestartet oder beendet wird und kontrollieren ihren eigenen Lernfortschritt. Der Einsatz extrinsischer Belohnungen kann allerdings hilfreich sein, um eine zunächst uninteressant scheinende Aktivität interessanter zu machen.
- **Herausforderung (Challenge):** Lernaktivitäten stellen kontinuierliche Herausforderungen dar, die den jeweiligen Fähigkeiten und dem jeweiligen Wissensstand der Lernenden entspricht. Sie sind weder zu einfach noch zu schwer und geben *unmittelbares Feedback*. Man weiß also direkt nach Ausführung einer Lernaktivität über Erfolg oder Misserfolg bescheid. Misserfolg ist demzufolge eine Möglichkeit zu lernen und nicht etwas, das um jeden Preis vermieden werden muss.
- **Wissbegierde (Curiosity):** Lernaktivitäten sollen die Neugier der Lernenden wecken. Dies kann über unterschiedliche Mittel und Wege erreicht werden. Beispielsweise durch bewusstes Einbauen von Inkonsistenzen, Lücken in der Story, Weglassen von Erklärungen oder nicht dokumentierte Funktionen („Easter Eggs“), sodass dem Lernenden die Möglichkeit gegeben wird, selbst auf Lösungen zu kommen und „Aha“-Erlebnisse zu haben.
- **Kontextualisierung (Contextualization):** Die Nützlichkeit und Sinnhaftigkeit der Lernaktivitäten müssen innerhalb ihres Kontextes (z.B. der Geschichte in die sie eingebettet sind) deutlich ersichtlich sein.

Malones Theorie intrinsisch motivierender Lehre

Eine weitere Theorie aus dem Jahr 1980 über die Elemente intrischer Motivation stammt von Thomas Malone, die aus dessen Beschäftigung mit der Frage, warum Spiele Spaß machen, entstand. Malone identifiziert drei Schlüsselelemente, die für das Entstehen intrinsischer Motivation verantwortlich sind [Kapp 2012:55f.]:

- **Wissbegierde (Curiosity):** siehe *Leppers Design-Grundsätze*
- **Herausforderung (Challenge):** siehe *Leppers Design-Grundsätze*
- **Fantasie (Fantasy):** Eine Umgebung, die „mentale Bilder von Dingen hervorruft, die durch die Sinne gegenwärtig nicht direkt wahrgenommen werden können oder sich in der direkten Erfahrungswelt der involvierten Person befinden“.

Taxonomie intrinsischer Motivation

Eine Zusammenführung der beiden vorangegangenen Theorien von Lepper und Malone findet sich in der „Taxonomie intrinisischer Motivation“ („The Taxonomy of Intrinsic Motivations“) wieder. Die Taxonomie gliedert sich in zwei Teile [Kapp 2012:58f.]:

Teil 1: Innerliche Motivation

- **Herausforderung (Challenge):** Ziele, unvorhersagbare Ergebnisse, Feedback zur erbrachten Leistung, Selbstwertgefühl
- **Wissbegierde (Curiosity):** sowohl im Sinne von *sensorischer* als auch *kognitiver* Neugier
- **Kontrolle (Control):** Entscheidungen treffen, Optionen wählen, auf Unvorhergesehenes reagieren, Macht und Autonomie ausüben
- **Fantasie (Fantasy):** emotionale und kognitive Aspekte, Verzahnung der Vorstellungskraft der Lernenden mit den zu lernenden Fähigkeiten

Teil 2: Zwischenmenschliche Motivation

- **Kooperation (Cooperation):** Zusammenarbeit von Lernenden (bzw. Spielenden) mit der Absicht ein gemeinsames Ziel innerhalb des Spiels zu erreichen.
- **Konkurrenz (Competition):** Gegen andere bzw. einen anderen Spieler arbeiten, um ein Ziel zu erreichen (z.B. in Form eines Wettkampfs).
- **Anerkennung (Recognition):** Erfolge und Errungenschaften (*Achievements*), die für andere erkennbar sind, sodass die harte Arbeit, die notwendig ist, um das Spiel zu meistern, entsprechend anerkannt wird.

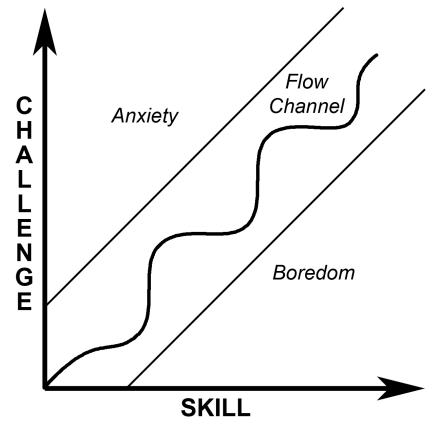
Selbstbestimmungstheorie der Motivation

Die Selbstbestimmungstheorie der Motivation (Self-Determination Theory) besagt, dass es im wesentlichen drei Faktoren sind, die unsere Motivation ausmachen [Bozarth 2011]:

- **Kompetenz** - die Fähigkeit, jene an einen gestellten Herausforderungen zu bewältigen
- **Autonomie** - die Möglichkeit freie Entscheidungen zu treffen und selbstbestimmt zu handeln
- **Eingebundenheit** (***Relatedness***) - die Zugehörigkeit zu einer Gruppe oder die Kooperation mit bzw. die Konkurrenz zu anderen

Jedes gute Spiel beinhaltet diese drei Faktoren auf die eine oder andere Weise: Innerhalb ihres durch ihre Regeln eingeschränkten Handlungsspielraums an Möglichkeiten stellen Spiele Herausforderungen an uns, die wir mittels unserer aktuellen Fähigkeiten lösen können (Kompetenz). Sie geben uns die Möglichkeit frei über unsere Spielzüge, Strategien und Handlungen innerhalb des Spiels zu enscheiden (Autonomie). Weiters befinden wir uns entweder in Kooperation mit unseren Mitspielern oder in Konkurrenz zu unseren (realen oder simulierten) Gegenspielern (Relatedness).

Flow



Der „Flow Channel“ befindet sich zwischen den Zuständen Angst und Langeweile [Abb.Flow.Channel]

Flow ist eine von Mihály Csíkszentmihályi entwickelte Theorie [Csíkszentmihályi 1990] über den Zustand des völligen Aufgehens in einer Tätigkeit, der gleichsam mit dem Verlust des Zeitgefühls und des Selbstkonzepts einhergeht. Voraussetzungen für Aktivitäten, die Flow erzeugen, sind unter anderem die Setzung klarer Ziele, unmittelbares Feedback und das Tun einer Tätigkeit um ihrer selbst Willen (dies bezeichnet Csíkszentmihályi als „autotelisch“ - die Tätigkeit hat ihr Ziel also in sich selbst).

Den sogenannten „Flow Channel“ [Abb.Flow.Channel] platziert Csíkszentmihályi zwischen den psychologischen Zuständen Angst (Überforderung) und Langeweile (Unterforderung). Hohe Herausforderungen („challenges“) bei gleichzeitig zu geringen Fähigkeiten („skills“) führen zu Angst bzw. Überforderung. Niedrige Herausforderungen bei gleichzeitig hohen Fähigkeiten führen zu Langeweile bzw. Unterforderung. Flow kann nur dann entstehen, wenn die Herausforderungen den jeweiligen Fähigkeiten der Person angemessen sind. Steigen die Fähigkeiten der Person, müssen also auch die Herausforderungen schwieriger werden und vice versa. Der „Flow Channel“ weist demzufolge in Richtung stetig wachsender Komplexität.

ARCS-Modell

Beim ARCS-Modell handelt es sich um ein von John Keller vorgestelltes Modell für *Instruktionsdesign*, das vor allem bei der Gestaltung von Lehr- und Lernsoftware sowie beim E-Learning zum Einsatz kommt. ARCS ist ein Akronym, das für die folgenden Komponenten steht [Kapp 2012:53f.]:

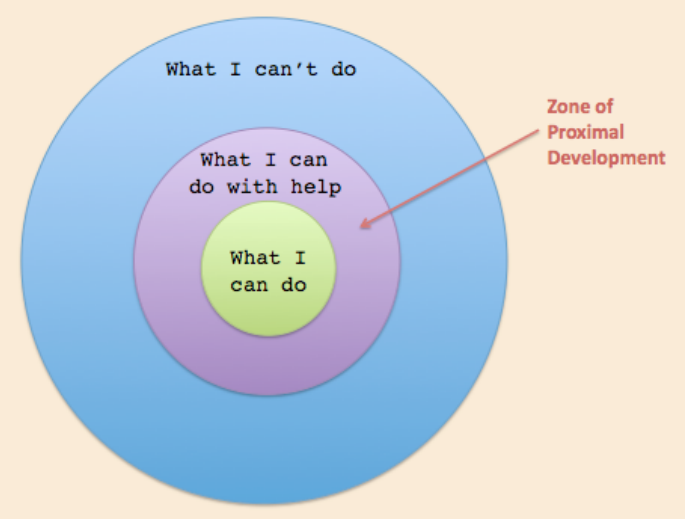
- **Aufmerksamkeit** (*Attention*): Das Lernmaterial muss die Aufmerksamkeit der Lernenden gewinnen. Dies kann auf unterschiedliche Arten geschehen: beispielsweise durch das Anregen von Neugierde, das Auftreten von Überraschungen (Element des Zufalls / *Variabilität*) oder *Fragestellungen*, die interessant und von unmittelbarem Nutzen sind.
- **Relevanz** (*Relevance*): Die Relevanz des Lerninhalts muss auf eine dieser Arten gegeben sein:
  (a) *Zielorientierung*: Es ist klar, welches Ziel es zu erreichen gilt, und warum dieses Ziel wichtig ist.
  (b) *Motivübereinstimmung*: Das Motiv (die Absicht) der Lernenden stimmt mit dem Motiv der Instruktion überein (z.B. Risikofreude, Leistungsbedürfnis, Wunsch nach Anerkennung)
  (c) *Vertrautheit*: Neues Wissen lässt sich einfach in die vorhandene Wissensbasis eingliedern und mit dieser vernetzen, die Zusammenhänge sind gut erkennbar.
- **Zuversicht** (*Confidence*): Aufgaben müssen derart gestaltet sein, dass die Lernenden zuversichtlich sein können, diese erfolgreich abzuschließen. Es ist sinnvoll, größere Aufgaben in kleinere Teilaufgaben zu zerlegen, um regelmäßige Erfolgserlebnisse zu ermögichen, die die Zuversicht steigern.
- **Zufriedenheit** (*Satisfaction*): Zufriedenheit mit Lernmaterialien kann beispielsweise durch positive Verstärkung (Lob und Belohnung) für gute Leistungen herbeigeführt werden. Wesentlich wertvoller ist es allerdings, wenn die Lernenden ihr gelerntes Wissen auch anwenden können. Gute Instruktion zeigt also Anwendungsmöglichkeiten des Wissens auf, das sie vermittelt. Weiters ist sie konsistent in ihrem Aufbau und ihren Bewertungskriterien.

Scaffolding

Hierbei handelt es sich um ein Lehrkonzept, das auf der vom konstruktivistischen Psychologen Lev Vygotsky entwickelten *„Zone der proximalen Entwicklung“* („Zone of proximal development“) basiert. Diese ist laut Vygotsky *der Abstand zwischen dem aktuellen Entwicklungsstand der Problemlösungsfähigkeiten eines Kindes und der Größe des Potentials an Problemlösungsfähigkeiten unter Anleitung eines Erwachsenen oder in Zusammenarbeit mit fortgeschrittenen Gleichaltrigen.* [Kapp 2012:66f.]

Scaffolding (engl.: Gerüst) ist eine Technik, die die Lernenden schrittweise anleitet, bis sie in der Lage sind, die Aktivität selbst auszuführen. Hierbei werden den Lernenden Orientierungshilfen, Tipps und Unterstützung gegeben, um Aufgaben zu bewältigen, die sie ohne diese Hilfestellungen nicht bewältigen würden. Sobald diese Hilfen nicht mehr benötigt werden, werden sie Schritt für Schritt entfernt.

In Computerspielen wird Scaffolding auch in Form von steigender Levelkomplexität umgesetzt: in den Levels lernt man unter Anleitung jene Fähigkeiten, ohne die man die jeweils folgenden Levels nicht abschließen könnte. Begonnen wird mit einem unkomplizierten Interface, einfachen Aufgaben und wenigen Handlungsoptionen. In jedem Folgelevel nehmen diese an Komplexität und damit an Möglichkeiten zu.



„Zone of Proximal Development“ (Vygotsky)
[Abb.ZPD]

Gamedesign-Elemente

Basierend auf dem lerntheoretischen Hintergrund des vorangegangenen Abschnitts ist es nun an der Zeit, jene Gamedesign-Elemente zu identifizieren, die für die Umsetzung eines Gamification-Konzepts notwendig sind.

Zehn Zutaten großartiger Spiele

Reeves und Read nennen in [Reeves 2009] die *„Zehn Zutaten großartiger Spiele“* („Ten ingredients of great games“): (1) Selbstrepräsentation mit Avataren, (2) dreidimensionale Umgebungen, (3) narrativer Kontext, (4) Feedback, (5) Reputation, Ränge und Levels, (6) Marktplätze und Ökonomien, (7) Wettbewerb unter klaren, bindenden Regeln, (8) Teams, (9) Parallele Kommunikationssysteme, die leicht konfigurierbar sind, (10) Zeitdruck.

Eine ausführliche Beschreibung von Gamedesign-Elementen findet sich zudem bei [Kapp 2012:25ff.]. Eine Auswahl von Gamedesign-Elementen, die speziell für den Einsatz im E-Learning geeignet sind, beschreibt [Raymer 2011]. Da sich die beiden Listen inhaltlich überschneiden, seien sie hier zusammenfassend und teils durch eigene Überlegungen ergänzt dargestellt:

Abstraktion von Konzepten und der Wirklichkeit

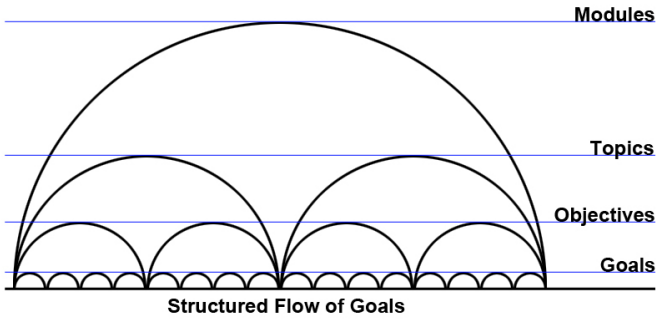
Ein wesentliches Element von Spielen ist, dass sie grobe Generalisierungen von Konzepten und Abstrahierungen der Wirklichkeit vornehmen. Das daraus resultierende Modell wird auch als *„operating model“* bezeichnet, da der Spieler bzw. die Spielerin seine Handlungen (Operationen) innerhalb dieses Modells ausübt und das Modellsystem dadurch verändert. Diese Operationen sind vom Spiel vorgegeben und üblicherweise begrenzt (beispielsweise mögliche Zugoptionen beim Schach).

Bei der Abstraktion von Ereignissen, Ideen und Wirklichkeit kommen *Repräsentationen* zum Einsatz, die auch hypothetisch, imaginär oder fiktional sein können. So repräsentieren die Schachfiguren beispielsweise Krieger auf einem Schlachtfeld, die Kriegssituation ist in ihrer Komplexität jedoch auf die Größe eines simplen Spielbretts reduziert.

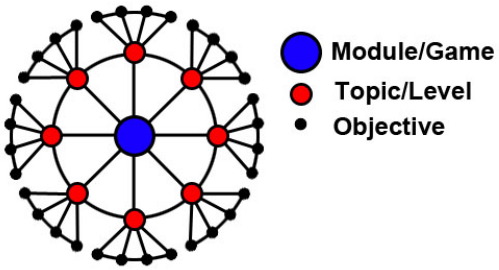
Abstraktion ermöglicht auch die Implementierung simpler Kausalitätsbeziehungen (z.B. hat eine reale Steuererhöhung komplexere Auswirkungen als in einer Städtesimulation), das Nichtvorhandensein von äußeren Störfaktoren (die aufgrund der simplifizierten Darstellung nicht abgebildet werden müssen) und die einfachere Erlernbarkeit von Konzepten aufgrund von Fehlertoleranz (z.B. zieht ein Unfall in einem Autorennspiel nicht automatisch eine Autoreparatur oder einen Krankenhausaufenthalt nach sich). [Kapp 2012:26f.]

Ziele

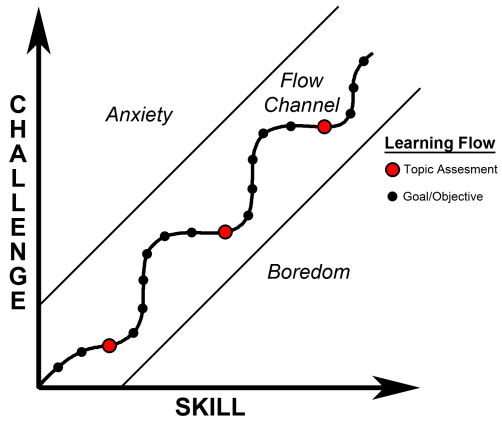
Das Hinzufügen von Zielen ist für viele jenes entscheidende Element, das eine Spielerei („play“) zum Spiel („game“) macht. So ist Herumlaufen im Park zwar zweifellos eine schöne Beschäftigung, jedoch noch kein Spiel. Erst die Setzung eines Zieles - wie beispielsweise schneller als die anderen zu einem bestimmten Punkt (Ziel) zu laufen - macht das Spiel aus. Ein Ziel fügt Sinn, Fokus und ein messbares Ergebnis zur Spielerei hinzu, es ist eine Methode, die Qualität spielerischer Handlungen messbar zu machen. Ziele sind spezifisch und klar. Es ist eindeutig, wann ein Ziel erreicht wurde. In den meisten Spielen ist das auch visuell gut sichtbar. So ist etwa bei einem Autorennen jederzeit gut erkennbar, welcher Spieler bzw. welche Spielerin führt. Auch auf einem einfachen Spielbrett sieht man meistens (z.B. anhand eines Markierungsspielsteins), wie weit man vom Ziel des Spieles im aktuellen Moment entfernt ist. Eine visuelle Darstellung des bisherigen Fortschritts und damit auch der Entfernung zum Ziel ist eine effektive Methode des Feedbacks, die den Vergleich mit anderen Mitspielenden erlaubt. Es sollte möglich sein, Ziele mit unterschiedlichen Strategien oder Ansätzen zu erreichen. [Kapp 2012: 28f.]



Lineare Strukturierung von Lernzielen
[Abb.Goals.linear]



Nichtlineare Strukturierung von Lernzielen
[Abb.Goals.nonlinear]



Sequenz von Teilzielen und Modulzielen (*Topic Assessments*) im E-Learning
[Abb.Flow.Goals]

Überdies müssen Ziele gut strukturiert sein. Während ein Spiel ein großes Gesamtziel haben sollte, ist es wichtig, dieses in kleinere Einheiten (Teilziele) zu zerlegen. Dies kann entweder linear [Abb.Goals.linear] oder nichtlinear [Abb.Goals.nonlinear] geschehen, was dem (im Abschnitt *Bebras-Tasks: Beispiele* bereits erwähnten) Prinzip des „Divide and Conquer“ („divide et impera“) entspricht. Raymer [Raymer 2011] erwähnt außerdem, dass gute Spiele und gutes E-Learning ihre Teilziele derart setzen, dass sie sich im Flow-Channel (siehe Abschnitt *Flow*) befinden [Abb.Flow.Goals].

Regeln

Die wesentliche Essenz eines jeden Spiels sind seine Regeln. Regeln legen fest, welche Handlungsoptionen die Spieler/innen haben, welche Voraussetzungen gegeben sein müssen, um das Spiel zu spielen und welche Zustände gültige Endzustände (Spielziele) sind. Es lassen sich verschiedene Kategorien von Regeln in Spielen identifizieren [Kapp 2012:30f.]:

- **Operationale Regeln** („*Operational Rules*“): Jene Regeln, die beschreiben, wie das Spiel gespielt wird. Beispielsweise ist die Regel „Wer eine 6 würfelt, zieht 6 Felder weiter und darf sofort noch einmal würfeln“ eine operationale Regel. Sobald man ein grundlegendes Verständnis der operationalen Regeln eines Spieles hat, kann man es spielen.
- **Konstitutive Regeln** („*Constitutive or Foundational Rules*“): Dies sind die Regeln, die dem Spiel zugrunde liegen und nur vom Spielautor bzw. Spielentwickler verstanden werden müssen. Ein Verständnis der konstitutiven Regeln kann einem Spieler / einer Spielerin allerdings einen Vorteil verschaffen: Wird z.B. ein Spiel mit 2 Würfeln gespielt, so ist die Wahrscheinlichkeit des Würfelergebnisses (Augensumme) 7 mit 1:6 (0,167) deutlicher höher als die Wahrscheinlichkeit des Würfelergebnisses 2 mit 1:36 (0,028). Im Brettspiel „Die Siedler von Catan“ ist es daher z.B. sinnvoller, seine Siedlungen an Felder zu bauen, die mit Zahlen gekennzeichnet sind, die eine hohe Würfelwahrscheinlichkeit haben. Ein Verständnis dieser konstitutiven Regel ist nicht Voraussetzung, um das Spiel spielen zu können, es ist aber bei der Planung der Spielstrategie jedenfalls hilfreich.
- **Implizite Regeln *oder* Verhaltensregeln** („*Implicit Rules or Behavior Rules*“): Implizite Regeln sind die ungeschriebenen Gesetze eines jeden Spieles. So können in einer Poker-Runde beispielsweise ganz bestimmte Verhaltensregeln gelten (Coolness, Pokerface...). Sie zu missachten kann das Missfallen der Mitspielenden und im schlimmsten Fall einen Spielausschluss zur Folge haben.
- **Unterrichtete Regeln** („*Instructional Rules*“): Hierbei handelt es sich um einen Spezialfall von Regeln in Lernspielen. Gemeint sind jene Regeln (bzw. Konzepte), die durch das Spielen des Spiels erlernt werden sollen. So ist es auch bei der in dieser Arbeit entwickelten Applikation der Fall: durch das Befolgen der operationalen Regeln des Spieles lernt der spielende Paradigmen und Konzepte der Informatik, also Regeln, die im Gebiet der Informatik gelten und durch das Spiel *indirekt* unterrichtet werden.

Konflikt, Wettbewerb, Kooperation

In Spielen stehen wir immer in einer Relation zu unseren Mitspielenden oder Gegner/innen bzw. zum Spielsystem selbst. Diese Relation kann entweder Konflikt, Wettbewerb oder Kooperation sein [Kapp 2012: 31f.]:

Konflikt in Spielen ist dann vorhanden, wenn eine Herausforderung durch einen ernstzunehmenden Gegner besteht, der besiegt werden muss. Hierbei kann es sich sowohl um reale Gegenspieler/innen als auch um vom Computer gesteuerte Figuren handeln. Auch das Spielsystem selbst kann auf verschieden Arten dem Spielenden Hindernisse in den Weg legen und dessen Fortschritt hemmen. Das Ziel von Konfliktsituationen in Spielen ist es meist, den Gegner/die Gegnerin zu schwächen oder zu vernichten, jedenfalls aber direkt mit ihm/ihr zu interagieren.

Im Gegensatz dazu wird beim **Wettbewerb** nicht direkt mit den Mitspielenden interagiert. Durch verschiedene Methoden von Feedback (beispielsweise durch visuelle Darstellung der Position auf der Rennstrecke bei einem Autorennen) ist es möglich, sich mit seinen Mitspieler/innen zu messen bzw. zu vergleichen. Ziel beim Wettbewerb ist es, besser zu sein als die anderen und persönliche Bestleistungen zu erzielen.

Kooperation wiederum besteht dann, wenn man mit den Mitspielenden kollaboriert, um ein gemeinsames Gruppenziel zu erreichen. Oft ist es (z.B. in Rollenspielen) ausschließlich im Team möglich ein derartiges Ziel zu erreichen. [Raymer 2011] zufolge liegt der große Erfolg sozialer Medien und von Spielen wie FarmVille hauptsächlich am kooperativ-sozialen Aspekt. Kaum etwas sei uns so wichtig wie die Meinung unserer Freund/innen und derer, die wir respektieren. Folglich sollten gutes E-Learning und gute Lernspiele die Möglichkeit bieten, sich gegenseitig zu motivieren und sich gegenseitig Lob auszusprechen und Hilfestellungen anzubieten. Wichtige Voraussetzung dafür ist das Setzen eines gemeinsamen Zieles. Raymer erwähnt zusätzlich, dass Belohnungssysteme wesentlich effektiver sind, wenn andere Mitspielende diese Belohnungen auch sehen können.

Obgleich es viele Spiele gibt, die nur eine dieser drei sozialen Komponenten beinhalten, kombinieren viele gute Spiele alle drei Komponenten. So ist es im Online-Rollenspiel World of WarCraft z.B. möglich, alleine zu spielen und sich mit anderen zu messen (Wettbewerb), gegen Spielgegner und Monster zu kämpfen (Konflikt) und sich in sogenannten Gilden zusammenzuschließen, um gemeinsame Ziele (schneller) zu erreichen (Kooperation).

Zeit

Zeit und Zeitbeschränkungen spielen in vielen Spielen eine wichtige Rolle. Zeitbeschränkungen können z.B. in Form eines Zeitzählers oder einer Sanduhr gewissen Aktivitäten eine Maximaldauer zuweisen. Die Zeit der Erkundung und des Erforschens ist mit dem Einsetzen des Zeitzählers vorüber. Durch den dadurch erhöhten Stresslevel werden Konzentration, Fokus und konkrete Aktionen herbeigeführt, die sich darauf richten, das Spielziel zu erreichen. Es ist darauf zu achten, den Stresslevel weder zu niedrig noch zu hoch anzusetzen, um den Spieler/die Spielerin im *Flow Channel* (siehe *Flow*) zu halten. Eine Aufgabe des Einsatzes von Zeitbeschränkungen in Spielen kann sein, die Spieler/innen anzuleiten, sich ihre Zeit vernünftig einzuteilen und ihre Aufgaben zu priorisieren. Wenn man nicht unbegrenzt Zeit hat, wird es notwendig, sich zu überlegen, welche Aufgaben die wichtigsten sind und man daher zuerst erledigt. Weiters können Zeitbeschränkungen in Spielen natürlich auch dazu eingesetzt werden, die Gesamtdauer eines Spieles zu begrenzen.
Zeit kann in Spielen auch komprimiert auftreten. So vergehen in den Spielen der *Civilization*-Reihe hunderte Jahre innerhalb von Minuten. Erst diese Komprimierung ermöglicht es, dass Konsequenzen von Handlungen sofort ersichtlich werden und die Spielenden auf diese Weise Feedback erhalten. [Kapp 2012: 32f.]

Belohnungsstrukturen (Reward structures)

Es gibt vielfältige Möglichkeiten Belohnungsstrukturen in Spielen einzusetzen. Z.B. können Abzeichen (*Badges*), Punkte, eine virtuelle Währung oder Zusatzfunktionen, die sich freischalten lassen, als Belohnung fungieren. Um Belohnungen effektiv einzusetzen, sollte beachtet werden, dass sie der jeweiligen Aufgabe angemessen sind. Wenn z.B. das simple Eingeben des eigenen Namens die gleiche Belohnung nach sich zieht, wie eine Aufgabe, die einen erheblich höheren (mentalen) Aufwand erfordert, dann wird das unausweichlich zu Frustration führen. Eine gute Möglichkeit Belohnungen einzusetzen, besteht darin, Risikofreude und Aufwand zu fördern. So könnte man z.B. Hintergrundinformationen zu bestimmten Lernaktivitäten hinzufügen. Nimmt sich der/die Lernende die Zeit, diese zu studieren, so wird das entsprechend belohnt. [Kapp 2012: 33ff.] Raymer [Raymer 2011] weist darüber hinaus darauf hin, dass es wichtig sei, ein Belohnungsschema (*„Reward Schedule“*) festzulegen, welches aus folgenden drei Hauptkomponenten besteht:

- **Voraussetzung** („Prerequisite“): Was passieren muss, um die Belohnung zu erhalten.
- **Reaktion** („Response“): Die Präsentation der Belohnung.
- **Verstärker** („Reinforcer“): Eine der Voraussetzung angemessene Belohnung

Ein derartiges Belohnungsschema kann entweder in *fixen* oder *variablen* Intervallen organisiert sein, wobei es sich bei diesen Intervallen entweder um Zeitintervalle oder um eine bestimmte Anzahl von Aktionen handeln kann.

Die Belohnungen wiederum können entweder von kurzzeitiger (werden nicht aufgezeichnet) oder von dauerhafter (persistenter) Natur (werden aufgezeichnet und können später wieder abgerufen werden) sein. Während eine kurzzeitige Belohnung beispielsweise das Aufpoppen eines einfachen Satzes wie „Gut gemacht!“ oder eines kurzen Videos sein kann, manifestieren sich persistente Belohnungen beispielsweise als Abzeichen in der persönlichen Sammlung oder als virtuelles Geld auf dem Konto des oder der Spielenden. Es versteht sich von selbst, dass Abzeichen, die auch von den Mitspielenden gesehen werden können, in Normalfall einen motivierenderen Effekt haben, als solche, die nur der/die Spieler/in selbst sehen kann. Die Einbindung eines sozialen Aspekts ist also auch hier von Vorteil. Weiters sind Belohnungen, die einen tatsächlichen Einfluss auf das Spiel haben (z.B. Bonus-Levels, freigeschaltene Funktionen, virtuelles Geld für Einkäufe im spieleigenen Store...) solchen, die keinen konkreten Einfluss auf den Spielverlauf haben, vorzuziehen.

Feedback

Ein großer Vorteil von Videospielen gegenüber traditionellen Spielen und traditioneller Lehre ist, dass sie regelmäßiges, direktes und intensives Feedback geben. Die meisten guten Videospiele geben Echtzeit-Feedback über den persönlichen Fortschritt der Spieler/innen: Wie weit ist man vom aktuellen Spielziel entfernt? Wie viele Leben (wie viel Lebensenergie) hat man noch zur Verfügung? Wie viel Geld (wie viele Punkte) hat man derzeit? Wie viel Zeit verbleibt noch? An welchem Ort auf der Karte befindet man sich gerade? Welche Gegenstände hat man im Inventar? Auf welchem Stand sind die Teamkolleg/innen und Gegner/innen?

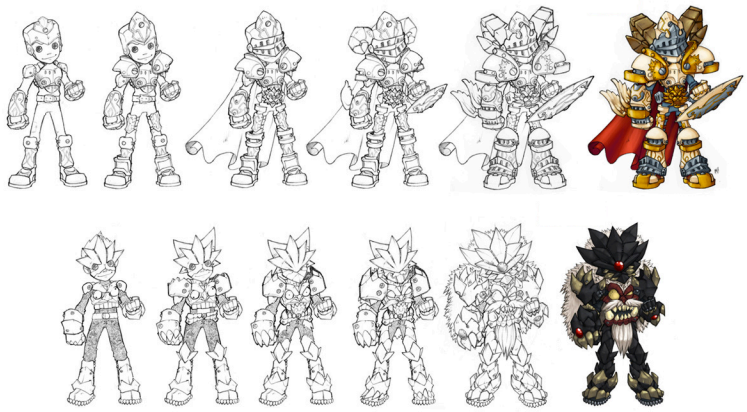
Videospiele liefern laufend informelles Feedback, das den Grad der „Korrektheit“ von Aktionen wiedergibt. Man erfährt augenblicklich, ob eine Handlung positive oder negative Konsequenzen hat bzw. ob richtig oder falsch gehandelt wurde. Dabei wird die „richtige“ Handlung aber eben nicht preisgegeben, mögliche korrekte Handlungen selbst herauszufinden obliegt dem/der Spielenden. [Kapp 2012: 35ff.]

Raymer weist darauf hin, dass in der Navigation von E-Learning und Lernspielen besonderer Wert auf regelmäßiges Feedback gelegt werden muss. Verlässt ein/e Lerner/in die Lernapplikation für einen längeren Zeitraum, so muss anhand des Feedbacks, das die Applikation gibt, auch zu einem späteren Zeitpunkt noch klar sein, an welchem Ort innerhalb der Navigation und innerhalb des Lernprozesses er oder sie sich befindet, um komplikationslos weiterarbeiten zu können, wo er

oder sie zuvor aufgehört hat. Dies kann beispielsweise geschehen, indem man Links zur Verfügung stellt, die auf essentielles Vorwissen oder wichtige Hilfsmaterialien verweisen.

In Beurteilungssituationen (Quizzes, Tests, Prüfungen) ist Feedback ebenfalls von großer Bedeutung: Es sollte immer erklärt werden, *aus welchem Grund* eine Antwort falsch ist, nicht bloß nur, *dass* sie falsch ist.

Ein besonders wichtiger Aspekt von Feedback im E-Learning ist die (meist visuelle) Darstellung des Fortschritts, die der/die Lernende bereits gemacht hat. Ein simples Beispiel hierfür ist der Fortschrittsbalken, der der/dem Lernenden auf einen Blick ersichtlich macht, wie weit sie/er noch vom Ziel entfernt ist. Hierbei ist wichtig, die Strukturierung der Lernziele (siehe *Ziele*) zu beachten und für jedes Teilziel den Fortschritt getrennt visuell abzubilden. Zum Beispiel könnte die Graphik, welche ein Modul innerhalb einer Lernsequenz darstellt, solange ausgegraut sein, bis alle Modulthemen, die wiederum mehrere Teilziele haben, erledigt wurden. Raymer schlägt darüber hinaus vor, Fortschrittsbalken nur dann anzuzeigen, wenn gerade ein Fortschritt erzielt wurde (um ein Gefühl der Belohnung zu erzeugen) und sie nicht permanent einzublenden. Jedoch sollten die Lernenden jederzeit einfach darauf zugreifen können (z.B. in Form einer Übersicht auf der obersten Ebene des Navigationsmenüs) um ihren bisherigen Fortschritt zu überprüfen.



Avatar-Upgrades als Methode der visuellen Darstellung des Lernfortschritts [Abb.Char.Upgr]

Neben Fortschrittsbalken schlägt Raymer noch eine weitere effektive Möglichkeit vor, den Fortschritt der Lernenden darzustellen: Avatar-Upgrades („Character Upgrades“). Er argumentiert, dass die Möglichkeit, selbst einen Avatar zu wählen und diesen im Laufe des Spielfortschritts mit Erweiterungen und Accessoires zu bestücken [Abb.Char.Upgr], unseren natürlichen Sammlerinstinkt ansprechen würde und demzufolge ein effektiver Motivator im E-Learning wäre. [Raymer 2011]

Levels

In Computerspielen treten üblicherweise drei verschiedene Arten von Levels auf: (1) Levels im Sinne von Missionen, die durchschritten werden müssen, bis das Spielziel erreicht wurde, (2) Levels im Sinne von Schwierigkeitsgraden, die man für ein Spiel wählen kann und (3) Levels im Sinne der Erfahrungsstufe, auf der sich ein Spieler befindet (z.B. „Level-18-Krieger“ in einem Rollenspiel). Diese drei Konzepte von Levels treten in Videospielen häufig auch simultan auf.

- **Spiele-Levels** (*Game Levels*): Spiele-Levels reduzieren den (Handlungs-)Spielraum der Spielenden und erleichtern so nicht nur die Programmierarbeit (erneut sei auf das Prinzip „divide and conquer“ hingewiesen), sondern tragen auch zur Übersicht im Spiel bei. In jedem Level ist nur eine kleine Anzahl von Teilzielen zu erfüllen. Die Story entwickelt sich in jedem Level ein kleines bisschen weiter. Weiters werden in jedem Level neue Fähigkeiten (Skills) gelernt, auf die die Folgelevels aufbauen (können). Manche Levels können auch nur zum Üben vorhandener Skills dienen. In späteren (komplexeren) Levels müssen diese Skills oft unter größerem Leistungsdruck und in Kombination angewandt werden. Darüber hinaus sind Levels natürlich auch ein Motivator. Jedes bestandene Level stellt ein Erfolgserlebnis (ein erreichtes Teilziel) dar, welches den Wunsch entstehen lässt, auch das nächste Level zu bestehen.
- **Schwierigkeitslevels** (*Playing Levels*): Um Spiele einer möglichst großen heterogenen Zielgruppe zugänglich zu machen, sollte es möglich sein, diese in unterschiedlichen Schwierigkeitsgraden (einfach - mittel - schwierig) zu spielen. Dies erhöht auch die Wahrscheinlichkeit wiederholten Spielens. Jemand, der das Spiel bereits auf einfachem Level durchgespielt hat, kann es erneut auf mittlerem und schwierigem Level probieren. Für E-Learning und Lernspiele schlägt Kapp drei andere Level-Modi vor: *Demonstration* (die Aktivität wird vorgezeigt und kann 1:1 nachgemacht werden), *Übungsmodus* (die Aktivität kann geübt werden, man erhält dazu Hilfestellungen und Tipps), *Testmodus* (die Aktivität muss ohne Hilfestellungen ausgeübt werden).
- **Erfahrungslevels** (*Player Levels*): Im Laufe eines Spiels sammeln die Spielenden Erfahrung, die oft mit sogenannten Erfahrungspunkten (XP) abgegolten wird. Eine bestimmte Anzahl an XP führt meist zum Aufstieg ins nächsthöhere Erfahrungslevel (z.B. Level-10-Magier in Rollenspielen). XP kann z.B. durch die Erfüllung von Aufgaben, Überwindung von Hindernissen oder das Besiegen von Gegnern gesammelt werden. Das persönliche Erfahrungslevel kann zur Reputation gegenüber den Mitspielern dienen, aber auch ein Gefühl des Erfolgs vermitteln, da man auf höheren Erfahrungslevels meist Privilegien (zusätzliche Fähigkeiten und Attribute) genießt, die einem auf den unteren Erfahrungslevels verwehrt bleiben. [Kapp 2012: 37ff.]

Story

Die Geschichte bzw. Story eines Spieles stellt den Bewertungsrahmen bzw. Kontext eines Spieles dar, innerhalb dessen der/die Spieler/in die Dinge bewertet. Die Story misst den Dingen Bedeutung zu und lenkt damit auch die Handlungen der Spielenden. Storys können rudimentär (Beispiel Schach: simple Story zweier kriegerischer Fraktionen, die sich in den Bezeichnungen der Spielfiguren zeigt), aber auch sehr komplex (Beispiel World of WarCraft: hochkomplexes Fantasie-Universum mit einer sehr großen Anzahl möglicher Story-Verläufe) sein. Schon der Name eines Spieles kann eine Geschichte in der Vorstellung der Spielenden entstehen lassen (Beispiel Space Invaders: simple Pixelgrafiken).

Gute Lernspiele vermischen die Story, die auf den von den Lernenden durchgeführten Aufgaben (Tasks) basiert, mit interaktiven Spielelementen, um dem Spieler dabei zu helfen, die erwünschten Verhaltensweisen, Aktionen und Denkmuster zu erlernen. Im Gegensatz zu Auflistungen von Anweisungen und Prozeduren sind Geschichten einfacher zu merken, was daran liegen könnte, dass sie eine emotionale Dimension hinzufügen und ein mentales Modell bereitstellen, das die Einordnung und Bewertung von Handlungen und Ereignissen innerhalb des Spieles möglich macht. Kapp erwähnt außerdem, dass gute Storys aus folgenden Elementen bestehen: Charaktere, Handlung (*Plot*), Spannung (*Tension*) und Auflösung (*Resolution*). Es sind diese vier Elemente guter Geschichten, die den/die Spieler/in schrittweise durch das Spiel bis zur Erfüllung des Spielziels geleiten. [Kapp 2012: 41f.]

Ästhetik

Gerade in Lehr- und Lernspielen wird im Gegensatz zu Unterhaltungsspielen oft viel zu wenig Augenmerk auf deren Ästhetik gelegt. Schönheit, Kunst, Detailverliebtheit und Konsistenz der graphischen Darstellung sind essentielle Aspekte dieser Ästhetik. Nun kann eine anspruchsvolle visuelle Darstellung ein schlechtes Spiel zwar nicht zu einem guten Spiel machen, sie kann allerdings ein gutes Spiel zu einem großartigen machen. Erst die anspruchsvolle Darstellung ermöglicht oft das Eintauchen in die Story bzw. Fantasiewelt des Spiels. Auch klassische Brettspiele wie Schach können z.B. durch kunstvoll gestaltete Figuren einen ganz besonderen Reiz bekommen. Es sei an dieser Stelle angemerkt, dass Ästhetik nicht mit Realismus gleichgesetzt werden darf. Grafiken in Spielen übermitteln (oft auf möglichst simple Art und Weise) visuelle Informationen und Symbolik. [Kapp 2012: 46ff.]

Wiederholung (Replay)

Die Möglichkeit zu scheitern und Aktivitäten wiederholen zu können sollte beim Gamedesign nicht außer Acht gelassen werden. Erst durch die Möglichkeit des Scheiterns können die Spielenden funktionierende von nicht funktionierenden Strategien unterscheiden und somit dazulernen. Das Wissen jede Aktivität wieder von vorne beginnen zu können, ermöglicht erst das Aufkommen von Neugier, Forschungsdrang und explorativem Lernen. Es fügt sozusagen einen Freiheitsgrad hinzu, der in der klassischen Lehre, in der Scheitern unerwünscht ist, meist fehlt. Spiele sollten jedoch Maßnahmen ergreifen, um zu häufiges Scheitern an der selben Aufgabe zu vermeiden. So kann die Spielumgebung den Spielenden bei zu häufigem Scheitern an der selben Aktivität beispielsweise Hilfestellungen anbieten, den Schwierigkeitsgrad senken oder eine alternative Aufgabe anbieten. Die Möglichkeit nicht geschaffte Aktivitäten zu wiederholen, kann auch einen nicht zu unterschätzenden Motivationseffekt haben: Das Erfolgserlebnis, eine Aufgabe geschafft zu haben, an der man bereits mehrmals gescheitert ist, ist weitaus größer, als eine Aufgabe gleich beim ersten Mal zu schaffen. [Kapp 2012: 48f.]

Kategorisierung

In [Deterding 2011|2] nehmen Deterding, Dixon, Khaled und Nacke die folgende Kategorisierung von Gamedesign-Elementen vor:

Kategorie	Beschreibung	Beispiele
Spiel-Interface-Designpatterns	Übliche und erfolgreiche Interaktionsdesign-Komponenten und Design-Lösungen für bekannte Probleme innerhalb eines Kontexts.	Badges, Levels, Leaderboard
Gamedesign-Patterns und Spielmechanik	Häufig wiederkehrende Teile des Designs eines Spiels, die den Spielablauf betreffen.	Zeitbeschränkung, beschränkte Ressourcen, Runden
Gamedesign-Prinzipien und Heuristiken	Evaluative Guidelines zur Behandlung von Designproblemen oder zur Analyse bestehender Designvorschläge	Kontinuierlicher Spielverlauf, klare Ziele, verschiedene Spiel-Arten
Spiel-Modelle	Konzeptionelle Modelle der Komponenten von Spielen oder des Spiel-Erlebens (*game experience*)	Fantasie, Story, Neugier, Herausforderung
Gamedesign-Methoden	Gamedesign-spezifische Verfahren und Prozesse	Spieltestung, spielzentriertes Design, wertbewusstes Spieldesign

[Tab.GDE] Kategorisierung von Gamedesign-Elementen

Designkonzept

Design Document

Der folgende Abschnitt stellt das *Design Document* des Gamification-Prozesses für die Tasks des Biber-Wettbewerbs 2013 und 2012 dar. Das Design Document bildet eine wichtige Grundlage für den Entwicklungsprozess. Es stellt einen Sammelpunkt für Gedanken, Ideen und Ansätze der Spielentwicklung dar, jedoch keine exakte Schritt-für-Schritt-Anleitung des Entwicklungsprozesses. [vgl. Kapp 2012: 205]

Kurzdarstellung des Designkonzepts

Auf Grundlage der Tasks des Bebras-Contests 2013 und 2012 wird eine mobile Applikation in Form eines Spieles entworfen, dessen Arbeitstitel „Bebras" ist. Ziel dieses Spiels soll es sein, möglichst viele (im Idealfall alle), der im Spiel enthaltenen Tasks erfolgreich abzuschließen. Um dieses Ziel zu erreichen, kann zu Beginn des Spieles aus **vier Levels** (Benjamin, Meteor, Junior, Senior) gewählt werden. Diese Levels entsprechen den Altersgruppen, in denen der Bebras-Contest organisiert ist. Ein Spiel besteht aus **drei Modulen** (jeweils ein leichtes, mittleres und schweres Modul) mit **jeweils sechs Tasks** und separat dazu **drei Bonus-Tasks**. Aus welchen Tasks die Module bestehen hängt vom gewählten Level (Altersgruppe) des Nutzers ab. Der oder die Spielende wird durch einen **Avatar** in Form eines farbigen Bibers dargestellt.

Nach Abschluss eines Tasks erhält man eine bestimmte Anzahl an Erfahrungspunkten (**XP**) und Bebras-Coins (**BC**). Bei falsch beantworteten Fragen verliert man Erfahrungspunkte. Mit Bebras-Coins können zwei Dinge gekauft werden: Hilfestellungen (**Hints**) und **Bonus-Tasks**.

Weiters besteht die Möglichkeit, sich vor einem Task dazu zu entscheiden, einen **Timer** (eine Zeitbeschränkung) zu setzen. Schafft man den Task in der vorgegebenen Zeit, so erhält man eine zusätzliche (von der Schwierigkeit der Aufgabe abhängige) Anzahl an Erfahrungspunkten (XP). Pro Modul lässt sich ein Timer setzen.

Der Lernfortschritt der Spieler/innen ist anhand einer **graphischen Darstellung der Tasks** ersichtlich (bestandene Tasks und Module werden graphisch hervorgehoben). Darüber hinaus können die Spieler/innen ihren XP-Punktestand auf einem **Leaderboard** eintragen und miteinander vergleichen.

Erwartetes Resultat

Erwartet wird der Prototyp einer mobilen Applikation, der alle soeben in der Kurzdarstellung des Designkonzepts genannten Gamedesign-Elemente (Levels, Module, Tasks, Avatare, XP, BC, Hints, Timer, graphische Darstellung des Lernfortschritts, Leaderboard) und die Bebras-Tasks derart implementiert, dass die Lernmotivation der Nutzer gefördert wird (ob bzw. wie erfolgreich dies gelungen ist, wird im letzten Teil dieser Arbeit evaluiert).

Hierzu sollen sowohl Multiple Choice-Tasks als auch Tasks mit Input implementiert werden. Da die Implementation der *interaktiven* Tasks des Bebras-Contests 2013 den zeitlichen Rahmen dieser Arbeit sprengen würde, werden an ihrer Stelle Tasks aus dem Katalog des Bebras-Contests 2012 implementiert.

Eine weitere Anforderung an den Prototypen besteht in der leichten *Erweiterbarkeit* mit den Tasks anderer Kataloge der Bebras-Wettbewerbe vergangener und zukünftiger Jahrgänge.

Lernziele

Die Lernziele der Applikation ergeben sich aus den Kategorien, in welche sich die Bebras-Tasks einordnen lassen (siehe Abschnitt *Bebras-Tasks: Klassifikation und Kriterien*).

Nachdem Abschluss aller drei Module des Spiels kennen die Spieler/innen...

- ...einige Konzepte symbolischer, numerischer und visueller Informationsrepräsentation, Kodierung und Verschlüsselung (**INF**)
- ...algorithmische Denkweisen und einfache Programmieraspekte (**ALG**)
- ...Beispiele der Anwendung von Computersystemen (**USE**)
- ...verschiedene Datenstrukturen wie beispielsweise Graphen (**STRUC**)
- ...Logik-Puzzles und logische Spiele und können diese lösen (**PUZ**)
- ...soziale, ethische, kulturelle und rechtliche Grundlagen von Informations- und Kommunikationstechnologien (**SOC**)

Hinweis: Abhängig davon aus welchen Tasks sich ein Modul zusammensetzt, kann es vorkommen, dass ein Modul nicht alle hier aufgezählten Lernziele abdeckt.

Tasks

Das Spiel „Bebras“ enthält Tasks aus den Katalogen der Bebras-Contests 2012 und 2013, deren genauer Wortlaut in **[Biber-Aufgaben 2012]** und **[Biber-Aufgaben 2013]** nachgelesen werden kann. Die folgende Tabelle strukturiert die Tasks in ihre jeweiligen Kategorien, Schwierigkeitsgrade sowie das in ihnen enthaltene Konzept. Die Schwierigkeitsgrade wurden teilweise geringfügig abgeändert, sodass die Modulstruktur, wie unten angegeben, eingehalten werden konnte.

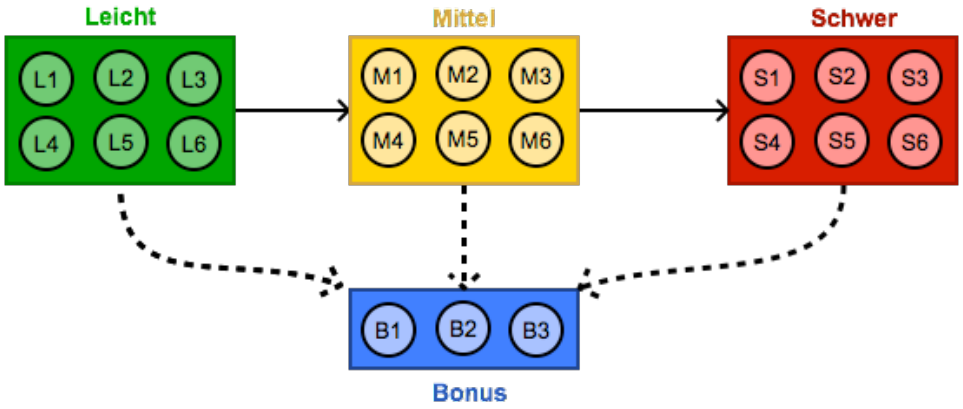
					Schwierigkeitsgrad			
ID	**Name**	**Katalog**	**Konzept**	**Kategorie**	**Ben**	**Met**	**Jun**	**Sen**
1	**Eismaschine**	2013	Automat	ALG	L	-	-	-
2	**Im Wald**	2013	Graph	STRUC	L	-	-	-
3	**Flipflop**	2013	Bit	INF	M	L	-	-
4	**Bibers Geheimcode**	2012	Kryptographie / Verschlüsselung	INF	L	L	-	-
5	**Datenübertragung**	2012	Code, Medium, Sender, Empfänger	INF	L	-	-	-
6	**Husch in den Busch**	2013	Einfaches Programm / Algorithmus	ALG	L	-	-	-
7	**Stempelmaschine**	2012	Änderung von Programmabläufen	ALG	M	L	-	-
8	**Schulausflug**	2013	Sinnhafte Kommunikation	SOC	L	-	-	-
9	**Signalfeuer**	2013	Graph / Kürzester Pfad / Breitensuche	STRUC	M	M	L	L
10	**Höchster Baum**	2013	Graph / Lokale Suche	STRUC	S	M	-	-
11	**Magische Tunnel**	2013	Funktionen	ALG	M	L	-	-
12	**Blumen pflanzen**	2012	Physische Merkmale von Maschinen, die Algorithmen ausführen	ALG	M	M	-	-
13	**Städte**	2013	Relationale Datenbank	STRUC	M	-	-	-
14	**Drehzeug**	2013	Binärer Baum	STRUC	S	M	M	L
15	**Erreicht er sein Ziel?**	2012	Algorithmus ausführen	ALG	S	-	-	L
16	**1-2-3 Kuchen**	2013	Stapelspeicher / Stack	STRUC	S	-	-	-
17	**Höhlenforschung**	2013	Graph	STRUC	S	-	-	-
18	**Getränkeautomat**	2013	Codierung / Endliche Automaten	INF / STRUC	S	-	-	-
19	**Passende Halskette**	2013	Graph / Kürzester Pfad	STRUC	-	L	L	-
20	**Was gibt‘s Neues**	2013	Netzwerk / Gerichteter Graph	STRUC	-	L	-	-
21	**Flughafen**	2013	Algorithmen / Josephus-Problem	ALG	-	M	L	-
22	**Ruderturnier**	2013	Binärsystem	INF	-	M	L	-
23	**Visitenkarten**	2012	Binärsystem	PUZ	-	S	L	L
24	**Flussdiagramm**	2013	Flussdiagramm	ALG	-	S	S	-

					Schwierigkeitsgrad			
ID	**Name**	**Katalog**	**Konzept**	**Kategorie**	**Ben**	**Met**	**Jun**	**Sen**
25	**Hobbiber**	2013	Graph / Optimierungsproblem	STRUC / ALG	-	S	M	-
26	**Flüsse und Dämme**	2012	Optimierung in Netzwerken	STRUC	-	S	L	-
27	**Dreiecksverschleierung**	2013	Verschlüsselung	INF	-	S	-	-
28	**Serielle Übertragung**	2013	Zeichenkodierung	INF	-	S	M	L
29	**Punktemuster**	2013	Formale Sprachen	ALG	-	-	M	-
30	**Halbetzen**	2012	Rekursion / Divide and conquer	ALG	-	-	M	L
31	**Zufallsbilder**	2013	Zufallsalgorithmen	ALG	-	-	M	M
32	**Fluss-Prüfung**	2013	Gerichteter Graph / Maximaler Schnitt	STRUC	-	-	S	M
33	**Freunde besuchen**	2013	Top-Down-Analyse / Modulo-Berechnung	ALG	-	-	S	M
34	**Gläser**	2012	Logische Probleme lösen	PUZ	-	-	S	S
35	**Niemals links**	2013	Minimaler Pfad	ALG	-	-	S	M
36	**Von A nach C**	2013	Einfaches Programm / Algorithmus	ALG	-	-	S	M
37	**Effizient kochen**	2013	Job Scheduling	ALG	-	-	-	M
38	**Bunte Perlenkette**	2013	Syntaxdiagramm	ALG	-	-	-	S
39	**Hotelschlüssel**	2013	Fehlertoleranz	INF	-	-	-	S
40	**Magische Maschine**	2013	Petri-Netz	STRUC	-	-	-	S
41	**Heimweg**	2013	Unvollkommenes Wissen formalisieren	PUZ	-	-	-	S
42	**RAID**	2013	RAID-Technologie	USE	-	-	-	S

[Tab.SBT] Struktur der Bebras-Tasks

Module

Module bestehen aus jeweils sechs Bebras-Tasks. Ein Spiel enthält drei Module: ein leichtes, ein mittleres und ein schweres Modul. Um die Module „Mittel“ und „Schwer“ starten zu können, muss das vorhergehende leichtere Modul jeweils vollständig abgeschlossen sein. Das Bonus-Modul enthält drei Bonus-Tasks, die sich mit Bebras-Coins freischalten lassen. Sie können von jedem Modul aus gestartet werden, vorausgesetzt man hat die notwendigen finanziellen Ressourcen.



[Abb.Module] Diagramm der Modulstruktur

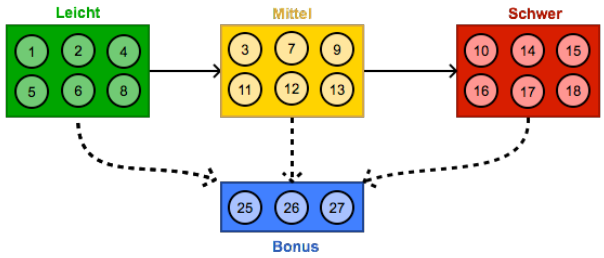
Bonus-Tasks

Bonus-Tasks sind Tasks, die nicht in den drei Standard-Modulen enthalten sind, sondern mit Bebras-Coins (BC) freigeschaltet werden können. Die Erfüllung von Bonus-Tasks bringt zusätzliche Erfahrungspunkte (XP). In einem Spiel können bis zu drei Bonus-Tasks freigeschaltet werden. Ein Bonus-Task kostet **12 BC**.

Levels

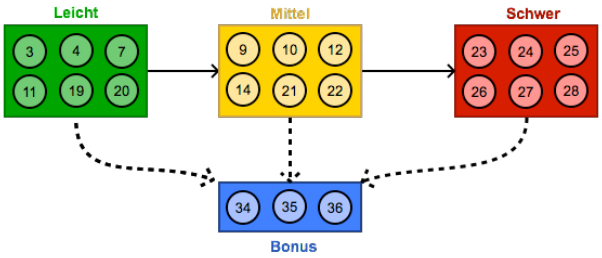
Das Spiel „Bebras“ kann auf vier verschiedenen Levels gespielt werden. Aus welchen Tasks sich die jeweiligen Module zusammensetzen, hängt vom gewählten Level ab. Die Schwierigkeitsgrade der Tasks wurden größtenteils aus dem Katalog übernommen, mussten in einigen wenigen Einzelfällen jedoch adaptiert werden, um sicherzustellen, dass jedes Modul (mit Ausnahme der Bonus-Module) aus genau sechs Tasks besteht. Die folgenden Diagramme geben die Task-Zusammensetzung der jeweiligen Module auf jedem Level an. Die Zahlen entsprechen der Task-ID aus [Tab.SBT]:

Level „Benjamin“



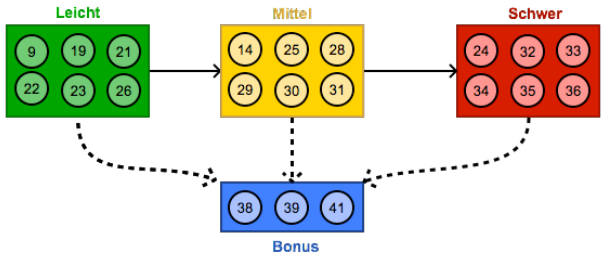
Modulzusammensetzung auf dem Level „Benjamin“ [Abb.Module.Ben]

Level „Meteor“



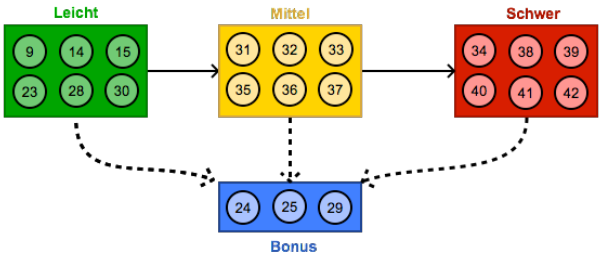
Modulzusammensetzung auf dem Level „Meteor“ [Abb.Module.Met]

Level „Junior“



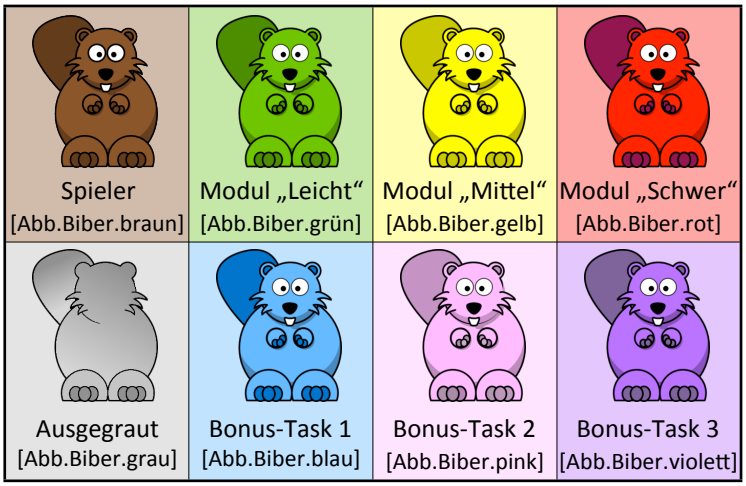
Modulzusammensetzung auf dem Level „Junior“ [Abb.Module.Jun]

Level „Senior“



Modulzusammensetzung auf dem Level „Senior“ [Abb.Module.Sen]

Avatare



Der Spieler wird als Avatar in Form eines braunen Bibers dargestellt. Mit dem Abschluss jedes Moduls können weitere Avatare (grün, gelb und rot) gesammelt werden. Die Avatare in blau, pink und violett lassen sich durch das Abschließen von Bonus-Tasks sammeln. Noch nicht gesammelte Avatare werden im Spiel ausgegraut dargestellt. Ziel des Spieles ist es, so viele Avatare wie möglich zu sammeln.

Erfahrungspunkte (XP)

Erfahrungspunkte (XP) können durch das Abschließen von Tasks gesammelt werden. Sie sind abhängig von der Schwierigkeit der Aufgabe. Die meisten Erfahrungspunkte lassen sich mit Bonus-Tasks erzielen. Für richtige Antworten beim ersten Versuch erhält man die volle Punktezahl an XP gutgeschrieben, beim zweiten Versuch erhält man noch ein Drittel der XP-Punktezahl und für falsche Antworten werden XP abgezogen. Durch dieses Schema soll verhindert werden, dass bei Multiple Choice-Fragen einfach nur geraten wird.

Folgendes Schema gibt die Verteilung von Erfahrungspunkten wieder:

XP-Schema	Richtig (1. Versuch)	Richtig (2. Versuch)	Falsche Antwort
Leicht	+6 XP	+2 XP	-2 XP
Mittel	+9 XP	+3 XP	-3 XP
Schwer	+12 XP	+4 XP	-4 XP
Bonus	+15 XP	+5 XP	-5 XP

[Tab.XP.Schema]

Hints

Hints sind Hinweise, die bei der Lösung von Tasks helfen sollen. Pro Task gibt es genau einen Hint, der durch Bezahlung eines Bebras-Coins (1 BC) freigeschaltet werden kann. Ist ein Hint einmal freigeschaltet, so bleibt er das auch dann, wenn der Task bzw. die App geschlossen und neu geöffnet wird.

Bebras-Coins (BC)

Bebras-Coins sind die virtuelle Währung des Spiels „Bebras“. Mit ihnen lassen sich Hints (Hinweise bzw. Tipps zur Lösung der Tasks) und Bonus-Tasks kaufen. Die Anzahl der Bebras-Coins, die man für eine abgeschlossene Aufgabe erhält, ist abhängig von der Anzahl der Versuche, die man für die Aufgabe benötigt hat, nicht aber vom Schwierigkeitsgrad der Aufgabe. Das folgende Schema gibt die Verteilung von Bebras-Coins im Spiel wieder:

BC-Schema

Startkapital	+ 3 BC
Bonus-Task freischalten	-12 BC
Hint freischalten	- 1 BC

	1. Versuch	**2. Versuch**	**>= 3. Versuch**
Tasks	+ 3 BC	+ 1 BC	+ 0 BC
Bonus-Tasks	+ 3 BC	+ 1 BC	+ 0 BC

[Tab.BC.Schema]

Im BestCase-Szenario (alle Tasks wurden beim ersten Versuch geschafft) kann der Spieler oder die Spielerin also insgesamt die folgenden Einnahmen erzielen:

```
                    Max(Einnahmen) =
Startkapital + 21 Tasks (inkl. Bonus-Tasks) beim ersten Versuch =
                  3 BC + (21 * 3 BC) = 66 BC
```

Das Maximum an möglichen Ausgaben wiederum beträgt:

```
                    Max(Ausgaben) =
21 Hints (inkl. Hints von Bonus-Tasks) + 3 Bonus-Tasks =
                  21 BC + (3* 12 BC) = 57 BC
```

Um sich **alle** Hints **und alle** Bonus-Tasks kaufen zu können darf also höchstens ein Verlust von 9 BC gemacht werden (beispielsweise indem 3 Tasks erst beim dritten oder vierten Versuch richtig beantwortet wurden).

Timer

Ein Timer ist eine Zeitbeschränkung, die man vor dem Start eines Tasks aktivieren kann. Schafft man den Task innerhalb der vorgegebenen Zeit, so erhält man die doppelte Anzahl an XP für diesen Task. Für Tasks des Moduls „Leicht“ wird der Timer auf 120 Sekunden gesetzt, im Modul „Mittel“ auf 135 Sekunden und im Modul „Schwer“ auf 150 Sekunden. Pro Modul kann man genau einen Timer setzen.

Overview (inkl. Avatar-Sammlung)

Der Overview ist eine graphische Darstellung des Lernfortschritts, die alle Module sowie die Spieler-Information (Name des Spielers, XP- und BC-Stand) visuell repräsentiert. Jedem Modul ist ein farbiger Avatar zugeordnet, der freigeschaltet wird sobald alle Tasks des Moduls positiv absolviert wurden. Ein im Overview dargestelltes Modul hat jeweils einen von drei Zuständen.

(1) Modul gesperrt (dargestellt durch ein Schloss)
(2) Modul freigeschalten (dargestellt durch einen grauen Avatar)
(3) Modul abgeschlossen (dargestellt durch einen farbigen Avatar)

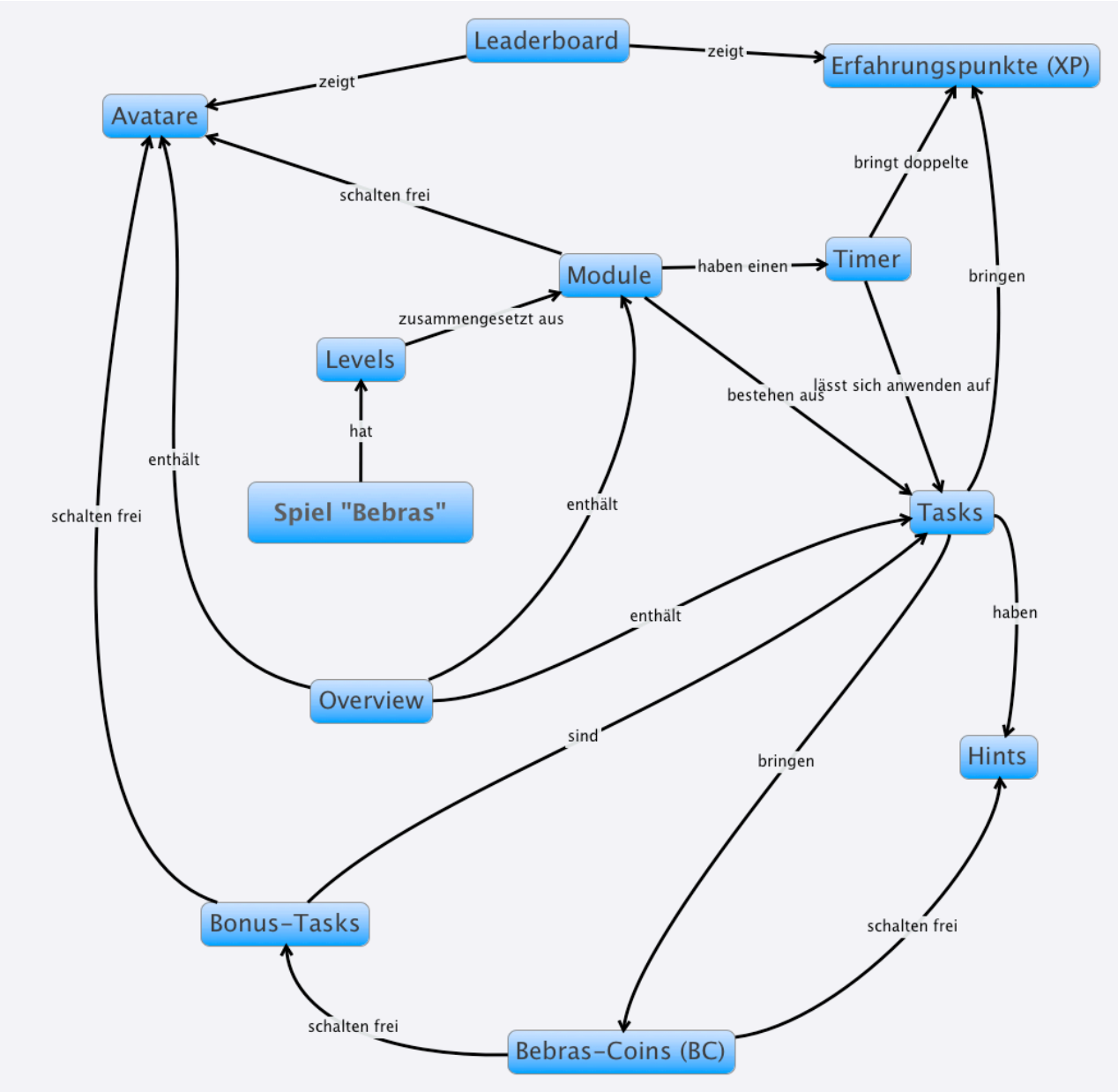
Bonus-Tasks werden auf dieselbe Art und Weise dargestellt, jedoch werden diese durch Bezahlung von 12 BC freigeschaltet.

Leaderboard (Highscores)

Das Leaderboard stellt die gesammelten Erfahrungspunkte (XP) einzelner Spielender dar. Es dient zum Vergleich des Lernfortschritts mit anderen Spielenden sowie mit den Spielständen vergangener Spiele und soll als zusätzliche Motivation wirken.

Concept Map

Abschließend seien die einzelnen Gamedesign-Elemente des Spiels „Bebras“ in Form einer Concept Map dargestellt. Diese gibt das Zusammenspiel bzw. die Relationen der einzelnen Gamedesign-Elemente zueinander wieder:



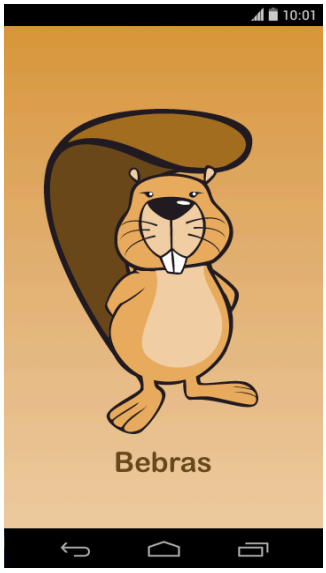
Zusammenspiel der einzelnen Gamedesign-Elemente, dargestellt als Concept Map

[Abb.GDE.ConceptMap]

User Interface Design

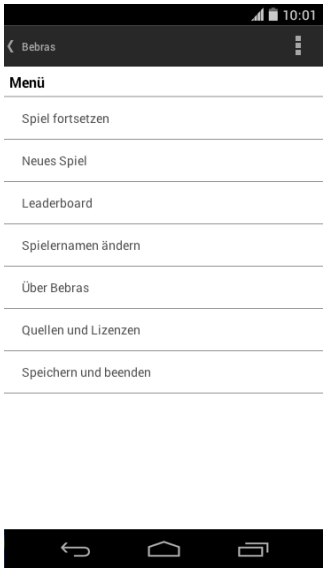
Mock-Ups

Das User Interface der App „Bebras“ unterteilt sich in mehrere Bereiche (Screens). Für jeden dieser Screens wurde ein Mock-Up erstellt, welches **als Vorlage und Referenz** für die Implementierung dienen soll. Hierbei ist anzumerken, dass die Mock-Ups vor allem als Hilfestelleung für die Implementierung dienen, jedoch keine verbindlichen Layout-Vorgaben darstellen.



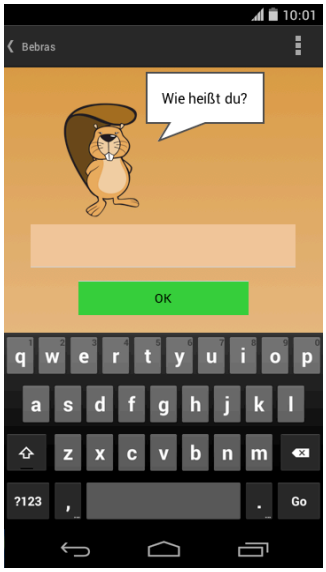
Startbildschirm

[Abb.Mockup.Start]



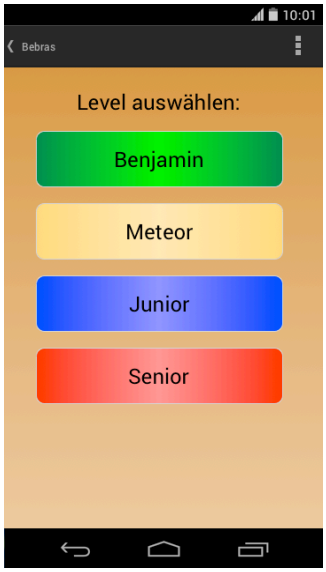
Hauptmenü

[Abb.Mockup.Menu]



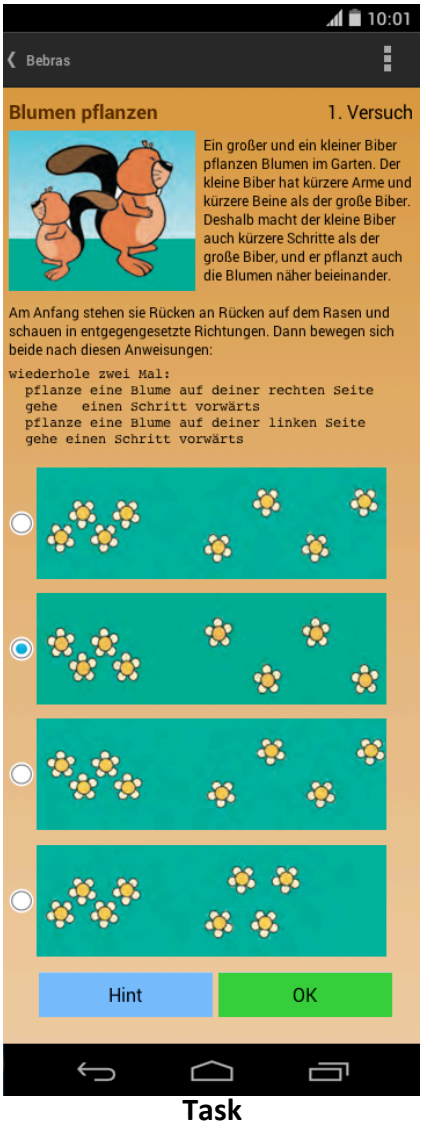
Eingabe des Spielernamens

[Abb.Mockup.Name]



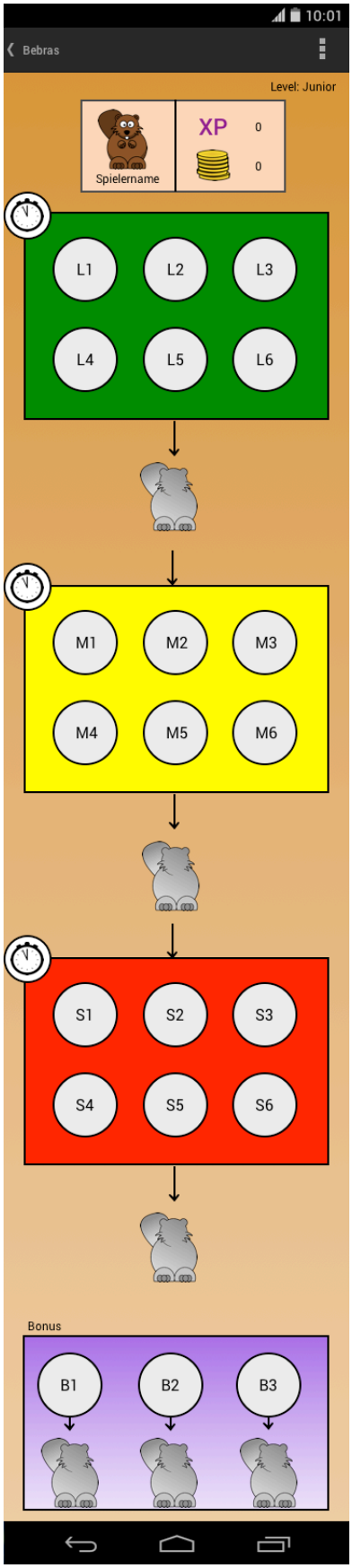
Level-Auswahl

[Abb.Mockup.Level]

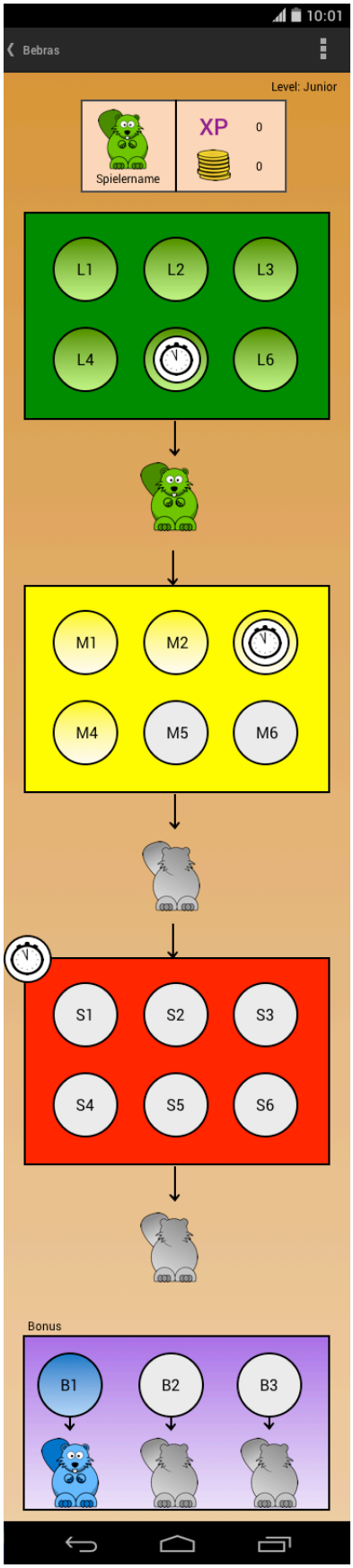


Task

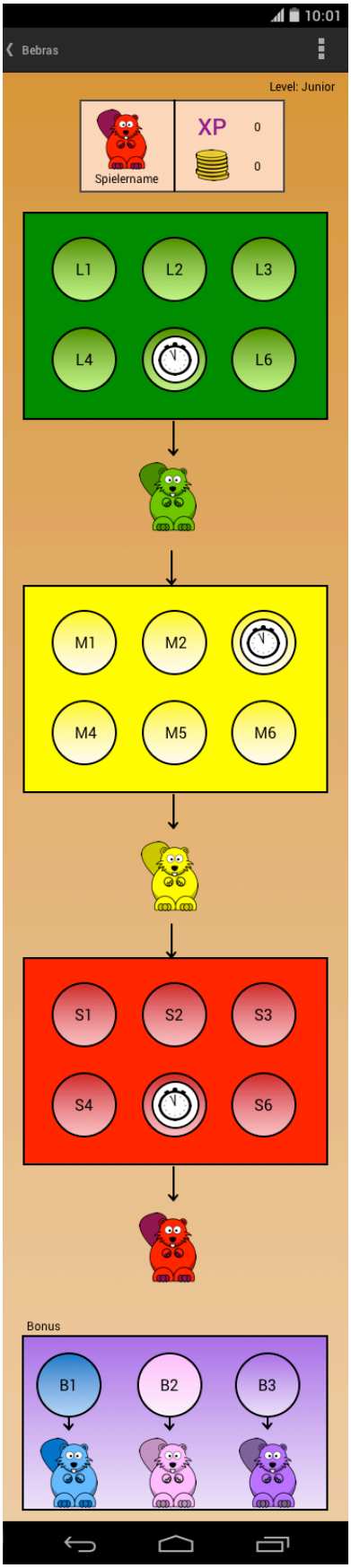
[Abb.Mockup.Task]



Overview (neues Spiel)
[Abb.Mockup.Overview.empty]

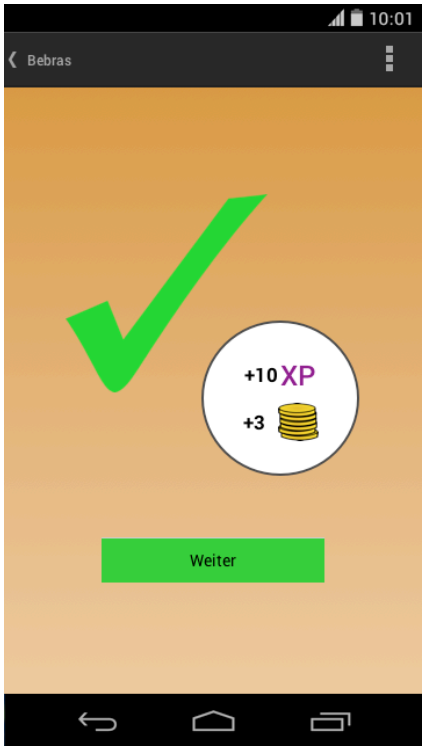


Overview (begonnenes Spiel)
[Abb.Mockup.Overview.half]



Overview (fertiges Spiel)
[Abb.Mockup.Overview.full]

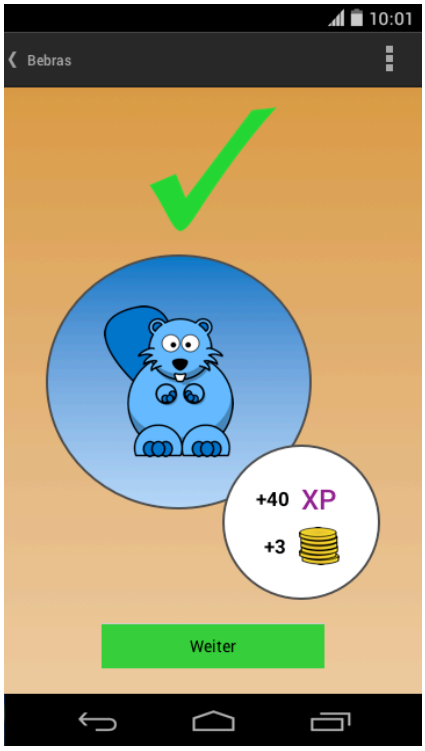
Task falsch
[Abb.Mockup.Task.Fail]



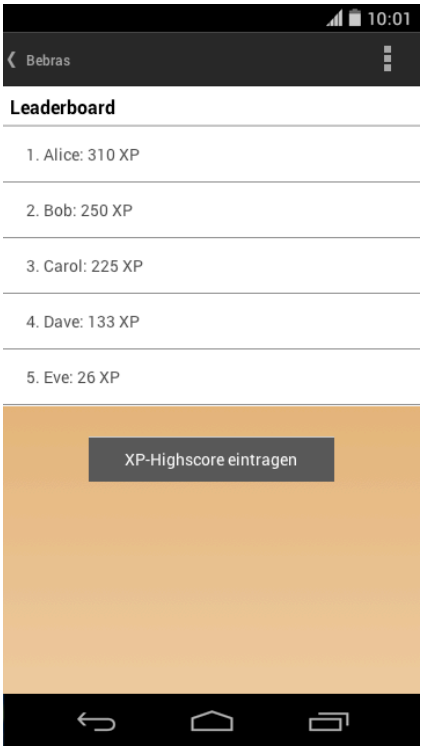
Task richtig
[Abb.Mockup.Task.Success]

Hint
[Abb.Mockup.Hint]

Modul abgeschlossen
[Abb.Mockup.Modul.Success]



Bonus-Task richtig
[Abb.Mockup.Bonustask.Success]



Leaderboard
[Abb.Mockup.Leaderboard]

Beschreibung der Mock-Ups

Startbildschirm

Der Startbildschirm wird direkt nach dem Öffnen der App für wenige Sekunden angezeigt. Er zeigt das Bebras-Maskottchen und den Namen der App. [Abb.Mockup.Start]

Hauptmenü

Das Hauptmenü [Abb.Mockup.Menu] wird im Anschluss an den Startbildschirm angezeigt. Das Hauptmenü bietet die folgenden Optionen:

- **Spiel fortsetzen**

  Setzt das zuletzt geöffnete Spiel an der Stelle fort, an der man aufgehört hat.

- **Neues Spiel**

  Löscht den Speicherstand des zuletzt geöffneten Spieles (falls vorhanden) und lädt ein neues Spiel. Level (Altersgruppe) und Name können in diesem Fall neu gewählt werden.

- **Leaderboard**

  Öffnet den Leaderboard-Screen [Abb.Mockup.Leaderboard].

- **Spielernamen ändern**

  Öffnet den Screen zum Ändern des Spielernamens [Abb.Mockup.Name].

- **Über Bebras**

  Zeigt allgemeine Informationen („About“) über die App und Information über den Autor an.

- **Quellen und Lizenzen**

  Zeigt die Quellen der benutzten Clip-Arts, Software und deren Lizenzen an.

- **Speichern und beenden**

  Speichert den aktuellen Spielstand und beendet die App.

Level-Auswahl

Beim Start eines neuen Spiels kann man aus den vier Levels Benjamin, Meteor, Junior und Senior wählen. Das gewählte Level legt fest, aus welchen Tasks sich die Module des Spiels zusammensetzen. [Abb.Mockup.Level]

Eingabe des Spielernamens

Dieser Screen erlaubt die Eingabe eines Spielernamens bzw. dessen Änderung falls bereits ein Name vorhanden ist. Der Spielername wird im Overview angezeigt. Weiters wird er für Einträge im Leaderboard benötigt. [Abb.Mockup.Name]

Overview

Der Overview ist die zentrale Anlaufstelle des Spiels. [Abb.Mockup.Overview.*] Er ermöglicht den Zugriff auf die Tasks und gibt einen Überblick über alle spielrelevanten Informationen:

Nutzerfenster

- Spielername
- aktuell ausgewählter Avatar
- XP-Punktestand
- BC-Kontostand

Modul-Bereich

- Tasks (Tasks können nur dann geöffnet werden, wenn sie sich in einem bereits freigeschalteten Modul befinden. Noch nicht abgeschlossene Tasks werden ausgegraut dargestellt.)
- Timer (pro Modul lässt sich ein Timer auf einen der Tasks dieses Moduls anwenden)
- Module (werden durch Abschluss des Vorgängermoduls freigeschalten)
- Bonus-Tasks (werden mit Bebras-Coins freigeschalten)
- Avatare (Noch nicht gesammelte Avatare sind ausgegraut und können nicht angewählt werden. Anwählen eines bereits gesammelten Avatars hat zur Folge, dass dieser im Nutzerfenster angezeigt wird.)

Task

Der Task-Screen zeigt den Namen des aktuellen Tasks, die Task-Beschreibung (inkl. Bilder), die Anzahl der Versuche, welche man bisher für den Task benötigt hat, die Antwort-Möglichkeiten (bzw. bei interaktiven Tasks einen entsprechenden interaktiven Bereich), den Hint-Button und den OK-Button. [Abb.Mockup.Task]

Eine Antwort wird erst nach Antippen des OK-Buttons gewertet (davor kann zwischen verschiedenen Antwortmöglichkeiten gewechselt werden bzw. können davor beliebige Manipulationen im interaktiven Bereich durchgeführt werden).

Hint

Ein Hint ist ein kurzer Text, der einen Hinweis auf die Lösung des Tasks gibt und kostet einen Bebras-Coin. Manche Hints schließen auch eine der Antwortmöglichkeiten aus. Er wird als Dialog angezeigt und kann durch Antippen des „Gelesen“-Buttons geschlossen werden. [Abb.Mockup.Hint]

Task erfolgreich

Mit einem großen grünen Häkchen wird angezeigt, dass man den Task richtig beantwortet hat. Darüberhinaus wird angezeigt, wieviele Erfahrungspunkte (XP) und Bebras-Coins (BC) man durch die richtige Beantwortung dazugewonnen hat. Antippen des „Weiter"-Buttons lässt den Spieler/die Spielerin zum Overview zurückkehren. [Abb.Mockup.Task.Success]

Bonus-Task erfolgreich

Zusätzlich zu den Erfahrungspunkten (XP) und Bebras-Coins (BC) wird bei einem Bonus-Task noch der gesammelte Avatar angezeigt, den man durch dessen Abschluss gesammelt hat.
[Abb.Mockup.Bonustask.Success]

Task nicht erfolgreich

Ein großes rotes „X" zeigt an, dass der Task falsch beantwortet wurde. Weiters wird angezeigt, wie viele Versuche man für den Task schon benötigt hat. Durch Antippen von „Weiter" kehrt der Spieler/die Spielerin zum Overview zurück. [Abb.Mockup.Task.Fail]

Modul abgeschlossen

Sobald der letzte Task eines Moduls richtig beantwortet wurde, wird im Anschluss an den „Task erfolgreich"-Screen dieses Tasks der „Modul abgeschlossen"-Screen des jeweiligen Moduls angezeigt. Es wird der Avatar in der Farbe des entsprechenden Moduls angezeigt, was bedeutet, dass dieser nun freigeschalten wurde. Durch Antippen von „Weiter" kehrt der Spieler/die Spielerin zum Overview zurück. [Abb.Mockup.Modul.Success]

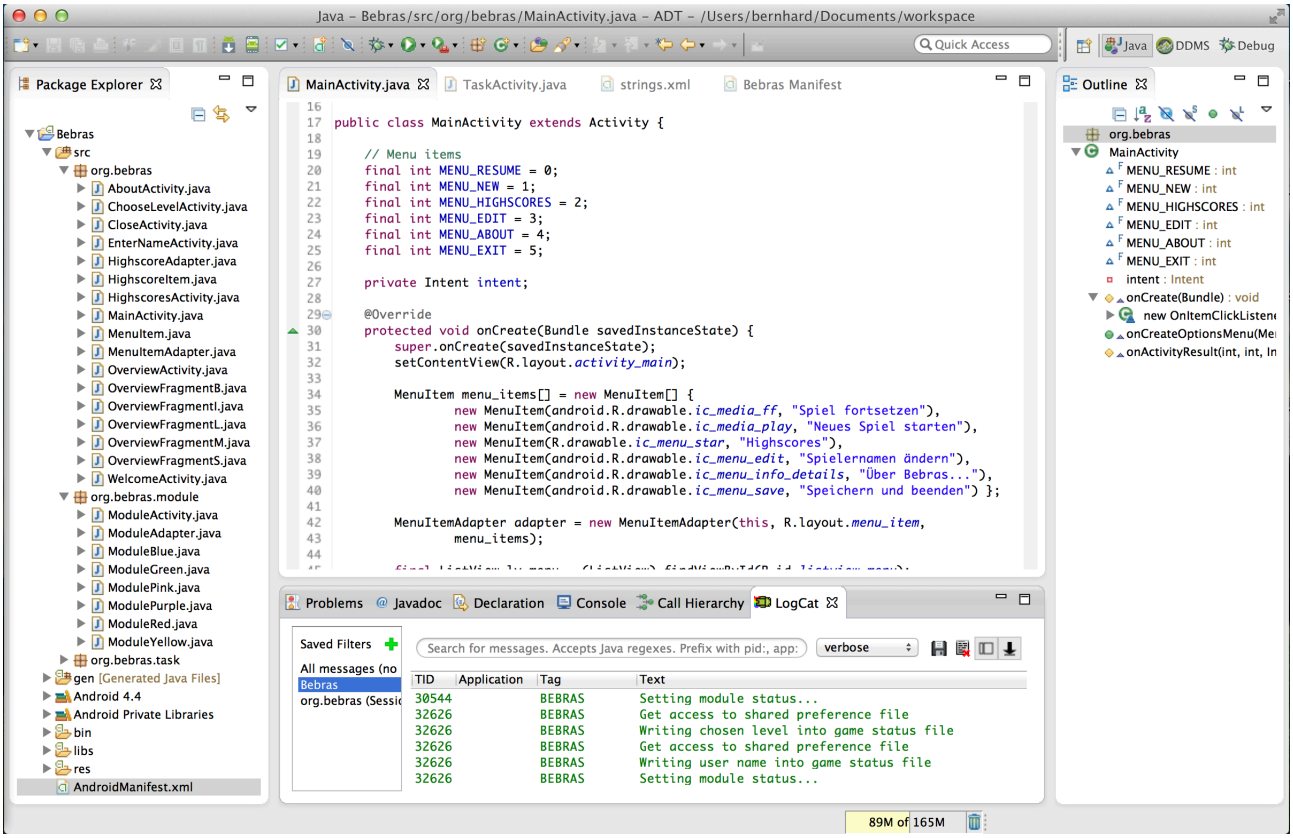
Leaderboard

Das Leaderboard zeigt die XP-Highscores der Spieler/innen an. Durch Antippen des Buttons „XP-Highscore eintragen" kann der XP-Punktestand des aktuellen Spieles im Leaderboard vermerkt werden. [Abb.Mockup.Leaderboard]

Prototyp

Technische Hintergrundinfos

Entwicklungsumgebung und Zielplattformen



[Abb.Eclipse] Entwicklungsumgebung: Eclipse mit integriertem ADT-Plugin

Der vorliegende Prototyp der App „Bebras“ wurde mit der freien Entwicklungsumgebung Eclipse mit integriertem ADT-Plugin (Android Development Tool) und dem Android SDK (Software Development Kit) in Java (Programmlogik) und XML (Layouts, Ressource-Files und Android-Manifest) entwickelt.

Getestet wurde die App auf einem Google „Nexus 4“-Smartphone sowie auf einem Google „Nexus 7“-Tablet. Auf beiden Geräten war die zum Zeitpunkt des Verfassens dieser Arbeit aktuelle Android-Version 4.4.4 (KitKat) installiert. Minimalvoraussetzung für das Ausführen der App ist die Android-Version 4.1 (Jelly Bean).

Implementierung der Layouts

Android-Applikationen setzen sich aus sogenannten Activities zusammen. Für diese Activities wurden jeweils eigene Layouts definiert. Die Layouts der Tasks und Module wurden teils statisch (XML) und teils dynamisch (Java, Zugriff zur Laufzeit) implementiert. Für alle Tasks wurde ein gemeinsames Layout-File definiert. Erst zur Laufzeit greift die Java-Klasse des entsprechenden Tasks auf dieses Layout-File zu und lädt die entsprechenden Inhalte des Tasks (Texte, Bilder) in das Layout. Darüberhinaus werden jene Bereiche des Layouts, die für den Task benötigt gewerden auf *sichtbar* bzw. jene, die nicht benötigt werden, auf *unsichtbar* gesetzt. Das folgende Beispiel illustriert die dynamische Layout-Erstellung beim Laden eines Tasks anhand des Task-Titels:

Im Layout-File weist das Attribut `android:id` dem `<TextView>`-Element die eindeutige ID „`tv_default_task_title`“ zu:

```
<TextView
        android:id="@+id/tv_default_task_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="5dp"
        android:paddingTop="5dp"
        android:text="@string/task_title"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textColor="@color/brown_dark"
        android:textStyle="bold" />
```

In einer Task-Klasse (z.B. Task01.java) kann nun folgendermaßen auf das Titel-Element des Tasks zugegriffen werden:

```
TextView title = (TextView) findViewById(R.id.tv_default_task_title);
title.setText(getResources().getString(R.string.title_activity_task01));
```

Die eigentlichen Inhalte der Tasks (Titel, Beschreibungen, Fragen, Antworten, Lösungen und Erklärungen) befinden sich in einem separaten XML-Ressource-File:

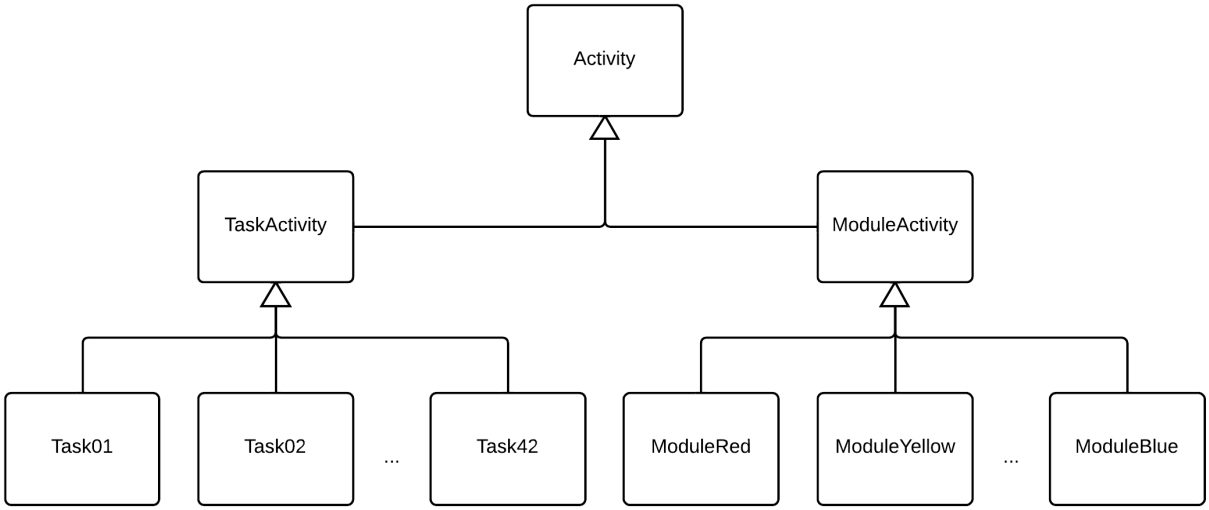
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="title_activity_task01">Titel des Tasks</string>
        ...
</resources>
```

Die Android-eigene Klasse „R“ weist allen Programmressourcen einen eindeutige Integer-Wert zu (im obigen Beispiel wird der Integer-Wert zum String des Task-Titels über `R.string.title_activity_task01` abgerufen. Die Methode `getResources().getString()` ermöglicht schließlich den Zugriff auf die Ressource.

Vererbung

Alle Activities in Java erben von der Klasse `android.app.Activity`. Zur Vermeidung von Redundanzen und um Änderungen in der Programmlogik zu vereinfachen wurde für die Task- und Modul-Klassen jeweils eine Super-Klasse `TaskActivity` bzw. `ModuleActivity` geschaffen. Diese Super-Klassen enthalten die gesamte Programmlogik, wohingegen die einzelnen Task- und Modul-Klassen vorrangig der dynamischen Layout-Erstellung (wie oben beschrieben) dienen.

Folgendes Klassendiagramm gibt die Vererbung innerhalb der Task- und Modul-Klassen wieder:



[Abb.Klassendiagramm] Tasks erben von der Klasse TaskActivity, Module von der Klasse ModuleActivity

Adapter

Abhängig davon welches Level gewählt wird, setzt sich ein Modul aus unterschiedlichen Tasks zusammen. Diese Anforderung machte eine dynamische Erstellung der Modul-Buttons, mit denen auf die einzelnen Tasks zugegriffen wird, notwendig. Android stellt hierfür sogenannte Adapter zur Verfügung - im konkreten Fall kam die Klasse `BaseAdapter` zum Einsatz. Durch Zusammenspiel des Layout-Elements `<GridView>` mit der Adapter-Klasse konnte erreicht werden, dass ohne unnötige Redundanzen im Programmcode die jeweils richtigen Tasks in ein Modul geladen werden. Die Task-Zusammensetzung wird hierbei in einfachen String-Arrays gespeichert und lässt sich nach Belieben ändern. Folgender Code-Ausschnitt gibt die Task-Zusammensetzung auf dem Level „Senior“ wieder:

```
String[] tasks_sen_green = { "09", "14", "15", "23", "28", "30" };
String[] tasks_sen_yellow = { "31", "32", "33", "35", "36", "37" };
String[] tasks_sen_red = { "34", "38", "39", "40", "41", "42" };
String[] bonustask_sen_purple = {"24"};
String[] bonustask_sen_pink = {"25"};
String[] bonustask_sen_blue = {"29"};
```

Die Module grün (leicht), gelb (mittel) und rot (schwer) beinhalten jeweils 6 Tasks, die Module violett, rosa und blau beinhalten die Bonus-Tasks.

Bilder (Drawables)

Alle in der App benutzten Graphiken stammen entweder aus den Biber-Katalogen 2012 und 2013, sind Teil des Android SDK oder stehen unter einer CC0 Public Domain-Lizenz.

Gespeichert werden diese Bilder (Drawables) in den Projektverzeichnissen /res/drawable. Weiters bietet Android die Möglichkeit Drawables für verschiedene Bildschirmauflösungen bereitzustellen. Diese werden in den Projektordnern /res/drawable-ldpi, /res/drawable-mdpi, /res/drawable-hdpi, /res/drawable-xhdpi und /res/drawable-xxhdpi abgelegt.

Analog zu String-Ressourcen ist auch bei Drawables der Zugriff zur Laufzeit über die Klasse „R“ möglich. Beispielsweise kann dem Layout-Element `ImageView img1` folgendermaßen ein Bild zugewiesen werden:

```
ImageView img1 = (ImageView) findViewById(R.id.iv_task_image_1);
img1.setImageResource(R.drawable.task01_img_1);
```

Diese Möglichkeit wurde, wie oben (anhand der String-Ressource des Task-Titels) beschrieben, zur dynamischen Erstellung der Layouts von Tasks und Modulen genutzt.

Speicherung des Spielzustands mit SharedPreferences

Zur persistenten Speicherung des Spielzustands wurden die von Android bereitgestellten SharedPreferences genutzt. Folgendermaßen kann auf die SharedPreferences zugegriffen werden:

```
SharedPreferences sharedPref = getSharedPreferences(getString(R.string.pref_key),
       Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
```

Folgendes Code-Beispiel illustriert die Verwendung von SharedPreferences und SharedPreferences.Editor anhand des Kaufs eines Hinweises:

```
int bc = sharedPref.getInt("bc", 0) - 1;
editor.putInt("bc", bc);
editor.putBoolean(task_id_string + "_hint_bought", true);
editor.commit();
```

Zunächst wird mit sharedPref.getInt(...) der aktuelle Stand an Bebras-Coins (BC) abgerufen (Standardwert: 0) und um den Wert 1 verringert (ein Hinweis kostet 1 BC). Nun wird der verringerte Wert mit editor.putInt(...) neu gesetzt. In Zeile 3 wird eine Boole'sche Variable auf `true` gesetzt, die indiziert, dass der Hinweis dieses Tasks nun gekauft wurde. Auf diese Weise kann bei einem Neustart des Tasks oder des Spiels verhindert werden, dass der Hinweis erneut gekauft werden müsste. In Zeile 4 werden mit editor.commit() die Änderungen wirksam.

Analog zu diesem Beispiel wurden die SharedPreferences dazu genutzt, um folgende Elemente des Spiels dauerhaft zu speichern:

- Spielername
- gewähltes Level
- bestandene und nicht bestandene Tasks
- Anzahl der Versuche, die man für einen Task benötigt hat
- abgeschlossene, freigeschaltete und gesperrte Module und Bonus-Tasks
- XP-Stand
- BC-Stand
- gesetzte oder bereits verwendete Timer

Wird ein Spiel fortgesetzt, so werden alle diese Werte wiederhergestellt. Beim Start eines neuen Spiels hingegen werden alle diese Variablen neu initialisiert.

Erweiterbarkeit

Da voraussichtlich auch in den kommenden Jahren Biber-Wettbewerbe stattfinden und demzufolge neue Aufgabenkatalog erscheinen werden, war die einfache Erweiterbarkeit der App ein wichtiges Kriterium.

Wie oben bereits beschrieben wurde beim Erstellen der App darauf wert gelegt (teilweise auch bedingt durch den Aufbau des Android-Frameworks), Layouts, Ressourcen und Programmlogik voneinander zu trennen. Durch die zusätzliche Verwendung von Adapter-Klassen und Vererbung bei Task- und Modul-Klassen ist es mit wenig Programmieraufwand möglich, die Tasks der App durch neue zu ersetzen bzw. mit diesen zu erweitern.

Der folgende Workflow gibt an, wie sich die App mit Tasks erweitern lässt (XX steht stellvertretend für die zweistellige Nummer des Tasks):

1.Anlegen der String-Ressourcen im File `task_contents.xml` **:**

```
<!-- Task XX -->
<string name="title_activity_taskXX">Titel</string>
<string name="taskXX_description_1">Beschreibung: Teil 1</string>
<string name="taskXX_description_2">Beschreibung: Teil 2</string>
<string name="taskXX_description_3">Beschreibung: Teil 3</string>
<string name="taskXX_description_4">Beschreibung: Teil 4</string>
<string name="taskXX_answer_A">Antwort A</string>
<string name="taskXX_answer_B">Antwort B</string>
<string name="taskXX_answer_C">Antwort C</string>
<string name="taskXX_answer_D">Antwort D</string>
<string name="taskXX_question">Frage</string>
<string name="taskXX_solution">{A|B|C|D} bei MC-Tasks | Text bei Input-Tasks</string>
<string name="taskXX_hint">Hinweis</string>
<string name="taskXX_explanation">Erklärung</string>
<string name="taskXX_why_cs">Das ist Informatik...</string>
```

2.Kopieren der Bilder des Tasks ins Projektverzeichnis `/res/drawable` **im Format PNG:**

Hierzu wurde folgende Konvention zur Namensgebung eingeführt:

taskXX_img_{1|2|3|4}.png	Bilder zur Beschreibung des Tasks
taskXX_img_{a|b|c|d}.png	Bilder zu den Antworten des Tasks
taskXX_expl.png	Bild für Dialog „Task-Erklärung“
taskXX_why.png	Bild für Dialog „Das ist Informatik

[Tab.DrawableNames] Konvention zur Benennung von Drawables

Hinweis: *die Einhaltung dieser Konventionen wird von der Klasse TaskActivity.java zwingend vorausgesetzt.*

3. Erstellung bzw. Adaptierung der Task-Klasse:

Beispiel-Klasse TaskXX.java:

```
package org.bebras.task;

import org.bebras.R;

import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

public class TaskXX extends TaskActivity {

	private final String task_id = "taskXX"; // Setzen der Task-ID

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		// lädt die String-Ressourcen taskXX_solution, taskXX_hint, taskXX_explanation und
		// taskXX_why_cs
		super.setTaskContents(task_id);

		// Zugriff auf die jeweiligen Layout-Elemente
		TextView title = (TextView) findViewById(R.id.tv_default_task_title);
		TextView taskDescription1 = (TextView) findViewById(R.id.tv_task_description_1);
		TextView taskQuestion = (TextView) findViewById(R.id.tv_task_question);
		ImageView img1 = (ImageView) findViewById(R.id.iv_task_image_1);
		LinearLayout ll_answers = (LinearLayout) findViewById(R.id.ll_text_answers);

		TextView answerA = (TextView) findViewById(R.id.tv_text_a);
		TextView answerB = (TextView) findViewById(R.id.tv_text_b);
		TextView answerC = (TextView) findViewById(R.id.tv_text_c);
		TextView answerD = (TextView) findViewById(R.id.tv_text_d);

		// Layout-Element für Text-Antworten auf sichtbar setzen
		ll_answers.setVisibility(LinearLayout.VISIBLE);

		// Texte und Bilder in das Layout laden
		title.setText(getResources().getString(R.string.title_activity_taskXX));
		taskDescription1.setText(getResources().getString(R.string.taskXX_description_1));
		img1.setImageResource(R.drawable.taskXX_img_1);
		taskQuestion.setText(getResources().getString(R.string.taskXX_question));

		answerA.setText("A) "+ getResources().getString(R.string.taskXX_answer_A));
		answerB.setText("B) "+ getResources().getString(R.string.taskXX_answer_B));
		answerC.setText("C) "+ getResources().getString(R.string.taskXX_answer_C));
		answerD.setText("D) "+ getResources().getString(R.string.taskXX_answer_D));
	}

	// Methode für Button „Hinweis anzeigen“ - implementiert in TaskActivity.java
	public void show_hint(View v){
		super.show_hint();
	}

	// Methode für Button „OK gedrückt“ - implementiert in TaskActivity.java
	public void ok_pressed(View v){
		super.ok_pressed();
	}
}
```

4. Hinzufügen des Tasks in das String-Array des Adapters:

Um den Task in ein Modul auf einem Level des Spiels zu integrieren, muss er lediglich im entsprechenden String-Array in der Methode `init_task_names` in der Klasse `ModuleAdapter` deklariert werden.

Zum Beispiel kann der Task XX (XX muss eine zweistellige ganze Zahl sein, der Klassenname muss TaskXX.java lauten) folgendermaßen zum Modul „Leicht“ (Grün) auf dem Level „Junior“ hinzugefügt werden:

```
String[] tasks_jun_green = { "09", "19", "21", "22", "23", "XX" };
```

5. Deklaration des Tasks im File AndroidManifest.xml

Da es sich bei Tasks um Activities handelt und Android eine Deklaration aller Activities im File `AndroidManifest.xml` verlangt, muss der Task zuguterletzt noch deklariert werden:

```
<activity
       android:name="org.bebras.task.TaskXX"
       android:label="@string/title_activity_taskXX"
       android:screenOrientation="portrait" >
</activity>
```

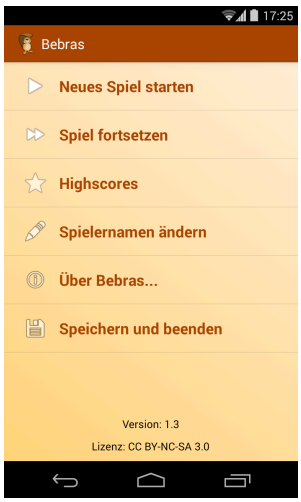
Das Attribut `android:name` gibt den Namen der Klasse des Tasks an, `android:label` verweist auf die String-Ressource `title_activity_taskXX`, die den Titel des Tasks enthält und `android:screenOrientation` wurde auf den Portrait-Mode gesetzt, was bedeutet, dass der Task nur im Hochformat angezeigt werden soll.

Dokumentation

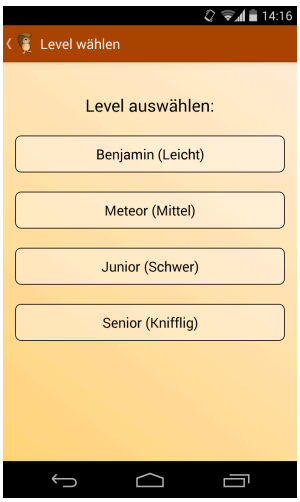
Start

Beim Start der App erscheint zunächst der Welcome-Screen [Abb.Bebras.Welcome] mit dem Bebras-Logo. Kurz darauf gelangt man in das Hauptmenü der App [Abb.Bebras.Menu].

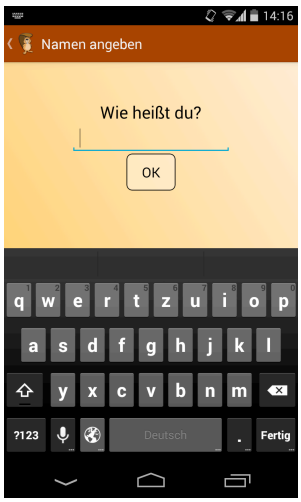
Welcome Screen
[Abb.Bebras.Welcome]



Hauptmenü
[Abb.Bebras.Menu]



Level-Auswahl
[Abb.Bebras.Level]



Angabe des Spielernamens
[Abb.Bebras.Name.1]

Hauptmenü

Das Hauptmenü der App bietet die folgenden Optionen an:

Neues Spiel starten

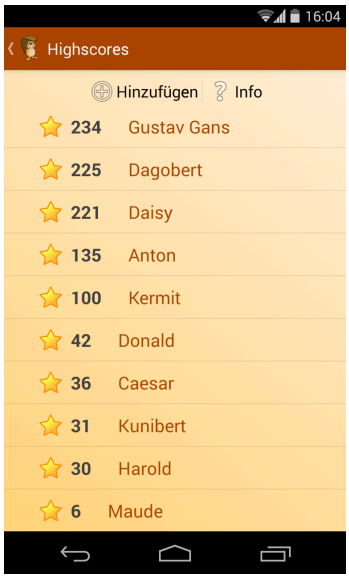
Hiermit wird ein neues Spiel gestartet. Der Spielstand des zuletzt gespielten Spieles wird zurückgesetzt. Beim Start eines neuen Spieles muss zunächst eines der vier Levels [Abb.Bebras.Level] *Benjamin* (Leicht), *Meteor* (Mittel), *Junior* (Schwer) und *Senior* (Knifflig) gewählt werden. Anschließend muss ein Spielername angegeben werden [Abb.Bebras.Name.1]. Nach diesen beiden Schritten gelangt man in die Overview-Ansicht der App.

Spiel fortsetzen

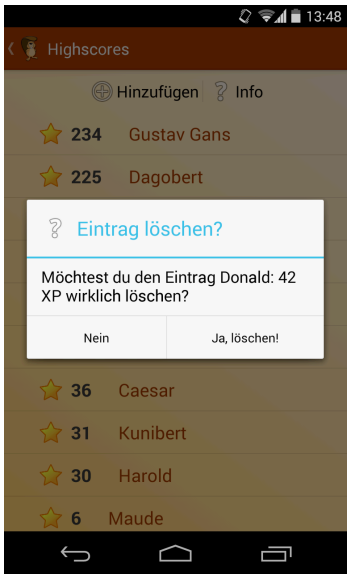
Ein bereits begonnenes Spiel fortsetzen. Der Spielstand des zuletzt gespielten Spieles wird wiederhergestellt. Nach Auswahl dieser Option gelangt man in die Overview-Ansicht des begonnenen Spieles.

Highscores

Hier können aktuelle Erfolge eingetragen und mit vergangenen verglichen werden. Durch einen Klick auf „Hinzufügen“ wird der aktuelle XP-Highscore gemeinsam mit dem Spielernamen in die Datenbank eingetragen. [Abb.Bebras.Highscores.1] XP-Scores kleiner gleich 0 können nicht eingetragen werden. Ein Highscore-Eintrag lässt sich löschen, indem man etwas länger darauf tippt und den darauf folgenden Dialog mit „Ja, löschen!“ bestätigt. [Abb.Bebras.Highscores.2]



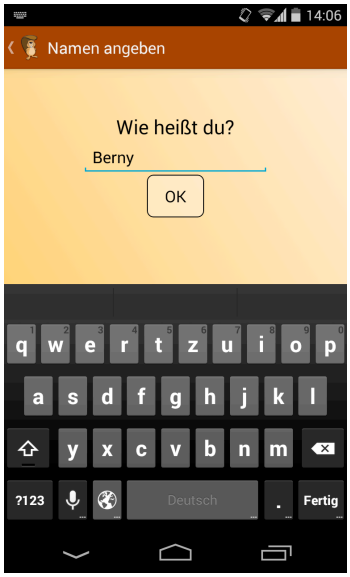
Highscores
[Abb.Bebras.Highscores.1]



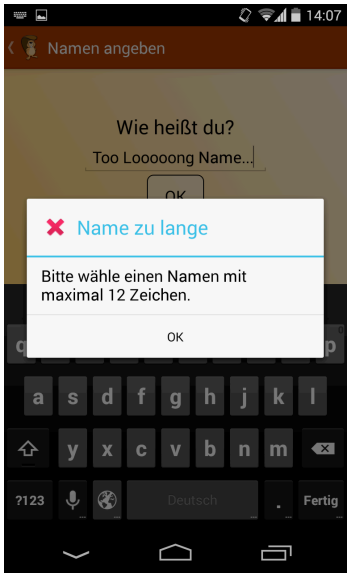
Highscore-Einträge löschen
[Abb.Bebras.Highscores.2]

Spielernamen ändern

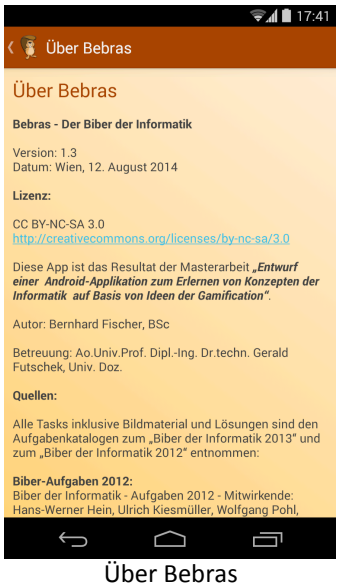
Über diesen Menü-Eintrag lässt sich der Spielername jederzeit ändern. [Abb.Bebras.Name.2] Ein Spielername darf eine maximale Länge von 12 Zeichen haben. [Abb.Bebras.Name.3]



Spielernamen ändern
[Abb.Bebras.Name.2]



Name zu lange
[Abb.Bebras.Name.3]



Über Bebras
[Abb.Bebras.About]

Über Bebras (About)

Zeigt die „About“-Seite [Abb.Bebras.About] mit folgenden Informationen zum Prototypen an:

- Version
- Datum
- Lizenz
- Allgemeine Informationen
- Quellen der Tasks, Clip-Arts und Sounds

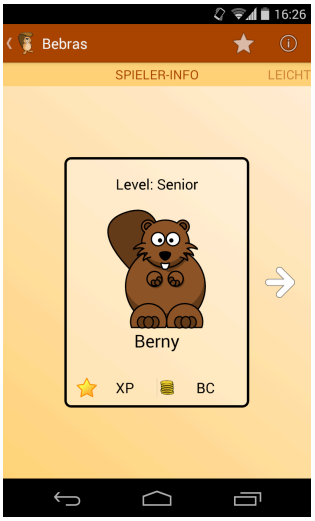
Speichern und beenden

Hierdurch wird der Spielzustand gesichert und die App beendet.

Overview-Ansicht

Die Overview-Ansicht ist die Hauptansicht des Spieles. Sie gliedert sich in fünf Bereiche, zwischen denen entweder durch eine Wisch-Geste nach links/rechts oder durch Tippen auf die entsprechenden Pfeile gewechselt werden kann. Oben wird die Aktionsleiste (Action Bar) angezeigt. Diese enthält einen Zurück-Button mit dem ins Hauptmenü zurückgekehrt werden kann, sowie Shortcuts zu den Highscores und zur „About“-Seite.

Die fünf Bereiche der Overview-Ansicht sind:



Spieler-Info
[Abb.Bebras.OV1]

Spieler-Info

Die Spieler-Info zeigt das gewählte Level, den Spieler-Avatar und den Namen des Spielers an. Durch einen Klick auf den XP- bzw. BC-Button kann der aktuelle Stand an Erfahrungspunkten bzw. Bebras-Coins abgerufen werden. Durch eine Wisch-Geste nach Links bzw. Tippen auf den nach rechts zeigenden Pfeil wird in die Overview-Ansicht des Moduls „Leicht“ gewechselt.

Modul Leicht / Mittel / Schwer (Overview-Ansicht)

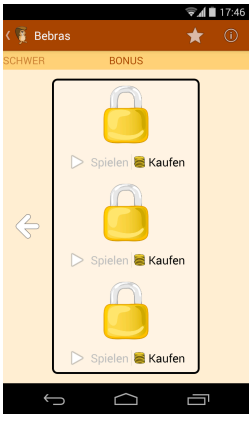
Modul „Leicht“ - abgeschlossen [Abb.Bebras.OV2]

Modul „Mittel“ - entsperrt [Abb.Bebras.OV3]

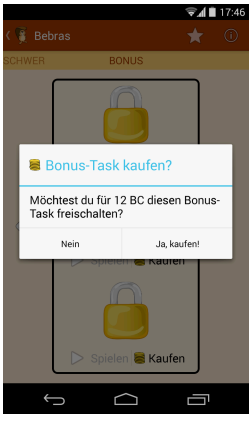
Modul „Schwer“ - gesperrt [Abb.Bebras.OV4]

Von hier aus kann der Status des jeweiligen Moduls eingesehen werden: ein grauer Avatar zeigt an, dass ein Modul entsperrt wurde [Abb.Bebras.OV3], das Schloss [Abb.Bebras.OV4] zeigt ein gesperrtes Modul an, ein farbiger Avatar sowie farbiger Hintergrund [Abb.Bebras.OV2] zeigen an, dass ein Modul abgeschlossen wurde. Durch Klick auf den Button „Spielen“ kann ein entsperrtes Modul geöffnet werden. Ein Modul wird durch Abschluss des Vorgänger-Moduls freigeschalten. Darüber informiert auch der Hinweis-Text, der bei Klick auf das Fragezeichen sichtbar wird.

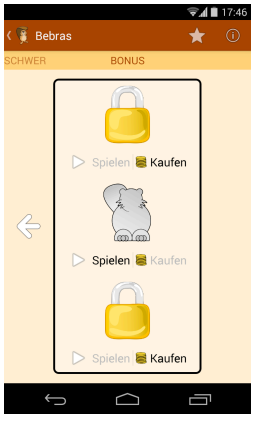
Bonus-Tasks (Overview-Ansicht)



Bonus-Tasks - gesperrt [Abb.Bebras.OV5]



Bonus-Task kaufen [Abb.Bebras. OV6]



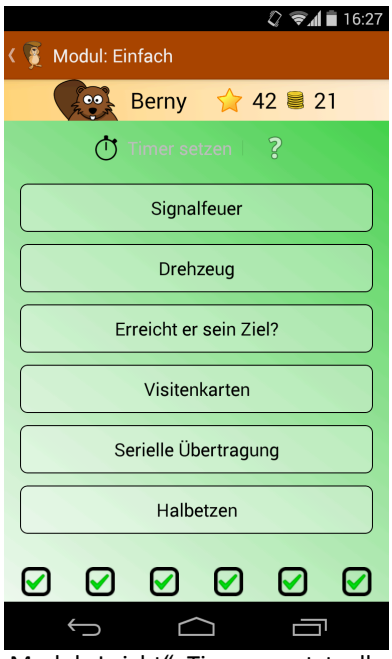
Bonus-Task entsperrt [Abb.Bebras. OV7]

2 Bonus-Tasks abgeschlossen [Abb.Bebras. OV8]

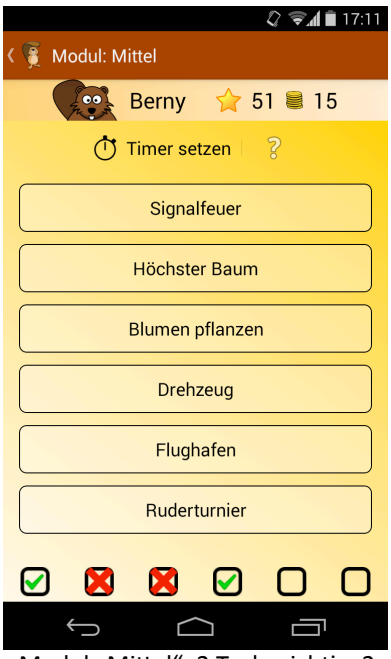
Pro Level gibt es drei Bonus-Tasks, die sich für jeweils 12 Bebras-Coins durch Klick auf den „Kaufen“-Button freischalten lassen [Abb.Bebras.OV5]. Hat man zu wenige BC, so erhält man einen

entsprechenden Hinweis. Nach dem Kauf eines Bonus-Tasks zeigt ein grauer Avatar an, dass dieser nun freigeschalten ist [Abb.Bebras.OV7]. Durch Klick auf den Button „Spielen“ gelangt man zum Bonus-Task. Ein abgeschlossener Bonus-Task wird durch einen farbigen Avatar angezeigt. Der Hintergrund des Bonus-Task-Bereiches der Overview-Ansicht nimmt die Farbe des zuletzt abgeschlossenen Bonus-Tasks an [Abb.Bebras.OV8].

Module (geöffnet)



Modul „Leicht“: Timer gesetzt, alle Tasks abgeschlossen [Abb.Bebras.Mod1]



Modul „Mittel“: 2 Tasks richtig, 2 Tasks falsch [Abb.Bebras.Mod2]

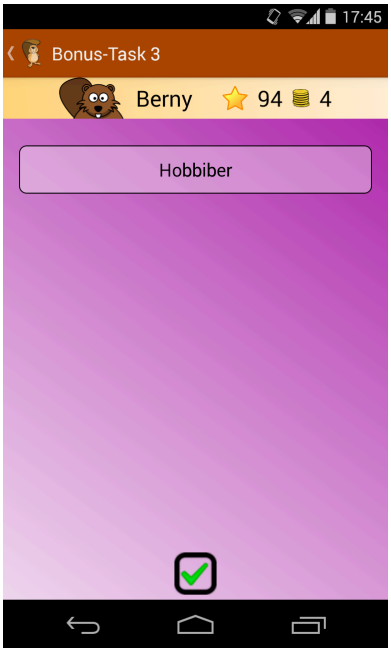
Modul „Schwer“: noch keine Tasks beantwortet [Abb.Bebras.Mod3]

Die drei Module „Leicht“, „Mittel“ und „Schwer“ beinhalten jeweils eine Header-Zeile, die den Spielernamen sowie den XP- und BC-Stand anzeigt. Darunter befindet sich ein Button zum Setzen eines Timers. Pro Modul kann für genau einen Task ein Timer (Leicht: 120 Sekunden, Mittel: 135 Sekunden, Schwer: 150 Sekunden) gesetzt werden. Für den Task, der als nächstes gestartet wird, wird nach dem Setzen eines Timers ein Countdown gestartet. Schafft man es, den Task innerhalb der vorgegebenen Zeit richtig zu beantworten, so erhält man die doppelte XP-Punktezahl. (Um Schummeln zu verhindern, ist es nicht möglich, einen Timer auf einen Task anzuwenden, der bereits geöffnet wurde.) Jedes Modul enthält 6 Buttons, mit denen auf die jeweiligen Tasks zugegriffen werden kann. Darunter befinden sich Checkboxen, die den Fortschritt anzeigen. Hierbei repräsentiert ein grünes Häkchen einen korrekt beantworteten Task, ein rotes X steht für einen falsch beantworteten. Ein leeres Quadrat bedeutet, dass der entsprechende Task noch nicht geöffnet wurde. Besteht man den letzten Task eines Moduls, so wird ein Sound abgespielt, der bedeutet, dass das Modul abgeschlossen und das Folge-Modul freigeschalten wurde.

Bonus-Tasks (geöffnet)

Bonus-Task 1
[Abb.Bebras.Bonus1]

Bonus-Task 2
[Abb.Bebras.Bonus2]



Bonus-Task 3
[Abb.Bebras.Bonus3]

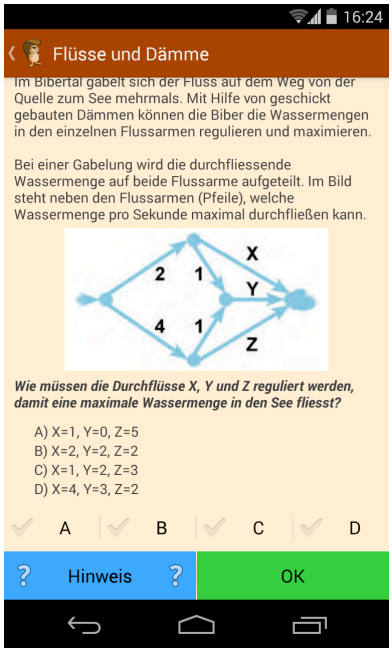
Bei geöffneten Bonus-Tasks handelt es sich um eine Sonderform von Modulen, die aber nicht 6, sondern jeweils nur einen Task enthalten und wie bereits mehrfach erwähnt durch Bezahlung von 12 BC freigeschalten werden können. Bonus-Tasks bringen die meisten Erfahrungspunkte (15 XP beim ersten Versuch, 5 XP beim zweiten Versuch), jedoch auch die größten Abzüge bei falschen Antworten (-5XP). (Siehe [Tab.XP.Schema] im Kapitel *Designkonzept*.)

Tasks

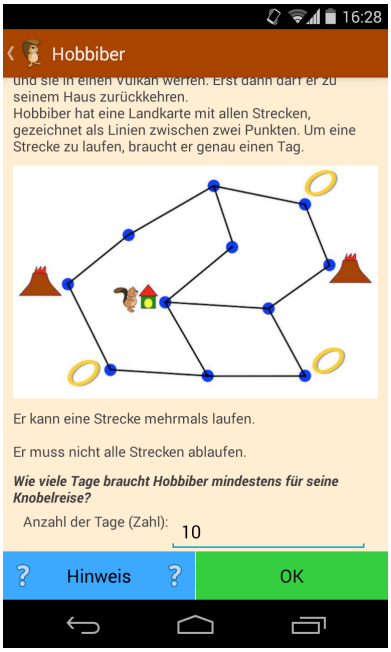
Die Task-Ansicht besteht aus einem scrollbaren Bereich, welcher den Titel des Tasks, den Angabetext und Bilder sowie etwaige Antwortmöglichkeiten enthält. Darunter befindet sich - je nachdem, ob es sich um einen Multiple-Choice-Task oder einen Task mit Input über die Tastatur enthält - entweder ein Bereich, der die Buttons A, B, C und D enthält (von denen jeweils nur einer ausgewählt werden kann) oder ein Bereich zur Eingabe des Input-Textes.

Darunter wiederum befinden sich zwei Buttons: der Hinweis-Button mit dem für 1 BC ein Hinweis gekauft werden kann sowie der „OK“-Button mit dem die ausgewählte bzw. eingegebene Lösung bestätigt wird. Am meisten XP und BC erhält man, wenn man einen Task beim ersten Mal richtig beantwortet, beim zweiten Versuch erhält man noch ein Drittel der Punkte. Für falsche Antworten gibt es Punkte-Abzüge. (Das dazugehörige Schema entnehme der interessierte Leser / die interessierte Leserin den Tabellen [Tab.XP.Schema] und [Tab.BC.Schema] im Kapitel *Designkonzept*.)

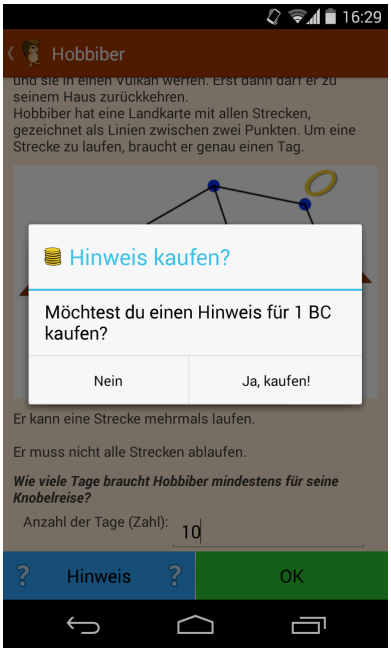
Nachdem ein Task richtig beantwortet wurde, erscheint zunächst ein Dialog mit dem Titel „Richtig“, welcher die Erklärung der Lösung (und eventuell ein Lösungsbild) enthält. Bestätigt man diesen mit „OK“, so erscheint ein zweiter Dialog mit dem Titel „Das ist Informatik!“ - dieser stellt den Zusammenhang des jeweiligen Tasks mit Konzepten, Paradigmen und Aufgabengebieten aus der Informatik her.



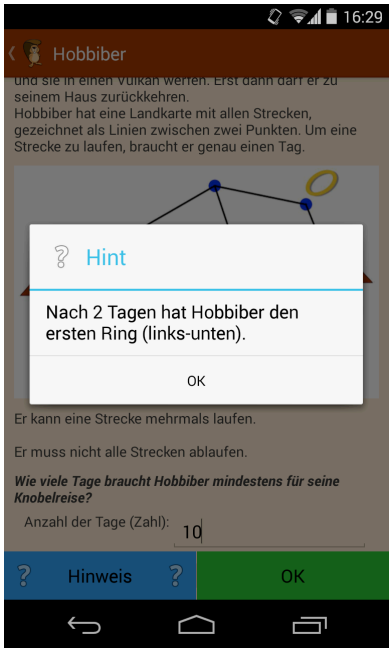
Multiple-Choice-Task
[Abb.Bebras.Task.MC]



Task mit Input
[Abb.Bebras.Task.Input]

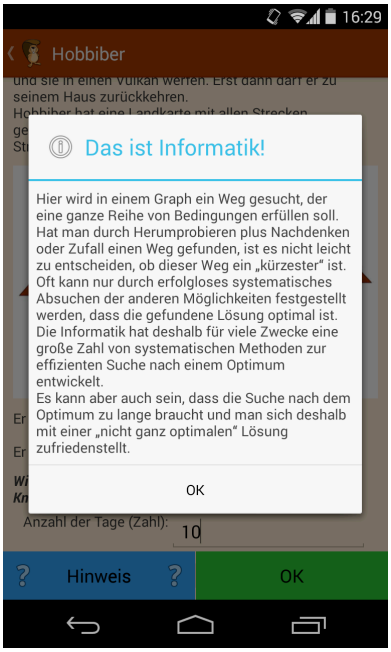


Dialog: „Hinweis kaufen?“
[Abb.Bebras.Hint1]



Hinweis (Hint)
[Abb.Bebras.Hint2]

Dialog: „Richtig“
[Abb.Bebras.Task.expl]



Dialog: „Das ist Informatik!“
[Abb.Bebras.Task.why]

Evaluierung

Methodik

Nach Abschluss des Entwicklungsprozesses wurde eine ***summative Evaluation*** des Prototyps durchgeführt:
„Die summative Evaluation wird am Ende einer Softwareentwicklung eingesetzt. Sie kann nicht zur Qualitätsverbesserung, sondern nur zur Bewertung und damit zur Einsatzentscheidung beitragen. Weiterhin können die Erkenntnisse in Nachfolgeprodukte oder späteren Versionen Verwendung finden. Gegenüber der formativen Evaluation werden nicht nur einzelne Punkte bewertet, sondern auch die Durchgängigkeit der Realisierung.“ [Klante 1999]

Zur Durchführung dieser summativen Evaluation wurde mit Hilfe des Online-Umfragedienstes SurveyMonkey (https://de.surveymonkey.com) ein Online-Fragebogen erstellt. Dieser Online-Fragebogen wurde von insgesamt 20 Teilnehmenden mit unterschiedlichem Bildungshintergrund beantwortet. Voraussetzung hierfür war die vorhergehende Installation und Testung des Prototyps auf einem Android-Device (Smartphone oder Tablet). Der Fokus dieses Online-Fragenkatalogs lag auf folgenden zwei Themen:

- *Motivation* (Wie sinnvoll wurden die Gamedesign-Elemente eingesetzt?)
- *Usability* (Wie gut lässt sich die Software benutzen?)

Ziel dieses Online-Fragebogens war einerseits die Identifikation von qualitativen Mängeln bzw. Fehlern im Prototypen, sodass deren Behebung in zukünftige Versionen der App einfließen kann, andererseits die Beantwortung der wissenschaftlichen Fragestellung, sprich:
Inwiefern ist eine sinnvolle Integration der Gamedesign-Elemente in den Prototypen gelungen und wo herrscht diesbezüglich Verbesserungspotenzial?

Konkret gliederte sich der Fragenkatalog in folgende Abschnitte:
- *Anonyme Fragen zur Person* (Alter, Geschlecht, Selbsteinschätzung der Informatik-Kenntnisse)
- *Fragen zur Usability des Prototyps*
- *Fragen zur Motivation durch die Gamedesign-Elemente*
- *Ranking der Gamedesign-Elemente nach deren Wichtigkeit*
- *Freitextantworten (Anregungen und Kritik)*

Ergebnisse

Teilnehmende

Insgesamt nahmen 20 Personen, davon 11 Männer und 9 Frauen [Abb.Stat.Geschlecht], an der Umfrage teil, die alle zuvor den Prototypen der App auf einem Android-Device (Smartphone oder Tablet) getestet hatten. Die überwiegende Mehrheit der Teilnehmer/innen stammte zum Zeitpunkt der Umfragebeantwortung aus der Alterskategorie der 20- bis 29-jährigen (14 Personen), zwei Personen waren zunter zwanzig, vier Personen über dreißig Jahre alt [Abb.Stat.Alter].

An dieser Stelle sei darauf hingewiesen, dass ursprünglich geplant war, vor allem Schülerinnen und Schüler zu befragen, da diese ja die Zielgruppe der Biber-Wettbewerbe darstellen. Da die Evaluation jedoch in der Ferienzeit stattfand, gestaltete sich dieses Vorhaben als schwer realisierbar.

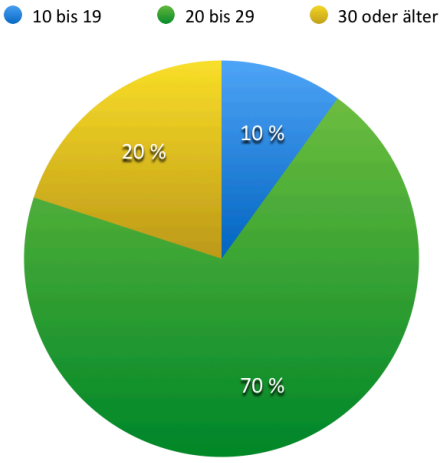
Im Gegensatz zu den Biber-Wettbewerben richtet sich die App „Bebras“ an jede Interessierte Nutzerin und jeden interessierten Nutzer beliebigen Alters, weshalb die Durchführung der Evaluierung mit einer älteren Testgruppe annehmbar erschien.

Die eigene Informatik-Kompetenz schätzten drei der Befragten mit „gering“, sechs mit „mittel“, sieben mit „hoch“ und vier mit „sehr hoch“ [Abb.Stat.Skills1], im Durchschnitt also zwischen „mittel“ und „hoch“ [Abb.Stat.Skills2] ein. Da die Umfrage nicht vorrangig unter Informatiker/innen durchgeführt wurde, ist davon auszugehen, dass von einigen der Befragten hierbei von der eigenen Anwendungskompetenz ausgegangen wurde und nicht unbedingt von logisch-analytischen Fähigkeiten.

Alter

	Anzahl	Prozent
10 bis 19	2	10 %
20 bis 29	14	70 %
30 oder älter	4	20 %

[Tab.Stat.Alter]

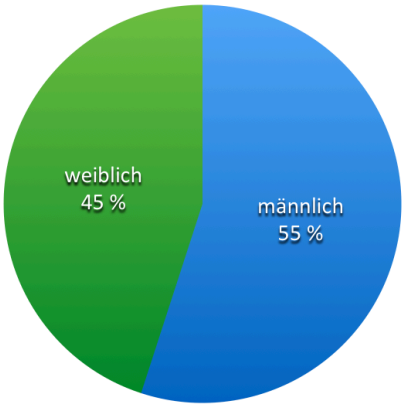


[Abb.Stat.Alter]

Geschlecht

	Anzahl	Prozent
männlich	11	55 %
weiblich	9	45 %

[Tab.Stat.Geschlecht]

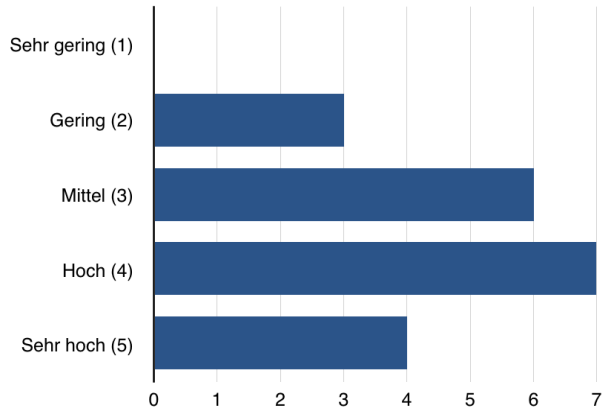


[Abb.Stat.Geschlecht]

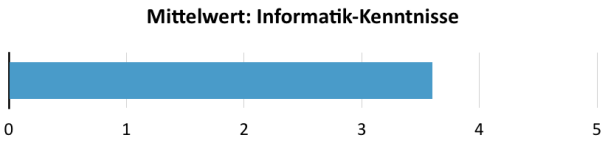
Wie würdest du deine Informatik-Kenntnisse einschätzen?

Sehr gering (1)	Gering (2)	Mittel (3)	Hoch (4)	Sehr hoch (5)	Beantwortungen	Mittelwert
0	3	6	7	4	20	**3,6**

[Tab.Stat.Skills]



[Abb.Stat.Skills1]



[Abb.Stat.Skills2]

Usability des Prototyps

Zur Evaluierung der Usability wurden insgesamt 17 Aussagen definiert, die jeweils mit einer von fünf Optionen („Stimme überhaupt nicht zu“, „Stimme nicht zu“, „Weder noch“, „Stimme eher zu“, „Stimme völlig zu“) bewertet werden konnten. [Tab.Stat.Usability]

Hierbei erreichten folgende Aussagen die **höchste Zustimmung** ($4{,}25 \leq \bar{x} < 4{,}5$):

- Es ist einfach ein Spiel abzubrechen und dort fortzusetzen, wo ich aufgehört habe. ($\bar{x} = 4{,}45$)
- Die App ist insgesamt übersichtlich strukturiert. ($\bar{x} = 4{,}35$)
- Die Module selbst sind übersichtlich strukturiert und gut zu bedienen. ($\bar{x} = 4{,}35$)
- Es ist einfach Hinweise zu kaufen. ($\bar{x} = 4{,}35$)
- Der Spielername lässt sich leicht ändern. ($\bar{x} = 4{,}35$)
- Es ist einfach und klar verständlich, wie man Module freischaltet. ($\bar{x} = 4{,}26$)
- Die App ist insgesamt benutzerfreundlich. ($\bar{x} = 4{,}25$)

Relativ hohe Zustimmung erhielten diese Aussagen ($4 \leq \bar{x} < 4{,}25$):

- Die Tasks sind übersichtlich strukturiert und gut zu bedienen. ($\bar{x} = 4{,}20$)
- Die Navigation ist einfach und sinnvoll. ($\bar{x} = 4{,}20$)
- Die verwendeten Bilder sind gut zu erkennen und hilfreich bei der Lösung der Tasks. ($\bar{x} = 4{,}15$)
- Die App ist insgesamt optisch ansprechend. ($\bar{x} = 4{,}11$)
- Es ist einfach Highscores einzutragen oder zu löschen. ($\bar{x} = 4{,}10$)
- Es ist einfach Bonus-Tasks zu kaufen. ($\bar{x} = 4{,}10$)
- Die Modul-Übersicht ist übersichtlich strukturiert und gut zu bedienen. ($\bar{x} = 4{,}05$)

Diese beiden Aussagen erreichten nur **geringe Zustimmung** ($3{,}75 \leq \bar{x} < 4$):

- Die Texte der Tasks sind gut lesbar und verständlich. ($\bar{x} = 3{,}95$)
- Die Dialoge sind gut lesbar, hilfreich und verständlich. ($\bar{x} = 3{,}90$)

Die Qualität der Sounds erhielt die **geringste Zustimmung**:

- Die Sounds finde ich ansprechend. ($\bar{x} = 3{,}45$)

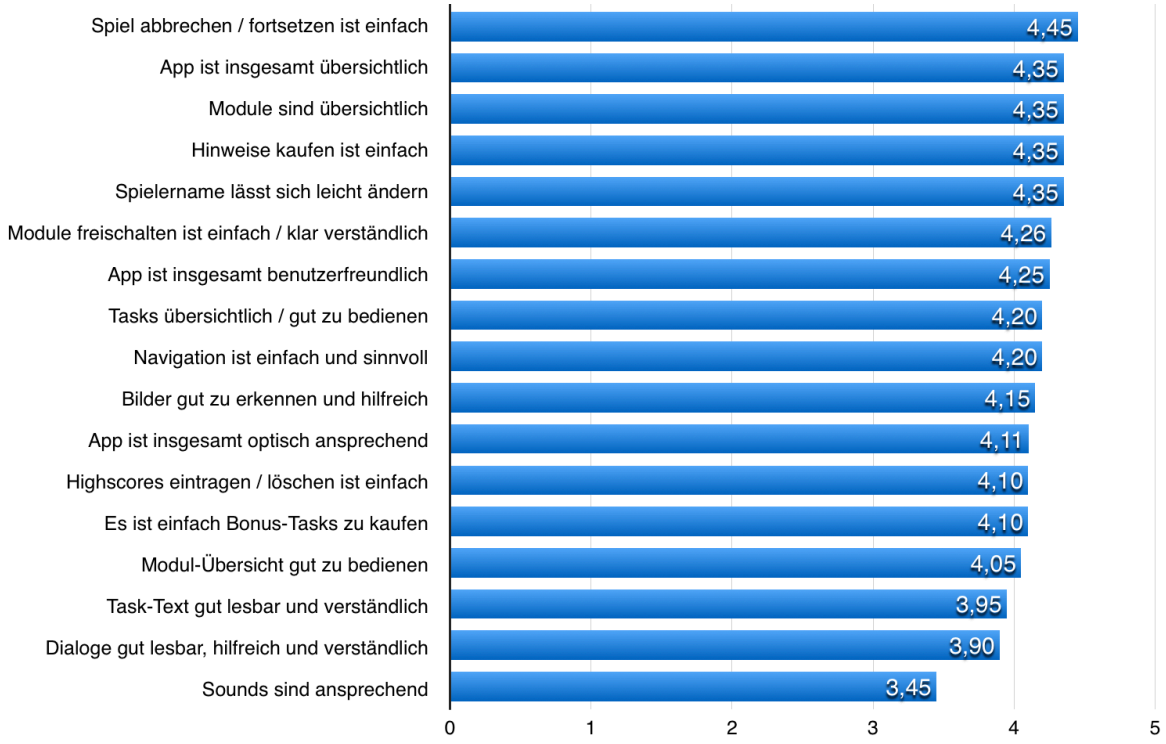
	Stimme überhaupt nicht zu (1)	Stimme eher nicht zu (2)	Weder noch (3)	Stimme eher zu (4)	Stimme völlig zu (5)	Antworten	Mittelwert
Es ist einfach ein Spiel abzubrechen und dort fortzusetzen, wo ich aufgehört habe.	0	1	1	6	12	20	**4,45**
Die App ist insgesamt übersichtlich strukturiert.	0	0	2	9	9	20	**4,35**
Die Module selbst sind übersichtlich strukturiert und gut zu bedienen.	0	0	1	11	8	20	**4,35**
Es ist einfach Hinweise zu kaufen.	0	0	3	7	10	20	**4,35**
Der Spielername lässt sich leicht ändern.	0	1	2	6	11	20	**4,35**
Es ist einfach und klar verständlich, wie man Module freischaltet.	0	1	1	9	8	19	**4,26**
Die App ist insgesamt benutzerfreundlich.	0	0	0	15	5	20	**4,25**
Die Tasks sind übersichtlich strukturiert und gut zu bedienen.	0	1	1	11	7	20	**4,20**
Die Navigation ist einfach und sinnvoll.	0	1	2	9	8	20	**4,20**
Die verwendeten Bilder sind gut zu erkennen und hilfreich bei der Lösung der Tasks.	1	1	2	6	10	20	**4,15**
Die App ist insgesamt optisch ansprechend.	0	1	3	8	7	19	**4,11**
Es ist einfach Highscores einzutragen oder zu löschen.	0	2	3	6	9	20	**4,10**
Es ist einfach Bonus-Tasks zu kaufen.	0	1	4	7	8	20	**4,10**
Die Modul-Übersicht ist übersichtlich strukturiert und gut zu bedienen.	0	1	2	12	5	20	**4,05**
Die Texte der Tasks sind gut lesbar und verständlich.	1	1	3	7	7	19	**3,95**
Die Dialoge sind gut lesbar, hilfreich und verständlich.	1	1	1	13	4	20	**3,90**
Die Sounds finde ich ansprechend.	1	2	7	7	3	20	**3,45**

[Tab.Stat.Usability]

Insgesamt wurden also all jene Aussagen, die die Struktur, Navigation, Übersichtlichkeit und Einfachheit des Prototyps betreffen mit hoher bis sehr hoher Zustimmung bewertet.

Verbesserungspotenzial konnte bei den Texten der Tasks und Dialoge geortet werden. Hier sollte eine zukünftige Version der App jedenfalls die Option mitbringen, die Schriftgröße an die persönliche Sehleistung anzupassen. Die Formulierung „gut lesbar und verständlich“ bzw. „gut lesbar, hilfreich und verständlich“ stellte sich im Nachhinein als suboptimal heraus, da dadurch keine scharfe Abgrenzung zwischen der Bewertung der Lesbarkeit und der Formulierung der Texte gezogen werden konnte. In den Textkommentaren der Evaluierung wurde jedoch explizit auf die Lesbarkeit der Texte eingegangen.

Die geringste Zustimmung erhielt die Aussage „Die Sounds finde ich ansprechend“. Daraus schlussfolgernd sollte eine zukünftige Version des Prototyps jedenfalls die Option zur Verfügung stellen, die Sounds in den Einstellungen der App abzustellen, deren Lautstärke zu regulieren und eventuell zwischen verschiedenen Sound-Profilen zu wechseln.



[Abb.Stat.Usability]

Gamedesign-Elemente

Motivation durch Gamedesign-Elemente

Um festzustellen, welche Gamedesign-Elemente des Prototyps am meisten bzw. am wenigsten zur Motivation der Nutzer/innen beitragen, wurden zwei Methoden eingesetzt: einerseits eine Bewertung von 12 Aussagen nach dem selben Schema, welches auch bei der Evaluierung der Usability zum Einsatz kam, andererseits ein Ranking der Gamedesign-Elemente nach der empfundenen Wichtigkeit derselben.

Folgende Aussage erreichte die **höchste Zustimmung**:

- Es macht Spaß, Tasks richtig zu beantworten. ($\bar{x} = 4{,}75$)

Ebenfalls **hohe Zustimmung** erreichten die folgenden Aussagen ($4 \leq \bar{x} < 4{,}5$):

- Ein Modul abzuschließen (Avatar freischalten und nächstes Modul entsperren), erzeugt ein Erfolgserlebnis. ($\bar{x} = 4{,}45$)
- Die Gliederung der Tasks in die Module "Leicht", "Mittel", "Schwer" und in Bonus-Tasks finde ich gelungen. ($\bar{x} = 4{,}30$)
- Es erzeugt ein Erfolgserlebnis, Erfahrungspunkte für abgeschlossene Tasks zu erhalten. ($\bar{x} = 4{,}05$)

Etwas **geringere Zustimmung** fand sich bei diesen Aussagen($3{,}5 \leq \bar{x} < 4$):

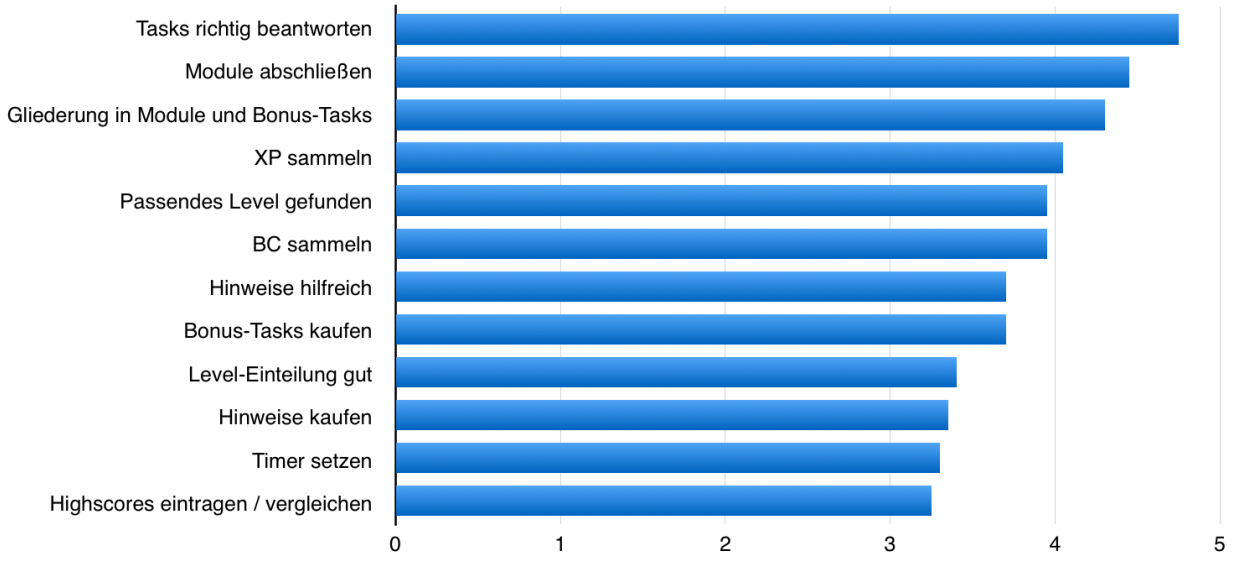
- Ich konnte ein für meine Fähigkeiten passendes Level finden. ($\bar{x} = 3{,}95$)
- Es erzeugt ein Erfolgserlebnis, Geld (BC) für abgeschlossene Tasks zu erhalten. ($\bar{x} = 3{,}95$)
- Die Hinweise sind hilfreich bei der Lösung der Tasks. ($\bar{x} = 3{,}70$)
- Es macht Spaß, Bonus-Tasks zu kaufen. ($\bar{x} = 3{,}70$)

Die **geringste Zustimmung** fand sich bei folgenden Aussagen ($3 \leq \bar{x} < 3{,}5$):

- Die Einteilung in die Levels "Benjamin" (Leicht), "Meteor" (Mittel), "Junior" (Schwer) und "Senior" (Knifflig) finde ich gut. ($\bar{x} = 3{,}40$)
- Es macht Spaß, Hinweise zu kaufen. ($\bar{x} = 3{,}35$)
- Es motiviert mich, wenn ich einen Timer setze, unter Zeitdruck stehe und mehr Punkte für einen Task sammeln kann. ($\bar{x} = 3{,}30$)
- Es motiviert mich, meinen Highscore einzutragen und mit anderen zu vergleichen. ($\bar{x} = 3{,}25$)

	Stimme überhaupt nicht zu (1)	Stimme eher nicht zu (2)	Weder noch (3)	Stimme eher zu (4)	Stimme völlig zu (5)	Antworten	Mittelwert
Es macht Spaß, Tasks richtig zu beantworten.	0	0	0	5	15	20	**4,75**
Ein Modul abzuschließen (Avatar freischalten und nächstes Modul entsperren), erzeugt ein Erfolgserlebnis.	0	0	2	7	11	20	**4,45**
Die Gliederung der Tasks in die Module "Leicht", "Mittel", "Schwer" und in Bonus-Tasks finde ich gelungen.	0	0	4	6	10	20	**4,30**
Es erzeugt ein Erfolgserlebnis, Erfahrungspunkte (XP) für abgeschlossene Tasks zu erhalten.	0	1	4	8	7	20	**4,05**
Ich konnte ein für meine Fähigkeiten passendes Level finden.	0	3	2	8	7	20	**3,95**
Es erzeugt ein Erfolgserlebnis, Geld (BC) für abgeschlossene Tasks zu erhalten.	1	1	5	4	9	20	**3,95**
Die Hinweise sind hilfreich bei der Lösung der Tasks.	0	3	5	7	5	20	**3,70**
Es macht Spaß, Bonus-Tasks zu kaufen.	0	1	9	5	5	20	**3,70**
Die Einteilung in die Levels "Benjamin" (Leicht), "Meteor" (Mittel), "Junior" (Schwer) und "Senior" (Knifflig) finde ich gut.	1	5	4	5	5	20	**3,40**
Es macht Spaß, Hinweise zu kaufen.	1	2	9	5	3	20	**3,35**
Es motiviert mich, wenn ich einen Timer setze, unter Zeitdruck stehe und mehr Punkte für einen Task sammeln kann.	1	3	8	5	3	20	**3,30**
Es motiviert mich, meinen Highscore einzutragen und mit anderen zu vergleichen.	1	4	7	5	3	20	**3,25**

[Tab.Stat.Motivation]



[Abb.Stat.Motivation]

Insgesamt erreichten alle Aussagen einen Mittelwert von $\bar{x} > 3$, also positive Zustimmung. Es kann demnach davon ausgegangen werden, dass die Integration der Gamedesign-Elemente *summativ* gelungen ist. Wenig überraschend war, dass die Tasks selbst und der Versuch diese zu lösen, den größten Motivationsfaktor des Spieles ausmachen. Auch die Gliederung in Module und Bonus-Tasks sowie das Sammeln von XP und BC stellten sich als durchaus motivierend für die Nutzer/innen heraus.

Der Motivationsfaktor von Bonus-Tasks und Hinweisen war etwas geringer als erwartet. Mögliche Erklärungsversuche hierfür sind der relativ hohe Preis, der für Bonus-Tasks zu zahlen ist (12 BC) und Hinweise werden vermutlich mit einer negativen Emotion verbunden, da man diese ja just dann benötigt, wenn man nicht im Stande war, eine Aufgabe ohne Hilfestellung zu lösen. Nichtsdestotrotz wurden Hinweise im folgenden Ranking als eines der wichtigsten Gamedesign-Elemente (Platz 3) bewertet.

Durch Implementierung zusätzlicher Tasks könnte verhindert werden, dass Tasks in mehr als einem Level vorkommen und so die Zustimmung zur Level-Einteilung möglicherweise verbessern.

Die geringste Zustimmung als positive Motivationsfaktoren erhielten Highscores und Timer. Unter Zeitdruck zu stehen wurde von den wenigsten Teilnehmenden als motivierend wahrgenommen. Eine Verbesserungsmöglichkeit der Highscores wäre die Implementierung eines Online-Leaderboards, welches es Nutzer/innen ermöglicht, ihre Scores weltweit mit anderen zu vergleichen. Hierfür wäre eine Anbindung an soziale Netzwerke vermutlich sinnvoll.

Ranking der Gamedesign-Elemente

Um festzustellen, welche Gamedesign-Elemente für die Nutzer/innen wichtiger waren als andere, wurden die Teilnehmenden der Evaluierung gebeten, diese in einem Ranking nach deren Wichtigkeit zu sortieren. Zur Feststellung des durchschnittlichen Rankings erhielt ein auf Platz 1 gereihtes Gamedesign-Element 9 Punkte, ein auf Platz 2 gereihtes 8 Punkte etc.

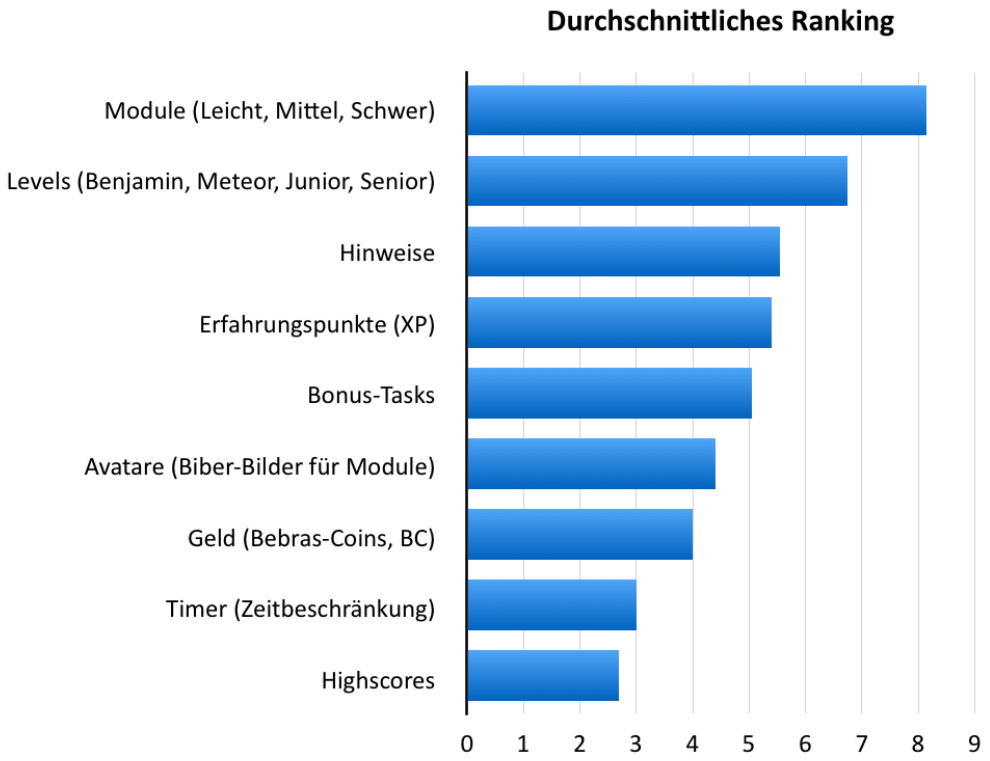
Die Gamedesign-Elemente erhielten im Schnitt folgendes Ranking:

1. Module ($\bar{x}$ = 8,15)
2. Levels ($\bar{x}$ = 6,75)
3. Hinweise ($\bar{x}$ = 5,55)
4. Erfahrungspunkte ($\bar{x}$ = 5,40)
5. Bonus-Tasks ($\bar{x}$ = 5,05)
6. Avatare ($\bar{x}$ =4,40)
7. Geld ($\bar{x}$ = 4,00)
8. Timer ($\bar{x}$ = 3,00)
9. Highscores ($\bar{x}$ = 2,70)

Module und Levels stellten sich als die wichtigsten ($\bar{x} > 6$) Gamedesign-Elemente heraus. Im Mittelfeld ($\bar{x} \geq 4$) fanden sich Hinweise, Erfahrungspunkte, Bonus-Tasks, Avatare und Geld (Bebras-Coins) wieder. Deutlich abgeschlagen auf den letzten beiden Plätzen ($\bar{x} \leq 3$) fanden sich Highscores und Timer wieder. Wie sich herausstellte, sind diese beiden Elemente in dieser Form kein wesentlicher Faktor für den Spielspaß.

	1	2	3	4	5	6	7	8	9	Antworten	Durchschnittliches Ranking
Module (Leicht, Mittel, Schwer)	10	6	2	1	1	0	0	0	0	20	**8,15**
Levels (Benjamin, Meteor, Junior, Senior)	5	6	3	1	1	1	1	1	1	20	**6,75**
Hinweise	0	1	7	3	4	3	0	2	0	20	**5,55**
Erfahrungspunkte (XP)	2	2	1	3	5	4	2	1	0	20	**5,40**
Bonus-Tasks	1	1	3	6	2	1	3	1	2	20	**5,05**
Avatare (Biber-Bilder für Module)	1	2	1	4	1	3	3	1	4	20	**4,40**
Geld (Bebras-Coins, BC)	1	1	2	1	3	2	3	4	3	20	**4,00**
Timer (Zeitbeschränkung)	0	0	0	1	3	4	3	5	4	20	**3,00**
Highscores	0	1	1	0	0	2	5	5	6	20	**2,70**

[Tab.Stat.Ranking]



[Abb.Stat.Ranking]

Kommentare und Anregungen

Im letzten Abschnitt der Evaluierung hatten die Teilnehmenden die Möglichkeit, Textantworten auf die Fragen, welche Eigenschaften des Prototyps ihnen besonders gut bzw. weniger gut gefielen, zu geben. Außerdem konnten Anregungen und Verbesserungsvorschläge abgegeben werden.

Folgende Eigenschaften des Prototyps wurden hierbei besonders **positiv** hervorgehoben:

- Interessante Tasks und Erklärungen (4 Erwähnungen)
- Einfache(r) Handhabung / Aufbau / Menüführung (4 Erwähnungen)
- Ansprechendes Design / Optische Gestaltung (2 Erwähnungen)
- Idee (eine Erwähnung)
- Soundeffekte (eine Erwähnung)

Negativ bewertet wurden die folgenden Eigenschaften:

- Soundeffekte (drei Erwähnungen)
- Lesbarkeit der Texte (zwei Erwähnungen)
- Fehlende Landscape Mode-Unterstützung (eine Erwähnung)
- Design der Bilder (eine Erwähnung)
- Fehlende Bilder (eine Erwähnung)
- Suchtpotenzial (eine Erwähnung)

Zusätzlich wurden die folgenden **Anregungen** und Verbesserungsvorschläge gegeben:

- Automatisches Abspeichern von Highscores
- Auswählen von Antworten durch Klick auf die Antwortbilder
- Ändern des Levels ohne neuen Spielstart
- Zusätzliche Sounds und Farbschemata freischalten
- Weniger Text, mehr Interaktivität
- Toast-Messages (Texte bei „?“-Buttons) länger anzeigen

Textantworten

Das gefällt mir besonders gut:

- „Schnelle Ansprache, Schlichte Menüführung“ [10.08.2014 15:11]
- „Anschauliche Beispiele bzw. Tasks und nachvollziebare erklärungen machen die vielen ungreifbare informatik greifbarer“ [08.08.2014 19:22]
- „(...) eine nette APP um ein paar logische Sachen zu lösen...“ [07.08.2014 11:19]
- „die Biber sind süß, der Aufbau einfach genug für ein Kind“ [05.08.2014 22:53]
- „Die Rätsel“ [05.08.2014 17:53]
- „Nette tasks“ [05.08.2014 02:25]
- „Idee“ [04.08.2014 19:31]
- „Lustige Beispiele. Interessant, die Zusammenhänge zu erkennen.“ [04.08.2014 09:47]
- „Das Design ist sehr ansprechend und liebevoll gestaltet.“ [04.08.2014 08:44]
- „Die optische Gestaltung“ [04.08.2014 08:34]
- „Angenehme Sounds, einfache Handhabung“ [03.08.2014 22:05]
- „Schön, leichte Handhabung“ [01.08.2014 13:16]

Das gefällt mir weniger:

- „Soundeffekte“ [10.08.2014 15:11]
- „Vielleicht das Design einzelner Bilder aus den Aufgaben verbessern“ [08.08.2014 19:22]
- „Der Text ist sowohl in der Aufgabenstellung, als auch bei den Lerninhalten viel zu klein und viel zu lang.. .größere Schrift und eine Gliederung in z.b. Seiten zum umblättern wären übersichtlicher als der gesamte Text in einer "Wurst" . Bei verschiedenen Aufgaben haben bei mir Bilder gefehlt, weshalb ich die Aufgaben nicht lösen konnte.“ [05.08.2014 22:53]
- „Die Töne“ [05.08.2014 17:53]
- „Viel zu laute sounds!!“ [05.08.2014 02:25]
- „Nur hochkant nutzbar“ [04.08.2014 19:31]
- „Ich finde für mobile Geräte sind "Casual Games" die einzig vernünftige Variante (off the pocket - play 1 minute - in the pocket). Auf Bebras muss ich mich einlassen. Ob das auf Mobilgeräten so gut funktioniert. Wenn, dann eher auf Tablets als auf Handys.“ [04.08.2014 09:47]
- „Die Hinweistexte sind vor allem am Tablet ziemlich klein - sie sind zwar lesbar, könnten aber in einer größeren Schrift sein.“ [04.08.2014 08:44]
- „Würde süchtig machen“ [03.08.2014 22:05]

Das möchte ich dem Autor noch sagen:

- „Anregungen: Highscore automatisch abspeichern. Auf Antwort-Bilder klicken, um Antwort auszuwählen. Schwierigkeit ändern, ohne neues Spiel zu beginnen. XP - wozu? (Evtl andere Sounds/Farbschemata... freischalten?)“ [10.08.2014 15:11]
- „Persönlich finde ich noch, dass die Fragen und Beispiele gut sind, aber das ganze Paket immer noch zu sehr "Oberlehrerhaft" daherkommt. Nach dem Motto: "jetzt haben wir was gelernt, das war lustig, aber jetzt mal wieder ernst." Viel (trockener) Text und so. Ich würde mir von einer Lernapp mehr zack-zack und Interaktivität und Aha-Erlebnisse wünschen. Weil "strebern" muss ich sowieso zu hause am Schreibtisch. Lernapps sind (meiner Meinung nach) eine Ergänzung und können das Strebern nicht ersetzen - sie können den Schmerz nur lindern ;=) Daher glaube ich, dass Lernapps einfache "anders" sein sollten. Wie anders weiss ich leider auch nicht. Dennoch, viel Erfolg.“ [04.08.2014 09:47]
- „Der text wenn man auf ? klickt sollte noch etwas länger sichtbar sein...“ [01.08.2014 13:16]

Conclusio

Auch wenn die - im Verhältnis zur für Biber-Wettbewerbe üblichen Zielgruppe - relativ alten (der Großteil war zwischen 20 und 29 Jahre alt) und mit zwanzig Teilnehmenden relativ kleinen Testgruppe der Evaluierung keine repräsentative Stichprobe der zukünftigen Nutzer/innen des Prototyps darstellte, konnte diese doch einige qualitative Vorzüge sowie qualitative Mängel aufzeigen. Besonders positiv bewertet wurden hierbei das Design, die optische Gestaltung sowie die einfache Handhabung und Menüführung. Kritik fand sich vor allem bei der Lesbarkeit von Task- und Dialog-Texten und bei der Qualität und Lautstärke der Sound-Effekte.

Auch konnte gezeigt werden, dass insgesamt eine gute Integration der Gamedesign-Elemente in den Prototypen gelungen ist. Die Ausnahme stellten hierbei die Gamedesign-Elemente „Highscores“ und „Timer“ (Zeitbeschränkungen) dar, die in dieser Form wenig Anklang bei der Testgruppe fanden. In etwaigen Weiterentwicklungen des Prototyps müssen diese Konzepte also überdacht und überarbeitet werden. Erste Ideen hierfür sind Online-Highscores mit Anbindung an soziale Netzwerke, sodass es möglich wird, persönliche Erfolge weltweit mit anderen Nutzer/innen der App zu vergleichen sowie ein „Online-Wettkampfmodus auf Zeit“ anstatt der Timer. Dieser Modus könnte es zwei oder mehreren Nutzer/innen erlauben, Tasks gleichzeitig zu öffnen. Die für den Task vergebenen Punkte sind anschließend von einem Countdown abhängig: wer den Task also zuerst beantwortet erhält mehr Puntke.

In das Designkonzept sowie in den vorliegenden Prototypen konnte letztendlich aus Gründen der Komplexitätsminimierung und des programmiertechnischen Zeitaufwands nur eine Teilmenge aller im theoretischen Teil vorgestellten Gamedesign-Elemente einfließen. Deshalb wäre beispielsweise eine Weiterentwicklung des Prototyps mit einer durchgehenden Story, Avatar-Upgrades (beispielsweise Items zum „Hochrüsten“ der Biber-Avatare) und Elementen der Kooperation durchaus überlegenswert.

Der vorliegende Prototyp stellt *eine mögliche Beantwortung* der wissenschaftlichen Fragestellung dar, wie motivierende und beim Erlangen der Lernziele unterstützende Gamedesign-Elemente in eine mobile Applikation zum Erlernen von Konzepten der Informatik integriert werden können. Selbstverständlich kann diese Beantwortung nur eine von vielen sein.

Zusammenfassung

Ziel dieser Arbeit war die Beantwortung der folgenden wissenschaftlichen Fragestellung:

„Wie können motivierende und beim Erlangen der Lernziele unterstützende Gamedesign-Elemente in eine mobile Applikation zum Erlernen von Konzepten der Informatik integriert werden?“

Zur Erreichung dieses Ziels kamen folgende Methoden zum Einsatz:

- Umfassende *Literaturrecherche* zum lerntheoretischen Hintergrund von Gamification und zur Identifikation motivierender Gamedesign-Elemente, die beim Erlangen von Lernzielen unterstützen können
- Erstellung eines *Designkonzepts* basierend auf dieser Recherche
- Entwicklung eines *Prototyps* in Form einer mobilen Applikation für das Betriebssystem Android
- *Summative Evaluation* mittels Online-Fragenkatalog zur Bewertung von Usability und Qualität der Integration der Gamedesign-Elemente in den Prototypen

Die vorliegende Masterarbeit gliedert sich in folgende Abschnitte:

Einleitung
In der Einleitung werden die Problemstellung, wissenschaftliche Fragestellung und geplante Methodik zur Beantwortung derselben erläutert. Weiters werden ein Überblick über den State-of-the-Art des Themenbereichs Gamification und der spielerischen Vermittlung von Konzepten der Informatik gegeben.

Bebras - Der Biber der Informatik
Dieser Abschnitt stellt die internationale Initiative Bebras vor, die in Österreich unter dem Namen „Der Biber der Informatik“ von der Österreichischen Computer Gesellschaft (OCG) getragen wird. Es werden ein Überblick über die Klassifikation und Kriterien von Tasks gegeben, die bei Biber-Wettbewerben zum Einsatz kommen und entsprechende Beispiele gegeben.

Das mobile Betriebssystem Android
Es wird ein kurzer Überblick über die Geschichte des mobilen Betriebssystems Android und über dessen technische Hintergründe und Architektur gegeben. Weiters finden das Android Software Development Kit (Android SDK) und das Android Developer Tools Plugin für Eclipse (ADT) Erwähnung, mit welchen der Prototyp implementiert wurde.

Theoretischer Teil

Im theoretischen Teil wird zunächst ein Überblick über lernpsychologische Theorien und Konzepte (u.a. extrinsische und intrinsische Motivation, Flow, ARCS-Modell, Scaffolding) gegeben, die für den Einsatz von Gamification von Bedeutung sind. Im Anschluss daran werden unterschiedliche Gamedesign-Elemente identifiziert, die für die Umsetzung von Gamification-Konzepten vonnöten sind (u.a. Levels, Regeln, Ziele, Belohnungsstrukturen, Feedback, Ästhetik).

Designkonzept

Dieser Abschnitt erläutert das Konzept der geplanten Integration von im theoretischen Teil identifizierten Gamedesign-Elementen in den Prototypen. Jedes Element wird ausführlich erklärt. Eine Concept Map stellt das Zusammenspiel der einzelnen Elemente im Gesamtkonzept dar. Weiters geben Mock-Ups einen Überblick über das geplante User Interface Design des Prototyps wieder.

Prototyp

Es werden technische Hintergundinformationen zur Entwicklung des Prototyps und ein Workflow zur Erweiterung desselben mit zusätzlichen Tasks gegeben. Außerdem wird die Benutzung des Prototyps mit vielen Screenshots detailliert dokumentiert.

Evaluierung

Der letzte Abschnitt erhält die Ergebnisse und Rückschlüsse der summativen Evaluation. Diese wurde mit Hilfe der Website https://de.surveymonkey.com zwischen 1. und 10. August 2014 durchgeführt. Ziel der Evaluierung war es, qualitative Mängel bzw. Fehler im Prototypen zu identifizieren sowie die Beantwortung der Frage, ob eine Integration der Gamedesign-Elemente in den Prototypen gelungen ist.

Quellen

[Bebras: History] Bebras Contest - History: http://bebras.org/history/ - Zugiff am 17.12.2013

[Biber-Aufgaben 2012] Biber der Informatik - Aufgaben 2012 - *Herausgeber:* Gerald Futschek, *Mitwirkende:* Hans-Werner Hein, Ulrich Kiesmüller, Wolfgang Pohl, Kirsten Schlüter, Michael Weigend, Ivo Blöchliger, Gerald Futschek, Christian Datzko, Bernd Kurzmann, Jürgen Frühwirth, Barbara Müllner, Jacqueline Peter. http://www.ocg.at/sites/ocg.at/files/medien/pdfs/Biber-Aufgaben2012-mitLoesungen-AT-web.pdf - Zugriff am 20.12.2013

[Biber-Aufgaben 2013] Biber der Informatik 2013 - Aufgaben mit Lösungen - *Herausgeber:* Gerald Futschek, *Mitwirkende:* Gerald Futschek, Jürgen Frühwirth, Bernd Kurzmann, Barbara Müllner, Peter Garscha, Piotr Michalski, Elisabeth Maier-Gabriel, Andrea Adamoli, Ivo Blöchliger, Brice Canvel, Christian Datzko, Hanspeter Erni, Beate Kuhnt, Jacqueline Peter, Simon Reichmuth, Marie-Thérèse Rey, Lena Theiler, Beat Trachsler, Hans-Werner Hein, Wolfgang Pohl. http://www.ocg.at/sites/ocg.at/files/medien/pdfs/Biber2013_Aufgaben-Loesungen.pdf - Zugriff am 17.12.2013

[Bozarth 2011] Bozarth, J.: „An Interview with Sebastian Deterding“ (2011), http://elearnmag.acm.org/archive.cfm?aid=2008214, *eLearn Magazine, Volume 2011 Issue 7, July 2011, Article No. 5*, ACM, New York, NY

[Csíkszentmihályi 1990] Csíkszentmihályi, Mihály.: „Flow: the psychology of optimal experience“ (1990), HarperCollins Publishers, New York, NY, HarperPerennial Modern Classics Edition 2008, ISBN 978-0-06-133920-2

[Dagienė 2008] Dagienė, V., Futschek, G.: „Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks“ (2008), In: R. T. Mittermeir, M. M. Syslo (Eds.), *Lect. Notes in Computer Science, 5090, Informatics Education – Supporting Computational Thinking*, pp. 19–30, Springer-Verlag Berlin Heidelberg 2008

[Deterding 2011|1] Deterding, S. et al.: „Gamification: Using game-design elements in non-gaming contexts“ (2011), *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, pp. 2425-2428, ACM, New York, NY

[Deterding 2011|2] Deterding, S. et al.: „From game design elements to gamefulness: defining *gamification*“ (2011), *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11)*, pp. 9-15, ACM, New York, NY

[Fitz-Walter 2011] Fitz-Walter, Z. et al.: “Orientation passport: using gamification to engage university students” (2011), *Proceedings of the 23rd Australian Computer-Human Interaction Conference (OzCHI),* pp. 122-125, ACM, New York, NY

[Futschek 2009] Futschek, G., Dagienė, V.: „A Contest on Informatics and Computer Fluency Attracts School Students to Learn Basic Technology Concepts“ (2009), *9th WCCE IFIP World Conference on Computers in Education-Education and Technology for a Better World*, pp.1-9

[Gee 2007] Gee, J.P.: „What video games have to teach us about learning and literacy“ (2007), Palgrave Macmillan, New York, NY, I. Title, ISBN 978-1-4039-8453-1

[Haberman 2011] Haberman, B., Cohen, A., Dagienė, V.: „The Beaver Contest: Attracting Youngsters to Study Computing“ (2011, June), *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (ITiCSE),* pp. 378-378, ACM, New York, NY

[Huizinga 1938] Huizinga, J.: „Homo Ludens. Vom Ursprung der Kultur im Spiel" (1938), Rowohlt Taschenbuch Verlag, Reinbek bei Hamburg, 22. Auflage April 2011, ISBN 978-3499554353

[Kapp 2012] Kapp, K.: „The gamification of learning and instruction: game-based methods and strategies for training and education" (2012), Pfeiffer, San Francisco, CA, I. Title, ISBN 978-1-118-09634-5

[Klante 1999] Klante, P. et al.: „Evaluation von Web-basierter Lernsoftware. Erweiterung der Usability-Bewertungsmethoden für arbeitsbezogene Software auf Lernsoftware" (1999), http://www-cg-hci.informatik.uni-oldenburg.de/resources/UniOl1999-Klante-DA-Evaluation%20webbasierter%20Lernsoftware.pdf - Zugriff am 10.06.2013

[Meerbaum 2010] Meerbaum-Salant, O. et al.: „Learning computer science concepts with scratch" (2010), *Proceedings of the Sixth international workshop on Computing education research (ICER),* pp. 69-76, ACM, New York, NY

[O'Rourke 2010] O'Rourke, J. et al.: „Learning computer science concepts using iPhone applications" (2010), *Journal of Computing Sciences in Colleges, Volume 25 Issue 6, June 2010,* pp. 121- 128, ACM, New York, NY

[Prensky 2003] Prensky, M.: "Digital game-based learning" (2003), *Computers in Entertainment (CIE) Journal, Volume 1, Number 1*, p. 21, ACM, New York, NY

[Raymer 2011] Raymer, R.: "Gamification – Using Game Mechanics to Enhance E-Learning" (2011), *Elearn Magazine, 2011 (9)*, p.3, ACM, New York, NY

[Reeves 2009] Reeves, B. and Read, J.L.: „Total Engagement: Using Games and Virtual Worlds to Change the Way People Work and Businesses Compete" (2009), Harvard Business School Press, Boston, MA

[Ward 2010] Ward, B. et al.: „Teaching computer science concepts in Scratch and Alice" (2010), *Journal of Computing Sciences in Colleges, Volume 26 Issue 2, December 2010,* pp. 173- 180, ACM, New York, NY

[Werner 2012] Werner, L. et al.: „Children learning computer science concepts via Alice game-programming" (2012), *Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE),* pp. 427-432, ACM, New York, NY

[WP.de:Android 2013] Wikipedia: Android (Betriebssystem)
http://de.wikipedia.org/w/index.php?title=Android_(Betriebssystem)&oldid=125431416 - Permalink, Zugriff am 20.12.2013

[WP.en:Android 2014] Wikipedia: Android (operating system)

http://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=586494444 - Permalink, Zugriff am 01.01.2014

[Zichermann 2011] Zichermann, G. and Cunningham, C.: „Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps" (2011), O'Reilly Media, Inc., ISBN 978-1449397678

Abbildungen

[Abb.Android.Arch] Android System Architecture
http://en.wikipedia.org/wiki/File:Android-System-Architecture.svg - Zugriff am 20.12.2013
Bild-Autor: Smieh, Erstellungsdatum: 29.06.2012

[Abb.Android.Home] Home screen of Android 4.4 shortly taken after installation on Nexus 4.
http://en.wikipedia.org/wiki/File:Android_4.4_with_stock_launcher.png - Zugriff am 20.12.2013
Bild-Autor: Himanis Das, Software-Autor: Google Inc., Erstellungsdatum: 15.12.2013

[Abb.Bebras.*] Screenshots der App, erstellt mit Google Nexus 4 (Android 4.4) am 04.08.2014

[Abb.Biber.blau] http://www.clker.com/cliparts/3/E/X/b/T/a/blue-beaver-md.png - Public Domain, Zugriff am 06.03.2014

[Abb.Biber.braun] http://www.clker.com/cliparts/8/7/7/f/12161376693358641351emmling_Cartoon_beaver.svg.med.png - Public Domain, Zugriff am 06.03.2014

[Abb.Biber.gelb] http://www.clker.com/cliparts/I/8/t/2/S/Z/yellow-beaver-md.png - Public Domain, Zugriff am 06.03.2014

[Abb.Biber.grau] selbst erstellt auf Basis von [Abb.Biber.pink]

[Abb.Biber.grün] http://www.clker.com/cliparts/p/D/L/r/K/E/green-beaver-md.png - Public Domain, Zugriff am 06.03.2014

[Abb.Biber.pink] http://www.clker.com/cliparts/c/j/b/i/k/L/beaver-md.png - Public Domain, Zugriff am 06.03.2014

[Abb.Biber.rot] http://www.clker.com/cliparts/T/x/O/K/R/f/red-beaver-md.png - Public Domain, Zugriff am 06.03.2014

[Abb.Biber.violett] http://www.clker.com/clipart-beaver-4.html - Public Domain, Zugriff am 06.03.2014

[Abb.BiberTask1] entnommen aus [Biber-Aufgaben 2012]

[Abb.BiberTask2] entnommen aus [Biber-Aufgaben 2012]

[Abb.Char.Upgr] entnommen aus [Raymer 2011] (zur Verfügung gestellt von: Mike Henry, Big Menace Industries)

[Abb.Eclipse] Screenshot: Eclipse mit integriertem ADT, erstellt am 23.07.2014

[Abb.Flow.Channel] entnommen aus [Raymer 2011]

[Abb.Flow.Goals] entnommen aus [Raymer 2011]

[Abb.GDE.ConceptMap] Concept Map der Gamedesign-Elemente, selbst erstellt mit XMind am 10.03.2014

[Abb.Goals.linear] entnommen aus [Raymer 2011]

[Abb.Goals.nonlinear] entnommen aus [Raymer 2011]

[Abb.Klassendiagramm] erstellt mit Lucidchart (https://www.lucidchart.com/) am 25.07.2014

[Abb.Mockup.*] erstellt mit Justinmind Prototyper am 21., 23. und 24.03.2014, Cliparts: Public Domain (http://www.clker.com)

[Abb.Module] erstellt mit Evolus Pencil am 04.03.2014

[Abb.Module.*] erstellt mit Evolus Pencil am 04.03.2014

[Abb.Stat.*] Diagramme zu den Auswertungstabellen der Online-Umfrage [Tab.Stat.*], erstellt mit Apple Numbers (iWork) am 11.08.2014

[Abb.ZPD] Zone of Proximal Development (Lev Vygotsky) http://www.innovativelearning.com/educational_psychology/development/zone-proximal-development.png - Zugriff am 17.01.2014

Tabellen

[Tab.BC.Schema] BC-Schema (Verteilung der Bebras-Coins), selbst konzipiert

[Tab.DrawableNames] Konvention zur Benennung von Drawables, eingeführt zur leichteren Erweiterbarkeit der App

[Tab.GDE] Kategorisierung von Gamedesign-Elementen, entnommen aus [Deterding 2011|2] und ins Deutsche übersetzt

[Tab.SBT] Struktur der Bebras-Tasks, basierend auf [Biber-Aufgaben 2012]

[Tab.Stat.*] Auswertungstabellen des Online-Fragenkatalogs, Evaluierung durchgeführt mit https://de.surveymonkey.com von 01.-10.08.2014

[Tab.XP.Schema] XP-Schema (Verteilung der Erfahrungspunkte), selbst konzipiert