Hamm-Lippstadt University of Applied Sciences

Bachelor Thesis

Game UI Prototyping with Unreal Engine 4

Author:
Hans Ferchland

Supervisor:
Prof. Dr. Christian Sturm
Examiner:

A thesis submitted in fulfilment of the requirements
for the degree of Bachelor of Science (B.Sc.)

February 2015

Declaration of Authorship

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Literatur und Hilfsmittel angefertigt habe. Wörtlich übernommene Sätze und Satzteile sind als Zitate belegt, andere Anlehnungen hinsichtlich Aussage und Umfang unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen und ist auch noch nicht veröffenticht.

Unterschrift: _____

Ort & Datum: _____

"It [Darwin's theory of evolution] was a concept of such stunning simplicity, but it gave rise, naturally, to all of the infinite and baffling complexity of life. The awe it inspired in me made the awe that people talk about in respect of religious experience seem, frankly, silly beside it. I'd take the awe of understanding over the awe of ignorance any day."

Douglas Adams, The Salmon of Doubt

HAMM-LIPPSTADT UNIVERSITY OF APPLIED SCIENCES

Abstract

Computational Visualistics and Design

Bachelor of Science (B.Sc.)

Game UI Prototyping with Unreal Engine 4

by Hans Ferchland

The purpose of this thesis was to implement a prototyping process for early game user interface (UI) development. The process evaluation and evaluation of the Unreal Engine 4 (UE4) was performed in a case study, implementing a role-playing game (RPG) UI for a character inventory.

Extensive research, related to prototyping techniques, game development, immersion and flow, was conducted; to sustain the case study project, game user interface design, heuristic evaluation, usability and user experience, were additionally reviewed.

The prototyping process defined, combines techniques from usability evaluation, software engineering and game development and has proven to deliver sufficient results. The current state of the prototype, the result of the case study, is far from being final; but not all requirements, identified in the study, were realized for the UI in the UE4 prototype, and remain to be implemented in the future.

Guldbrandsen and Storstein compiled a evaluation framework for the predecessor of the UE4, the Unreal Development Kit (UDK), it was applied in a qualitative discussion (Guldbrandsen and Storstein, 2010). The UE4 is a feature heavy state-of-the-art game engine; a large framework that offers flexible and fast iteration for game UI. It is developed continuously, providing new and improved features. While updates are generally positive, they also form additional requirements and risks for a product, the downside of “bleeding edge” development.

The author has used the UE4 for over a year, and is experienced with the framework and its general functionalities. Thus, the thesis grants more an expert point of view; the excellent support, documentation and community of the UE4 may compensate lack in experience for novice users.

Keywords: prototyping, unreal engine, user interface, usability evaluation, user-centred design, game development

Acknowledgements

First of all, i would like to thank my family for supporting and encouraging me and my education all my life. I thank my friends for their open minds, ears and advices on my work, life, the universe and everything. Thank you Epic Games, for an epic tool for comfortable game development. Many thanks to all the past and present developers from the "Caede" project team, lets finish this! Last, but not least, special thanks to all my teachers, professors and to my supervisor.

Contents

List of Figures

List of Tables

Abbreviations

AAA	Triple **A**
AS	**A**ction **S**cript
UI	**U**ser **I**nterface
UX	**U**ser E**X**perience
GUI	**G**raphical **U**ser **I**nterface
RPG	**R**ole **P**laying **G**ame
NPC	**N**on **P**layer **C**haracter
HCI	**H**uman-**C**omputer **I**nteraction
UCD	**U**ser-**C**entered **D**esign
QA	**Q**uality **A**ssurance
CW	**C**ognitive **W**alkthrough
HEP	**H**euristics **E**valuation of **P**layability
UE4	**U**nreal **E**ngine **4**
UMG	**U**nreal **M**otion **G**raphics
UEE	**U**sability **E**xpert **E**valuation
VE	**V**irtual **E**nvironment
FPS	**F**irst **P**erson **S**hooter
HUD	**H**ead-**U**p **D**isplay
CS	Adobe **C**reative **S**uite
API	**A**pplication **P**rogramming **I**nterface
VCS	**V**ersion **C**ontrol **S**ystem
OS	**O**perating **S**ystem

Dedicated to the free access to knowledge and education.

Chapter 1

Introduction

"For a moment, nothing happened. Then, after a second or so, nothing continued to happen."

Douglas Adams – The Hitchhiker's Guide to the Galaxy

This chapter will describe in which context the thesis is written. The first section outlines the development context. The second section declares my personal motivation and the more general motivation for the subject; and reasons, why exploring game development processes can be valuable. Then the projects goals and steps how to achieve it are described. It follows a description of the development methods, workflow and tools used. Furthermore the chapter covers the definition of research questions posed by the author.

1.1 Development Context

The project and the development of the game described in the thesis began about a year ago, in September 2013. It started as a independent group of people who already had the core game idea in mind; and began to evolve with the opportunity to be a Unreal Engine 4 (UE4) beta tester; which transitioned into an active independent developer team with contributors from Austria, Germany and the United Kingdom (Thrown Together Studios, 2015). The current project contributors are listed in appendix A but have varied throughout the projects progress. Most of the core team has a free license for the UE4, as a result of the beta participation. The communication and collaboration is mostly done with Microsoft Skype (Microsoft Corporation, 2014), Trello (Trello Inc., 2015) and Google Drive (Google Inc., 2014). The UE4 project files are stored on a rented root-server by the Host Europe GmbH (Host Europe GmbH, 2014) and access is restricted to the

project members only. Nevertheless, the thesis-related results will be published via GitHub (GitHub Inc., 2014); links to the repositories can be found in appendix A. The game's title was also improved from the working title "Kill the King" into "Caede" (Thrown Together Studios, 2015); a realistic role playing-game set in the medieval times. The core features are a complex fighting system, realistic NPC behaviour, beautiful environments, non-linear quests and dialogues, survival aspects, much humor, a consistent and immersive world and story. Additionally, a detailed game design document and related documents can be found in appendix B. Because of the more or less loose context of a independent developer team, the features are meant to change over time due to iterative processes. There is of course a rough date by which the game will be released: the end of 2015. This is a rather short development time, compared to some AAA games (Stapleton, 2011; Berghammer, 2007) also in number of developers per project.

FIGURE 1.1: Caede Mockup

1.2 Personal Motivation

For the author of this thesis the motivation for the project is comes mostly from the participation in the indy developer-team. The overall personal goal is, in fact, the finishing of the game "Caede", with a high quality standard, till end of 2015. The research contributed in this thesis is of great interest to the author. Specifically, the game development processes: iterative part of it, reflected in prototyping and testing; the UI and game design processes, and the user experience as a result of immersion and game flow, are under focus. From an internship at the Ubisoft Bluebyte in Studio, Mainz (Blue Byte GmbH, 2014) as UI- and Gameplay-programmer, there is some insight in the AAA game- and especially UI-development process on the author's side. That may help in the matter of finding needs

and improvements that can also be used by bigger game studios for their purposes. The Unreal Engine 4 development itself, is quite open for academical use (Epic Games Inc., 2014b; Epic Games Inc., 2014c); in addition to the contribution to the scientific community this thesis may also result in contribution to the Unreal Engine Wiki-pages (Epic Games Inc., 2014e) or the Answer-Hub (Epic Games Inc., 2015h). The motivation for this might also come from the Unreal Engine community as well as the motivation from Epic Games (Epic Games Inc., 2015a): giving as many people as possible access to knowledge and technology. The user interface design- and the iterative prototyping-process is the major interest of this thesis and has great relevance to the author. Developing such a process in detail and measuring it with the UE4 regarding some additionally developed criteria, may give evidence of the usage of the UE4 as a UI prototyping tool. These processes and criteria can be used with other tools in evaluation processes of both games and their development tools.

There are many reasons for further exploring the subject of game development (Pagulayan et al., 2003; Guldbrandsen and Storstein, 2010). Gaming itself has not only become popular recently, but has widely arrived in the mainstream society (Smithsonian American Art Museum, 2012; Plunkett, 2012). As the Software Engineering discipline evolved over time, it has developed methods, processes and criteria for application of engineering methods to software-design, -development and -maintenance. Those can be combined with methods, heuristics known from other disciplines like user experience, user interface design, prototyping and other agile processes to significantly enhance the outcome of a project. To find a fair method-combination, in order to reveal new approaches or even better solutions for existing problems in game development, is a central point of the projects research and development. There are several other important questions that may be of general interest: Why do game prototyping and why do UI prototyping, how to support creative processes for game UI, can the UI design process be done faster, and how to respect the continuous development of tools, features and state-of-the-art software and technology in the game UI design processes?

1.3 Project Goal

The goal will be to have a process for game UI prototyping and later, a refined prototyping process for the UE4 (1). This process is used in the project context, to generate significant research results. The Unreal Engine as tool in the project context and as a game engine, will be evaluated according to a second process. This process will be a general pattern to evaluate a game engine or similar tool for UI prototyping.

1.4 Development Method

In order to stay flexible during development, the methods and processes are mostly agile. This is also required by most of the prototyping processes and methods that are used in this thesis, since prototyping is highly iterative and synergizes well with agile methods. The task planning is managed with a digital kanban board tool *Trello* which is also used for the whole management of the team's tasks. All the tools used in the projects context are listed and described below in section 1.6. The progress will be reflected on the kanban board and the resulting artefacts are managed with the *Git* version control system (VCS) and a self-hosted cloud server, plus Googles (Google Inc., 2015) cloud service *Google Drive* (Google Inc., 2014). Creating 3D models and animations will not be part of the author's work in the thesis context, as they already exists in required amount and quality.

Early design scribbles will be created traditionally with pen-and-paper sketching, to outline general ideas and concepts. Digital artefacts will be produced with Adobes (Adobe Systems Incorporated, 2015) Creative Suite (CS), especially with *InDesign*, *Photoshop* or *Illustrator* for design related artefacts; *Visual Studio 2013* for code- and binary-files and the Unreal Engine 4 (see section 1.6) for the combination of both, which will reveal behaviour and functionality of design approaches. The development process will borrow techniques from (rapid) prototyping and game UI design for best purpose of the project goals.

1.5 Development Workflow

At the beginning of the thesis, the focus obviously lies on research, studying existing approaches and existing user interfaces that relate to the project. After this stage, the development of UI concepts, including layout, interaction design, colours, themes, fonts, icons, etc. and their evaluation, is the next major part. This stage is the iteration for the visual design, and continues as long as a significant progress is made, until the implementation and prototyping within the Unreal Engine will begin. Those two stages cannot be separated clearly and are not necessarily sequential. In fact, the interaction of games may require to combine and mix visual and functional prototypes, to investigate functionality, visual appeal and user satisfaction at the same time, and not only for one at a time. The prototyping process that will evolve during the project needs to fit into the described development methods (see section 1.4) and is therefore dependent on those.

1.6 Tools

This section covers the tools that are used during the project. A perspective on prototyping tools can be found in chapter 4; all of the following tools are used in chapter 6.

Unreal Engine 4

A game engine by Epic Games (Epic Games Inc., 2015a), the fourth iteration since 1991 and obviously the subject of the thesis. Most of the functional and interactive prototyping will be done with the UE4, using Slate and *Unreal Motion Graphics* (UMG) in particular for graphical user interface (GUI) implementations. The engine also provides fast and accessible options for exporting a prototype (cook & package), in order to test in on various platforms (Epic Games Inc., 2015d), with much ease.

Due to the beta participation and further work with the engine, the author has some experience with commonly used workflows in game-play programming, artificial intelligence, animations and blueprints in general. Regarding the user interface design and programming, the author has no practical experience with Slate and UMG. This may require a preceding learning phase, before the actual implementation can start, and needs to be regarded, while approaching the goals and interpreting the results of the project.

Content created in the UE4 will be referred to as "assets", this can be a game level, materials, cinematic sequences, blueprint scripts, AI navigation meshes, meshes, skeletons, animations, UI widgets or textures; they cover inherent functionality and/or content (Epic Games Inc., 2015i).

Visual Studio 2013 Ultimate

The Unreal Engine requires *Visual Studio* or *XCode* depending on the operating system (OS) for best experience and the entire feature-set. The *Linux* version is still "NOT production worthy yet as there are many issues that need to be ironed out." (Epic Games Inc., 2015c). In the project context, the combination of *Microsoft Windows* (Microsoft Corporation, 2015a) and *Visual Studio*, is used as it has proven to be the best experience for UE4 development. Features like the "Hot Reload" functionality, for compiling and debugging C++ game code during the editor- or even game-runtime, are an immense benefit for rapid game prototyping (Epic Games Inc., 2015j). The *Visual Studio* IDE has been used by the author in several previous projects and is thus known to a professional extend.

Adobe Creative Suite

The *Creative Suite* (CS) from Adobe (Adobe Systems Incorporated, 2015) is a large collection of tools to create media content. This tools are widely used by all kinds of designers and also in game development processes. The author has expert knowledge of several of the programs contained by the CS, and has used them in preceding projects. They take part in the projects design process, e.g. to create documents, mock-ups or vector and raster images.

Perforce

This is a software for version control – a version control system (VCS) – for files and folders (Perforce Software Inc., 2015). Perforce supports concepts for copy/lock, modify and merge, branches and labels and provides a balance of functionality and ease of use. Being popular in the games industry and also the first VCS supported by the UE4, it was used since then, to manage all game related project-content. This includes the visual assets, blueprints, maps, source-code and the project configuration; while the source-code is additionally versioned with Git. The perforce server offers access for the project members and administrative access for the author.

Git

Another VCS is Git (Git Project Team, 2015a), its design is distributed, where each user has a local copy of the code-repository and there exist no local and remote repositories. Each client can act as a server on their own; nevertheless there is a master repository-copy, also stored on, the previously mentioned, rented server (see section 1.1). The complete UE4 source-code is available on GitHub (GitHub Inc., 2014; Epic Games Inc., 2015b), provided, you have a valid subscription. Direct access to the code-base is required for some plug-ins or engine customization, but not necessary for common usage.

Trello

As already stated, Trello (Trello Inc., 2015) is used for task planning for the team. This tool is basically a digital kanban board; supporting different categories for tasks, assigning people and dates to them, labelling tasks, and many more useful functions for agile project management. The premium “Gold” version offers some more features, but most of them are only visual. The free version allows creating private organizations, which can collect multiple boards, e.g. to separate game design from programming tasks.

Skype

This is a instant messenger from *Microsoft* (Microsoft Corporation, 2015a) with telecommunications functionality, offering groups-chats, group calls and file exchange, to work more collaboratively (Microsoft Corporation, 2014). It is available for desktop, mobile and web platforms; and used by the team for daily discussions and meetings. Due to the fact, that all members are distributed over various locations, it is the major tool for communication.

1.7 Research Questions

The questions stated in this section are based on the context displayed in chapter 1 and target the goals discussed in section 1.3. They are the golden thread of the thesis and guide the thoughts on argumentation, implementation and evaluation. The most important field to explore is the prototyping process, especially for UI prototyping in games. Prototyping itself is a major part that needs deeper exploration for UI and games. That is important for video game companies as well as smaller independent developer teams to lower the development cost and reduce production time as also stated by Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010; Forbes.com, LLC^{TM}, 2008). This is most lately more a problem of choice than of quality or features because of the rising amount of low-cost but state-of-the-art game engines (Epic Games Inc., 2014d; Crytek GmbH, 2014; Unity Technologies, 2014a).

From this position the author raises the first research question and its sub-questions:

RQ1: What formal game UI prototyping processes exist?

- **RQ1.1:** What formal existing UI prototyping processes exist?
- **RQ1.2:** Which other formal prototyping processes exist?
- **RQ1.3:** Are those processes applicable to game UI prototyping?

The focus of the project, as mentioned in section 1.1, is the Unreal Engine 4 from *Epic Games, Inc.* (Epic Games Inc., 2015a; Epic Games Inc., 2014d). The questions that have been listed with **RQ1** will be explored as preparation for the UI prototyping process. The evaluation of the specific parts - formal features - that can be used for UI prototyping, especially with UE4, is the second major field to explore. The aim therefore is to gather relevant factors that can be used for evaluation of a game UI prototyping tool. The result should be, as announced in section 1.2, to have a process to evaluate a game engine in the aspect of UI prototyping. It may be interesting for video game companies, other

researchers and independent game developers, to have a formal process for their evaluation needs researched in this thesis.

The second set of questions relates thus:

RQ2: How does the Unreal Engine 4 evaluate as a UI prototyping tool?

RQ2.1: What is a common set of formal features for a UI prototyping tool?

RQ2.2: What needs to be evaluated in the matter of UI prototyping?

Chapter 2

Related Work

"Reality is frequently inaccurate."

Douglas Adams – The Restaurant at the End of the Universe

In this chapter an overview on related work and common research methodologies is given. The software engineering side and the user interface design side of research will be taken into account. It also covers what methods, techniques and theories are used in *thesis-related projects* and will present an outlook on what was considered and used during the *project in context of the thesis*.

2.1 Software Engineering

There exist various paradigms in software engineering as well as other disciplines (Basili, 1996). The ones displayed here are the analytic and the experimental paradigm. After the more established research paradigms from Basili, some more recent empirical methods from Easterbrook, Singer, Storey and Damian are displayed.

2.1.1 Analytic Paradigm

Basili describes the analytic paradigm as to "involve proposing a set of axioms, developing a theory, deriving results and, if possible, verifying the results with empirical observation" (see Basili, 1996, p. 2). Furthermore he states it as the mathematical method, a deductive analytical model which rather needs a experimental set-up for statistics and is more an analytical framework for model creation and understanding their limits. The software

product – its programs and various forms of modules – as mathematical objects, their relation and its analysis suffers the paradigm. (Basili, 1993).

2.1.2 Experimental Paradigm

The tree models to discuss "involve an experimental design, observation, data collection and validation on the process or product being studied" (see Basili, 1996, p. 3). Basili also defines the following terms in this context:

- **Hypothesis** – make assumptions in order to test its logical or empirical consequences
- **Study** – discover unknown, testing hypothesis, includes experimental, empirical and qualitative studies
- **Experiment** – study where the researcher has control over the studies experimental conditions and aspects of its independent variables
- **Controlled Experiment** – experimental conditions are randomly assigned to subjects, change an independent variable, thread all subjects the same for all other variables

All three models have in common that they propose, measure and analyse a model, but they differ in the models origin and the response to the analysis made.

Scientific According to Basili, the scientific method used in the physics discipline is best for understanding processes, product, people or environment in the software engineering discipline. We do "observe the world, propose a model or a theory of behavior, measure and analyze, validate hypotheses of the model or theory (or invalidate them), and repeat the procedure evolving our knowledge base" (see Basili, 1996, p. 3). The goal is to build a model in attempt to explain a phenomena based on observation of the world and to evaluate the models degree of representation. The scientific method can be used as an approach for evolutionary or revolutionary research if two variations of the same scientific approach are combined.

Evolutionary This method has is influences from manufacturing; the evolutionary attempt is to "observe existing solutions, propose better solutions, build/develop, measure and analyze, and repeat the process until no more improvements appear possible" (see Basili, 1996, p. 3). This approach requires a model to improve or build upon and the

decisive element lays in the careful analysis and measurement. For example one could investigate method-improvements in software development or compare two version of tools i regard of certain criteria and characteristics (Basili, 1996).

Revolutionary If there are no existing models to build upon, or the existing model relates rather distant (e.g. from another discipline), then the experimentation is called revolutionary. We begin with the proposal of our new model, then start "developing statistical/qualitative methods, applying the model to case studies, measuring and analyzing, validating the model and repeating the procedure" (see Basili, 1996, p. 3). The proposition of a new tool, process or method for software development or involved disciplines are examples for a revolutionary approach. As for the evolutionary method, the crucial elements are again the measurement and analysis.

2.1.3 Empirical Methods

Because software engineering is multi-disciplinary, it includes social and technical aspects that need exploration (Easterbrook et al., 2008). That covers human activities, of individuals and between them in teams or organization, and most important in this thesis it will also include the users of the software. According to Easterbrook et al. there are five empirical methods relevant in software engineering:

- Controlled Experiments (including Quasi-Experiments)
- Case Studies (both exploratory and confirmatory)
- Survey Research
- Ethnographies
- Action Research

Between those one should choose depending on precious project-relevant aspects: resources, access to test-subjects, the ability to change the variables in focus and the skill of the researcher itself. Beside the classification of research questions from Meltzoff (Meltzoff, 1998; Easterbrook et al., 2008), Easterbrook et al. also explains the importance of consideration of what one will accept as empirical truth, displays the role of theory-building and gives an interesting focus on validation and practical usage.

2.2 Human-Computer Interaction

It is known that HCI research is close to games and game design, but Johnson (Johnson and Wiles, 2003) points out that there is need for "bridging the gap between these two areas." (see Johnson and Wiles, 2003, p. 4). This thoughts were considered during the research in order to get a large overview on related research; regarding games, user interfaces, user experience, prototyping, evaluation and interviewing. While HCI is getting more popular in the gaming industry and games related research, some interesting approaches exist and will be discussed and considered further in this chapter (Desurvire and Wiberg, 2009; Desurvire et al., 2007). Other applications where found in Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010) in their prototyping process with the *Unreal Development Kit*, which is "adapted from traditional HCI." (see Guldbrandsen and Storstein, 2010, p. 57). Although some HCI methods may be useful, HCI related methods for heuristic evaluation are rejected by game- and usability-experts as " too broad " (see Fricker, 2012, p. 9), according to Fricker. The user experience (UX) domain in HCI is amongst the useful ones in aspect of evaluation, there is research on this field by Takatalo et al. (Takatalo et al., 2010), creating a framework for empirical analysis of UX in games. The iterative processes and concepts known from HCI also work for games or virtual environments (VE) and their importance is underlined by Bowman (see Bowman et al., 2001, p. 19) and Hix et al. (see Hix et al., 1999, p. 1). This chapter and section will only give an overview on the large related fields of games, software engineering and HCI; more onto prototyping and usability evaluation is discussed in depth in the chapters 3, 5 and 6.

2.2.1 Heuristic Evaluation

Heuristics have proven useful in user interface, user experience design and game design likewise (Federoff, 2002; Nielsen, 1994a; Malone, 1982; Fricker, 2012) and some broader view on this will be part of chapter 5. In her MA Thesis, Federoff examines and collects several heuristics and usability guidelines used in game or software development (Federoff, 2002; Nielsen, 1994a; Malone, 1982). Those can be used to evaluate the usability of games, and consequently used in the development process, they may support producing consistently successful games (see Fricker, 2012, p. 8-9). Federoff identifies 30 heuristics for games, ten of them are interface related, four relate to game mechanics and the other 16 to the game play. Nielsens "Ten Usability Heuristics" (Nielsen, 1994a) cover 14 of the 30 heuristics, as they are developed for software interfaces, they apply to the interface heuristics examined by Federoff (see Federoff, 2002, p. 13 ff).

2.2.2 Interviewing

Types and Techniques Since interviews are not only known to be from great use in game tests and usability tests in general, the interview method is from much importance and in high consideration (see Fricker, 2012, p. 13 ff). The interview or user interview is a qualitative kind of research which allows a much more direct way to ask questions and adapting in real time. In the field of user experience and usability test they are commonly used to gain opinions and impressions about designs or software. Seaman (Seaman, 1999) distinguishes three types of interviews: the structured, unstructured and semi-structured interview, but there exist several other typologies for interviews for different approaches and fields (Danzico, 2010).

Structured, Unstructured and Semi-structured Interview The three mentioned interview types differ in how much the interviewer is in control of the questions and answers. The structured interview provides the most control and leaves questions completely in control of the interviewer. This may be used to gain mostly quantitative results if designed more quantified than flexible. However the unstructured interview as opposite can give most of the control about the questions into the hands of the interviewee and therefore is much more flexible with unexpected informations that may arise. Although this can deliver the best qualitative results it remains too costly most of the times. The semi-structured interview combines the both extremes and likewise allows a efficient way to be flexible and gain qualitative results, where at the same time the interviewer remains in more control of the interview (see Seaman, 1999, p. 562).

Interviewing and Observing To extend Seamans embodiments: if combined with observation, the interview can provide insight on what happened and what was observed; e.g. in usability tests, the testers get confronted with the test-subject, where their reactions are observed and converted in additional questions. This can be done by an experienced or expert user, during the confrontation, by "think out loud" (see Fricker, 2012, p. 14) or additionally by the interviewer, requiring some individual empathic ability. In order to structure an interview, interview guides can be used by the interviewer to steer the direction. Taking notes during the interview helps to keep record of the content and is one option to track the qualitative data. Since this can distract a lot from the interview itself, recording it removes a lot of overhead and ensures the interviewers focus on the interview itself and can also used improve the interviewers abilities (see Seaman, 1999, p. 563) (see Conyer, 1995, Recording Methods).

2.2.3 Usability Methods

Usability Testing & Usability Expert Evaluation The *Usability Testing* and *Usability Expert Evaluation* described by Laitinen, Christou and Fricker (Laitinen, 2005; Christou, 2010; Fricker, 2012) seem proven to be useful in game development as *Frozenbyte* states according to Laitinen: "usability expert evaluation and testing was a very positive endeavour. Frozenbyte will continue to use usability testing for its future games" (Laitinen, 2005). Both methods have strong connection to interviews and qualitative research in particular. In a user expert evaluation, a number of usability experts review a game or product, at first independently and then meet together and evaluate again to collect, discuss, evaluate and rate usability problems that occurred bundled in a report. The evaluation can adjust to have less experts or a smaller set of features to review, to get faster reports and is applicable also in early stages of development since "The earlier the expert evaluation is done, the easier and more cost efficient it is to make changes." (Laitinen, 2005). Like Fricker and Desurvire suggest, one can also use heuristics to reduce cost and time (see Fricker, 2012, p. 14).

Usability Testing & Quality Assurance The fact, that game development studios commonly have a QA department – which is getting more importance constantly – with up to "one of the biggest departments" (Frozenbyte, 2014) in the studio, underlines the importance of a certain dedication to testing. The author itself experienced it in an equal degree at the previously (see chapter 1.2) mentioned internship, that the QA has much importance as feedback for developers as well as designers. The work of a QA engineer requires also some kind of expertise in terms of usability engineering and testing in particular, this may qualify a QA tester also as a usability expert given by the experience in his field and according usability knowledge. Involving the players more into the development process (e.g. early access models) is getting popular, needs more exploration and thus is in relevance of this thesis (Game Lab, 2015; Walker, 2014).

With more the target group and the player in focus, usability testing is excellent to get new perspectives on the game and how the player perceives and interprets it (Laitinen, 2005; Dumas and Redish, 1999). Where the usability testing is often chosen because it is " a handy tool in focusing the development work especially when time and/or resources are limited." (Laitinen, 2005), it should be used with the first working version of the game (or prototype) to get the most out of the effort (see Fricker, 2012, p. 13-14).

User Testing The *User Testing* method as described by Conyer (Conyer, 1995; Fricker, 2012) aims to "determine usability by assessing users using a product" (Conyer, 1995). If focuses on analysing the users behaviour during the use and stands against *Usability Testing*

with the human expert, the usability engineer, evaluating the test. Where the behaviour involves also a context, their tasks, the possible additional knowledge or experience they might have, ideal would be a well designed profile of player types matching the focus group of the game or product. Methods and tools from user experience design like personas are popular amongst (web) designers and may be used for profiles in *User Testing* and game design likewise (**wiggins2007building**; Gube, 2010; Blomquist and Arvola, 2002).

Usability Evaluation & Related Methods The usability testing, or usability evaluation was recognized by Christou being very related to "psychology experiments" (Christou, 2010); with need for controlled environment, including the machine the test is run on, the conditions and a meaningful time or definition of tasks or goals (see Fricker, 2012, p. 14). The testing does not need a defined time limit to end (or respectively: reach the goal, finish the task) but it "could be that the players need to complete one level in the game." (Christou, 2010) as a predefined goal for all participants and thus have infinite time.

According to Lee in *Usability Testing for Developing Effective Interactive Multimedia Software* (Lee, 1999), there exist five categories for usability testing dimensions: "(a) learnability; (b) performance effectiveness; (c) flexibility; (d) error tolerance and system integrity; and (e) user satisfaction" (Lee, 1999) (also see Fricker, 2012, p. 11). Lee and Conyer discuss some procedures for usability testing, including advantages and limitations of the methods/techniques: "Heuristic evaluation, Pluralistic walkthroughs, Formal usability inspections, Empirical methods, Cognitive walkthroughs, Formal design analysis." (Conyer, 1995). Some more on *Usability Evaluation* will be discussed in chapter 5.

2.2.4 Cognitive Walkthrough

Wharton and Lewis give a detailed impression on the *Cognitive Walkthrough* (CW) method in the *Handbook of Human-Computer Interaction* (Lewis and Wharton, 1997). With focus on first use problems, the CW aims on evaluating the ease of learning and first usage. The methods detailed procedure simulates the problems experienced by a user in the three phases: Preparation, Analysis and Follow-up (Lewis and Wharton, 1997; Conyer, 1995). Lee and Conyer condense the advantages of CW on being effective for predicting problems, and capturing cognitive processes (see Lee, 1999, Procedures for Usability Testing). They can combine along with "paper prototypes, simulations or full functioning prototypes" (see Conyer, 1995, Cognitive walkthroughs) and cover novice as well as expert users. Additionally, a CW forces designers to think about the user – its background knowledge, goals, strategies, task handling and cognitive complexity – during usage of a game, UI or a general product (see Conyer, 1995, Cognitive walkthroughs).

Further Cognitive Walkthrough Methods Because the CW method "is time consuming and tedious" (see Federoff, 2002, p. 39) and has other limitations (see Conyer, 1995, Cognitive walkthroughs) (see Lee, 1999, Procedures for Usability Testing), there exists an adapted method, also mentioned by Federoff, the "streamlined cognitive walkthrough by that was established to be used by large software companies" (see Federoff, 2002, p. 39) and thus being better applicable. Spencer (Spencer, 2000) concludes on the *Streamlined Cognitive Walkthrough* to appeal more efficient to managers and developers and as a "good way to profile a user interface for potential problem areas, identify many steps that may be problematic for users, and accurately predict many usability problems" (see Spencer, 2000, p. 359) but with possible mistaken results. Federoff states, that the adaption attempts to "reduce the time required to perform the method, eliminate designer defensiveness, and reduce lengthy design discussions" (see Federoff, 2002, p. 39) but also argues for heuristic evaluations and heuristic walkthroughs (Nielsen, 1994a; Sears, 1997; Federoff, 2002).

2.2.5 User-Centred Design (UCD)

The *User-Centered Design* (UCD) (1) is accepted in the games industry and software industry in general; but often and increasing in other technological, social, cultural, medical, commercial or logistical fields too (see Pagulayan et al., 2003, p. 4) (see Desurvire and Wiberg, 2009, p. 2) (see Fricker, 2012, p. 15). The method requests its place within the UI development, with input of "the future system's stakeholders: designers, developers, usability specialists, graphic experts, and end users." (see Coyette et al., 2007, p. 150) especially gathered through design meetings, prototyping with paper & pencil to create the condensed design.

The method is described further by the *User Experience Professionals Association* (User Experience Professionals, 2014) and is pictured in figure 2.1.

- **Specify the context of use:** Identify the people who will use the product, what they will use it for, and under what conditions they will use it.
- **Specify requirements:** Identify any business requirements or user goals that must be met for the product to be successful.
- **Create design solutions:** This part of the process may be done in stages, building from a rough concept to a complete design.
- **Evaluate designs:** The most important part of this process is that evaluation – ideally through usability testing with actual users – is as integral as quality testing is to good software development.

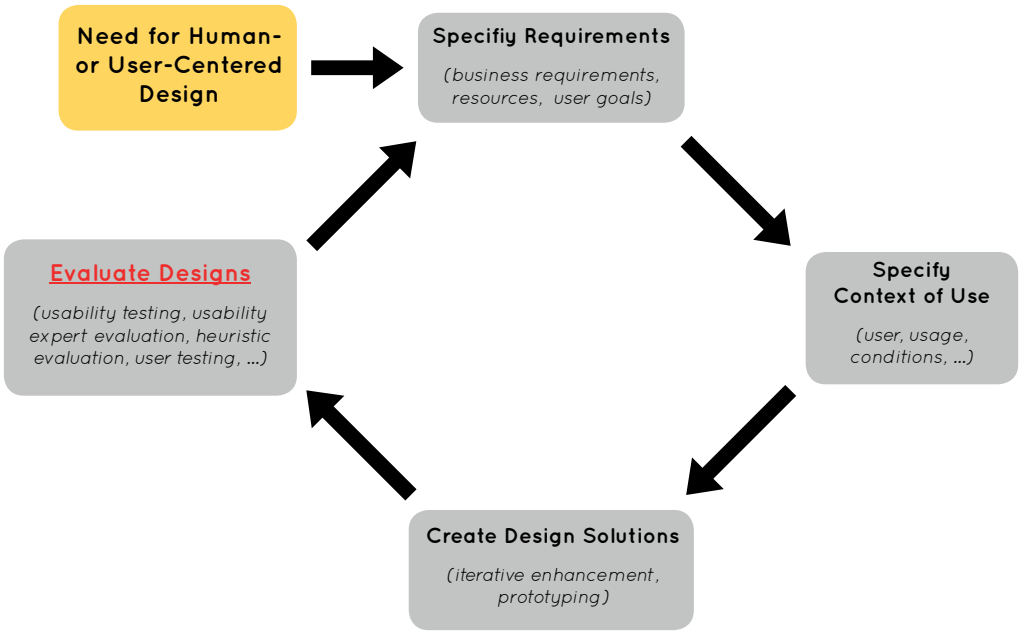


FIGURE 2.1: Human or User Centred Design method (User Experience Professionals Association, 2014)

2.2.6 Prototyping

This section only gives a short overview on prototyping research in general and is examined further in chapter 3. The literature and overviews on prototyping, also in relation to software engineering, are discussed by many experts. A more general outlook can be found in *Software Prototyping and Requirements Engineering* by Urban (Rome, 1992), with the different approaches on prototyping, pitfalls, requirements and specifications of prototyping techniques and tools. Another perspective is given by Hartson and Smith (Hartson and Smith, 1991) as they discuss evolutionary versus revolutionary prototyping, the rapid prototyping in particular with dedicated tools and approaches ("Experimental Systems" (see Hartson and Smith, 1991, p. 14)). More recent research on tools is provided by Myers et al. (Myers et al., 2000) and covers several valuable thoughts on tool-evaluation for chapter 4 and 7.

Landay and Myers (Landay and Myers, 2001) also emphasize the need of flexibility in early stages in UI design; Coyette et al. define the UI design as "a process that is intrinsically open , iterative and incomplete" (see Coyette et al., 2007, p. 151), and also relate to early UI design approaches as "extensively researched to identify appropriate techniques such as

paper sketching, prototypes, mock-ups, diagrams" (see Coyette et al., 2007, p. 151). They focus on multi-fidelity prototyping for an ease in adjusting the prototyping to different project requirements and resources (see Coyette et al., 2007, p. 151).

Like Guldbrandsen and Storstein, the author also aims for a more evolutionary approach in specifying the prototyping process that emerges from the research and appliance of trends, methods, techniques and tools (see Guldbrandsen and Storstein, 2010, p. 10). In their prototyping research they offer aspects of game prototyping in specific; relating to traditional software prototyping and user interface prototyping (see Guldbrandsen and Storstein, 2010, p. 56-60).

2.2.7 The Design Space Model

For the purpose of classification of UI and different elements in first person shooter (FPS) games, Fagerholt and Lorentzon (Fagerholt and Lorentzon, 2009) propose one model – see figure 2.2 – for design-space with spatial and fictional dimensions (see Fagerholt and Lorentzon, 2009, p. 48-49).

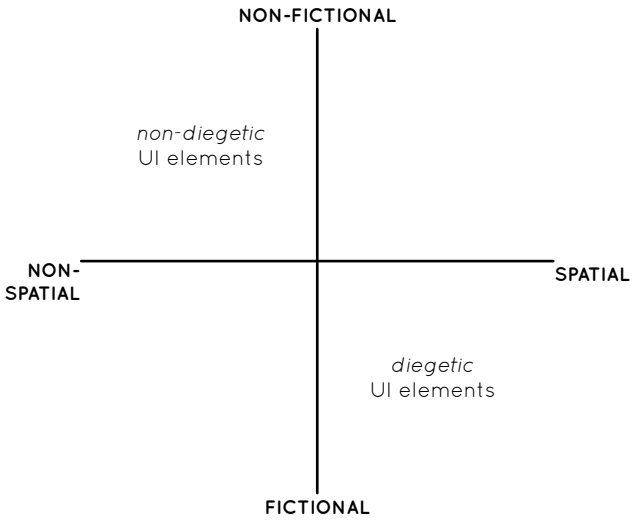


FIGURE 2.2: The first design space model proposed by Fagerholt and Lorentzon (see Fagerholt and Lorentzon, 2009, p. 49)

They further picture a model regarding rules, fiction and geometry (see figure 2.3) and can clarify the relation of UI elements (visual design, current state, controls, interaction design, feedback) or game elements in general, and the "role of the UI designer" (see Fagerholt and Lorentzon, 2009, p. 50).

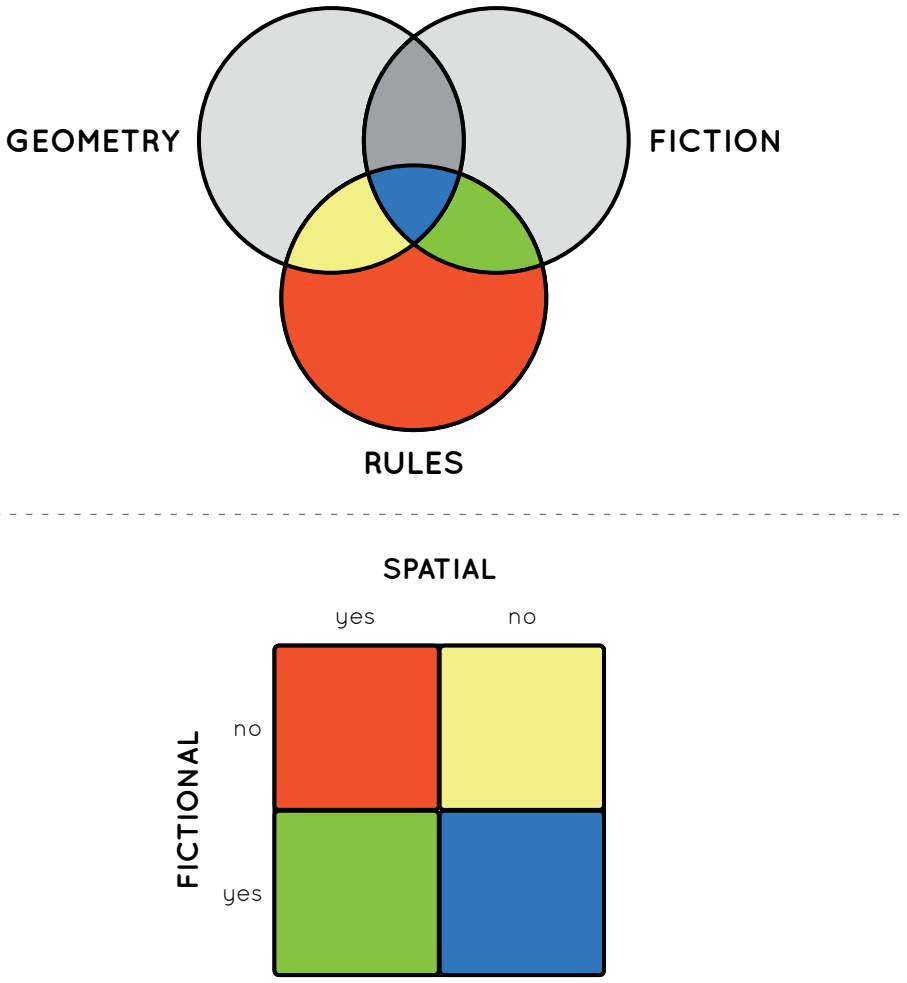


FIGURE 2.3: The second design space model proposed by Fagerholt and Lorentzon (see Fagerholt and Lorentzon, 2009, p. 50)

The model can be adapted for other game genres and other dimensions, like audio, animation or artificial behaviour.

2.3 Used Methodologies

As outlined in the previous sections, there exist many methods to approach the field of user interface prototyping and the tools involved during the process. Selecting the right amount an mix of methods seems critical for the relevance of the research of this thesis. With this thought, the *Mixed-Methods Approach* as described by Easterbrook et al. (see Easterbrook et al., 2008, p. 303-304) seems adequate. In the progress of the thesis research, prototyping processes are inspected in order to improve the existing knowledge, which suffers the *Action Research* approach mentioned in section 2.1.3. This process is applied in a *Case Study* or

more a *Case Qualitative Study* as the possible value lies in better understanding relations "between processes and product characteristics" (see Basili, 1996, p. 448) and is considered as having high relevance for independent developers and agile software development (Basili, 1993; Easterbrook et al., 2008; Basili, 1996).

The project leans partly on Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010) as they discuss the prototyping with the previous version of the UE, the UDK (Epic Games Inc., 2014a). The research of this thesis will adjust their prototyping process for its own needs if necessary (Guldbrandsen and Storstein, 2010; Basili, 1993). In terms of Basili this may also be viewed as an evolutionary approach, but the appliance on a case study definitely classifies as revolutionary (see section 2.1.2).

Beside the prototyping, there are several other methods from the HCI (see section 2.2) field that will be considered in design process of experiments of this thesis. Some of the relevant ones, like the interviews, focus groups, heuristic evaluation, usability testing or cognitive walkthrough, where listed in section 2.2. The mixed method approach therefore can combine methods to assess most of the possible outcomes and retrieve further research opportunities. One also might use *Experimental Design* as described by Seaman, as it also relies on combination, but respectively of both qualitative and quantitative methods for more "fruitful" results (Seaman, 1999).

Chapter 3

Prototyping Processes

> *"Sherlock Holmes observed that once you have eliminated the impossible then whatever remains, however improbable, must be the answer. I, however, do not like to eliminate the impossible."*
>
> Douglas Adams – Dirk Gently's Holistic Detective Agency

This chapter dedicates to prototyping techniques and their result: the prototype. Although prototyping exits amongst all constructive disciplines, this thesis focuses on software & usability engineering, design with game development in particular (see Guldbrandsen and Storstein, 2010, p. 51). the The prototype can be seen as scale model, which lacks in accuracy in which it reflects the real-world system and usually focuses on UI, system functionality or system performance (Weiser, 1982; Hartson and Smith, 1991). Where there exist various prototyping techniques or processes, the basic principle: *abstracting from the specific* is natural and the way we perceive the reality in general. The purpose of this chapter is to answer RQ1.1 and RQ1.2 (see 1.7) and choose a technique or a mixed set of techniques to apply them in the project.

3.1 Why do Prototyping?

The relevance of prototyping is acknowledged amongst the game industry and related scientific research (Fagerholt and Lorentzon, 2009; Guldbrandsen and Storstein, 2010; Crawford, 2014; Ollila et al., 2008; Rome, 1992). The author also recognized it in the mentioned internship (see 1.2), although there was little opportunity for studying the processes in particular, there was always the thought of *iterations* – instead of a final and unchangeable

result – in the definition of tasks, in order to approach changing requirements influenced by game design, art and engine development.

Comparable experiences where made by Federoff (Federoff, 2002), in prototyping being used in the pre-production phase, which may also reason from Shelley's thoughts about "the values of early prototyping" probably revealing "whether or not a game is going to work early on in development." (see Shelley, 2001, p. 3). As mentioned in chapter 2.2.6, Landay and Myers (Landay and Myers, 2001) also point out the need of flexibility in early design stages, which can be enabled with prototyping techniques. Further, Hartson et al. describes prototyping as "effective even when performed manually, especially in the early, conceptual stages of development." (see Hartson and Smith, 1991, p. 2), which again is a comparable thought: *use prototyping as early as you can, all the time*!

More with the psychological aspect in mind, Hartson and Smith (Hartson and Smith, 1991) argue for prototyping being a "*natural technique*" (see Hartson and Smith, 1991, p. 3) and according to developmental psychology, humans solve problems or develop new approaches while abstracting from the specific (see Hartson and Smith, 1991, p. 3). Thus needing only training for a specific set of prototyping rules, already having the natural ability to apply them.

With better resource efficiency (45% less effort according to Hartson and Smith (see Hartson and Smith, 1991, p. 5)), rapid prototyping techniques clearly argue for the statement: "should be done for nearly every game." (see Federoff, 2002, p. 36) as they can possibly prevent commercial (and functional) failures. If the user is more involved in the design and prototyping process, there will be a much higher of acceptance amongst the whole target group; the final prototyped game is easier to learn and use, and thus more what the player desires (see Hartson and Smith, 1991, p. 6). With consequent prototyping, there appears to be less pressure when it comes to deadlines (see Hartson and Smith, 1991, p. 6), and was also experienced by the author during the internship.

It seems, a *feature prototype* or "*game sketching*" (see Ollila et al., 2008, p. 2) (Agustin et al., 2007) of a specific (game) design aspect can be produced faster, where at the end of each iteration (or deadline), there is mostly some work done for next targeted iteration. This enables a fast functional approach to get a certain impression on the feature and then continuously change, adjust, improve, polish and observe, till there is no significant improvement. This may give acceptable results frequently, but can also end up in rejecting a feature at all or even rethinking (game) design: *survival of the fittest*. Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010) reflect on the evolutionary aspects of prototyping in relation to games in particular. With less experience in game design or trying something very innovative or revolutionary, there is a benefit in (paper) prototyping; the game play, design and mechanics can be observed or demonstrated to gain direct user

feedback and impression, during the iteration and testing (see Ollila et al., 2008, p. 2). Ollilia et al. lists the reasons to do so, especially during design or pre-production: "to test game design ideas or concepts, to generate new design ideas, and to probe the attitudes, opinions, and behavioral patterns of potential players" (see Ollila et al., 2008, p. 2). The ability to gain user feedback with a prototype is a most valuable one, and leads to enhanced communication (see Hartson and Smith, 1991, p. 6); between user and developer, the developer team and potentially the users community.

With the UI design being a process, which is intrinsically open, iterative and incomplete; prototyping can support the creative processes, as it leaves room for those by design, while generating feedback across all domains involved in prototyping (see Coyette et al., 2007, p. 151). Techniques, especially for early UI design, are researched by experts (Snyder, 2003; Ollila et al., 2008; Johnson and Wiles, 2003; Coyette et al., 2007), covering " paper sketching, prototypes, mock-ups, diagrams " (see Coyette et al., 2007, p. 151) as the most appropriate ones to use in prototyping processes. Prototyping supports iterative processes in game development in many aspects, creative processes like game design or UI design in particular. It naturally fits into the life-cycles in game development, in the bigger traditional productions and as well in smaller independent studios (Blow, 2007).

3.2 Dimensions & Kinds of Prototypes

3.2.1 Horizontal versus Vertical

Software prototypes are commonly classified in two dimensions: *horizontal* and *vertical*; varying in feature-set and functionality or realism (Nielsen, 1994b; Hartson and Smith, 1991; Guldbrandsen and Storstein, 2010). Where horizontal prototypes give an overview, not focusing functionality, and are usually more *top-down* approaches. They usually target the usability and experience (e.g. design & layout, efficiency, effectiveness, satisfaction, flow, cognitive load) and are typically used for graphical user interfaces (see Guldbrandsen and Storstein, 2010, p. 53). A vertical prototype focuses on functionality, single features or limited feature-sets are implemented close to the final product and more likely – for the feature(-set) – goes *bottom-up*. The relation between the dimensions is pictured below and in figure 3.1:

Horizontal Prototype: broad feature-set but with low functionality/realism

Vertical Prototype: limited features but those are functional/realistic

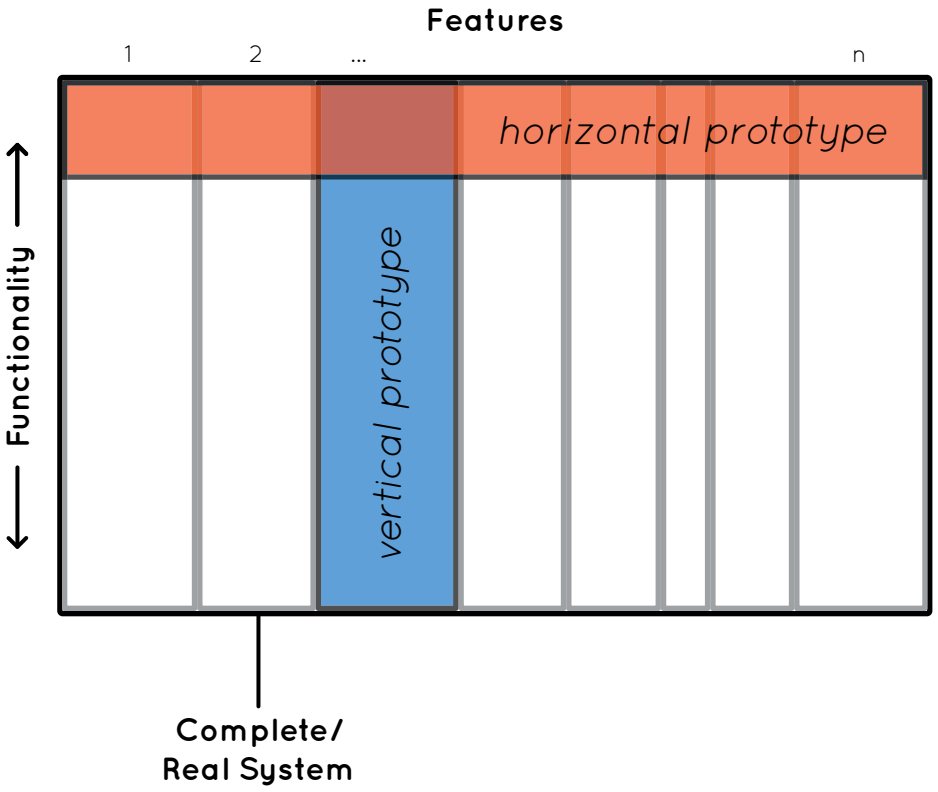


FIGURE 3.1: Prototype Dimensions according to Nielsen (Guldbrandsen and Storstein, 2010; Nielsen, 1994b)

3.2.2 Other Dimensions

Balancing the dimensions is crucial, while vertical prototypes are limited in exploring the relation to other features, horizontal prototypes lack functionality needs of complex problems or innovative approaches. Due prototyping scenario is reduced in vertical and horizontal dimension, having a clear idea of the resources, requirements, user (or player) and others involved in the process, helps choosing the right combination and thus the prototyping technique (see Hartson and Smith, 1991, p. 9). Hartson and Smith (Hartson and Smith, 1991) give another set of dimensions, that can be used to classify prototypes. They describe four *other dimensions* (see Hartson and Smith, 1991, p. 9):

Specification: how are interface designs specified

Maturation: how does the prototype grow/integrate into a (final) product (or even another more specialized prototype)

Scope: can the prototype include the whole system or just the interface, a single feature or feature-set

Executablility: can the prototype be tested & executed at any time

Friedl (Friedl, 2002) uses fidelity as dimension, ranging from the low-fidelity that can be said about pen-and-paper sketches to a high-fidelity fully functional prototype (see Guldbrandsen and Storstein, 2010, p. 54). The techniques are described by Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010); some of them will be discussed in section 3.3 and are also pictured in figure 3.2.

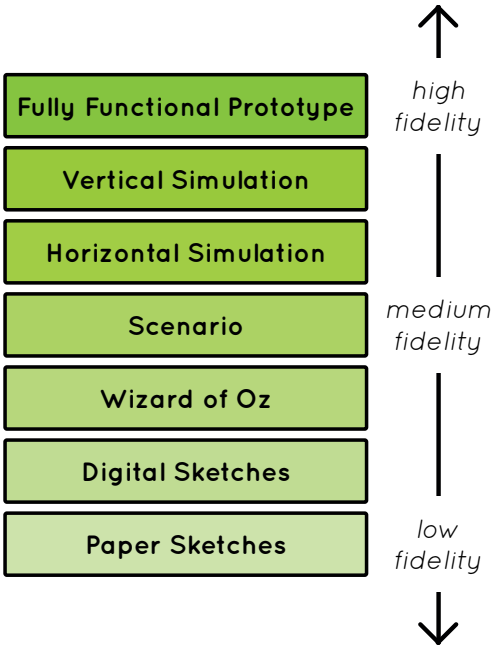


FIGURE 3.2: Seven prototyping methods ordered by fidelity according to Friedl (Guldbrandsen and Storstein, 2010; Friedl, 2002)

3.2.3 Kinds of Prototypes

To provide another concept to categorize prototypes, as mentioned at the chapter begin, prototypes usually cover UI, functionality or performance. In order to extend the UI aspects somehow, splitting them into interface and experience seems reasonable to cope with ongoing development in the user experience (UX) domain. So the author proposes the four *categories or kinds of prototypes*:

Interface: focus is on layout, layers, appeal, efficiency, effectiveness, satisfaction for the sake of design and usability

Experience: emphasizes on flow or ease of use, but also on efficiency, effectiveness and satisfaction and generally more onto UX

Functional: include a single function or feature, not focusing on interface or usability in the first place, to get an impression of mostly complex parts of a product

Performance: used to test the capabilities of a product, how it scales, especially time and hardware consumption

From the combination of different dimensions, different techniques or types have been established through research. In thesis purpose, the more UI and game related ones will be described. A in depth view of each technique is beyond the project scope, the references to detailed description provided accordingly.

3.3 Types of Prototyping

3.3.1 Rapid Prototyping

Traditionally targeting more the construction and fabrication domain, rapid prototyping has been applied to the software engineering and gaming domain as well (Hartson and Smith, 1991; Crawford, 2014). The term *rapid* already refers to the purpose of iterating faster, and get results earlier. A rapid prototype needs to execute most of the time, this ensures it can be evaluated through testing more frequently. Low-fidelity approaches, like pen-and-paper sketches, board game elements, or analogue and digital mock-ups are used, they are best because they have little requirements in skill, budget or time. This technique is commonly used to reveal design insufficiency and gather new ideas, which will than contribute to the whole development process. Some may refer onto rapid prototyping as an approach of throwaway prototyping, which might merely be a subtype of rapid prototyping (Guldbrandsen and Storstein, 2010; Ollila et al., 2008).

3.3.2 Throwaway Prototyping

A throwaway prototype – also referred to as *proof-of-concept* or *proof-of-principle* – is clearly designed with the possibility of discarding up to everything achieved during the process (Guldbrandsen and Storstein, 2010). There is enough insight gained, if there is a negative result, proving the approach as dead end, not usable or lacking functionality. This

technique can cover all *kinds of prototypes* (see 3.2.3), it is used to test for innovative ideas, whether they work or not; this may be an interface, a system architecture or experimenting with new or other hardware and platform (Ollila et al., 2008; Coyette et al., 2007).

3.3.3 Evolutionary Prototyping

The intend of this technique is starting with the core of a product and successively extend the scope until all final product requirements are accomplished. This requires to build the software architecture and according processes and therefore support extensibility and modifiability, to add new features, enhance existing ones and increase fidelity (Guldbrandsen and Storstein, 2010; Ollila et al., 2008). With evolutionary prototyping case studied by Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010), this technique apparently works in small teams; their method combines several aspects from evolutionary, rapid and throwaway prototyping, but they also include other approaches (see Guldbrandsen and Storstein, 2010, p. 67-69). As the evolutionary approach ensures the product evolves within the project boundaries, it is mostly very efficient and additionally is useful in contexts, where requirements are yet not fixed or known (see Gube, 2010, p. 54).

3.3.4 Wizard-of-Oz Prototyping

This is a very common technique to test user interfaces without having to implement functionality on the software's side. Instead the feedback is given by the operator (or "wizard"), reacting to the users interactions from "behind the scenes"; and is usually done with a second screen for the operator in order to manipulate the users view. It can expose patterns that were not expected by design, and then react with a fast adjustment to support that kind of interaction/pattern. The Wizard-of-Oz has its limits in the operator itself, because during the test-session the conceptual, motor and emphatic abilities matter for best text-execution (Guldbrandsen and Storstein, 2010; Ollila et al., 2008). This technique can be used in various contexts, for testing a mobile app (simulating e.g. the location), a FPS game (simulating the HUD in general) or a browser game (simulating e.g other users).

3.3.5 Physical Prototyping

A physical prototype can include all physical objects one can imagine: from board game figures, pen-and paper to trading card games; everything not relating to software (see Ollila et al., 2008, p. 2). Some may refer to a throwaway prototype or "*game sketching*" (see Ollila et al., 2008, p. 2), as the physical parts are clearly "thrown away"; nevertheless, the game design itself does not need to be discarded and may flow and evolve into further

prototypes. Thus, physical prototypes can be throwaway prototypes but not vice versa and again, they are pretty useful in proving a game design in various aspects. Where Augustin et al. (Agustin et al., 2007) consider physical prototyping or "*game sketching*" more as "very early prototype[s] sketches" (see Ollila et al., 2008, p. 2) rather then a real prototype.

Paper prototyping is probably the most commonly known technique and researched by experts along many disciplines (Ollila et al., 2008; Snyder, 2003; Frick et al., 2001; Pagulayan et al., 2003; Coyette et al., 2007). Being in focus of the UCD research, it has been used for UI design and testing, in order to test usability and user experience. Using a paper prototype is practical if there are no other tools available, for a fast presentation of ideas/concepts, and they support collaborative design processes. Using paper (or physical) prototypes in the early rapid prototyping process is very common, as mentioned in section 3.3.1.

3.3.6 Multi-fidelity Prototyping

With multi-fidelity prototyping, the thoughts of Friedl (Friedl, 2002) and other experts (Paternò and Volpe, 2006; Engelberg and Seffah, 2002; Plimmer and Apperley, 2004) on fidelity are carried further. Coyette et al. (Coyette et al., 2007) propose multi-fidelity as mixed-fidelity concept, where elements can have different fidelities, with having only one fidelity at a time to act upon. They target to make the transition from one fidelity to the next higher one easier, and designers appreciate such possibilities for a prototyping tool (Coyette et al., 2007). This shows aspects of evolutionary prototyping and could also be used in early rapid prototyping, for an ease in adjusting the prototyping process to different project requirements and resources (see Coyette et al., 2007, p. 151).

3.3.7 Team Prototyping

Federoff (Federoff, 2002) exposes the team prototyping technique to developers, during their research, with some interesting feedback. Whereas in team prototyping a group of designers, or all people involved in the prototyping process, gather around a big screen (e.g. beamer or TV) and together discuss layout and functionalities. Changes can be made during the process, and the findings should be recorded in order to be improved. Although, Federoff states "this method should [not] replace paper prototypes" (see Federoff, 2002, p. 37), due they are faster and easier created. It could be used in early stages, where the interface designer and game designer "produce the first iteration together." (see Federoff, 2002, p. 37).

3.3.8 Storytelling Prototyping

This technique is a mix of physical prototyping and a "wizard"-like narrator, which verbally explains what will happen next. It is suggested to explain puzzles, riddles or general game design ideas to players, and discover whether the player can solve or comprehend the respective design idea (Federoff, 2002). The general process is, to create a scenario with story, consider the users perspective: where can he see the current progress/state, why does he need to transition between to states (choose options) and also include design and user experience principles. With this done, you can perform the test on real users, supported by physical prototypes, explaining the story/scenario (Federoff, 2002; Ballay, 2012).

3.3.9 Video Game Prototyping

Like already outlined earlier (see section 3.1), there already exist conventions in academic research and within the video games industry (see Guldbrandsen and Storstein, 2010, p. 56). Again, Guldbrandsen and Storstein provide some aspects of prototyping games: *world design*, *system design*, *content design*, *game writing*, *level design*, *user interface design* (Guldbrandsen and Storstein, 2010; Schreiber, 2009). The aspects can be prototyped in a smaller set on its own, as there is mostly need of multiple prototypes due the multi-disciplinary of game development. World design, content design and level design mostly involve 2D and 3D representations within the game world, and the purpose of prototyping is commonly focusing on aesthetics, visual appearance, atmosphere, theme and consistence. The system design covers game mechanics and imminent functionality like saving and loading a game state. With user interface design and game writing more relating to the HUD or screen plane than to the game world, their prototyping benefits are immense as they connect game mechanics and world (Fagerholt and Lorentzon, 2009; Guldbrandsen and Storstein, 2010). Thus, the six aspects could be grouped into three categories:

World: involves content, level and world; everything perceived or within the virtual game world (e.g. models, animations, player perspective, challenge) and the overall atmosphere, appearance, aesthetics, consistence and theme

System/Simulation: the rules and mechanics; algorithms, functions and meta functionalities that enable the game design

Interface: how the player interacts with the game (keyboard, mouse, game-pad, touch, etc.) and how the game gives feedback (visual, auditory, haptic, etc.) to the player, about his state, progress and options within the game

The categories are free from actual game lore, style or genre; but a game may vary in the amount in which it needs prototyping for the given aspect or respectively category. As game writing has most influence on the game's lore and style, it is only considered related to UI design because of the fact that writings are displayed within the UI. It needs attention during UI design, and maybe also in relation of mechanics (e.g. a dialogue mechanic) or *system/simulation* and *world*.

Interface prototyping can be performed similar to approaches from HCI (Guldbrandsen and Storstein, 2010; Fullerton, 2014); using interface guidelines and heuristics together with evaluation for games is crucial and discussed in chapter 5. Nevertheless, important for the prototype is knowing the platform(s) targeted, and also the desired input devices supported by those platforms. If the game runs on PC and consoles, it may need special equipment like a game-pad on a PC, and has to be considered. Like the input devices, also all possible feedback opportunities should be taken into account. This can be a simple LED, vibration/rumbling, and of course the screen or sound. Since the interface design is one amongst may a creative processes in game development, it – as mentioned at the end of section 3.1 – highly profits by the use of early low-fidelity prototyping. Here again, it is necessary to also test the prototype with players, in order to uncover potential pitfalls and insufficiencies in game and UI design (see Guldbrandsen and Storstein, 2010, p. 59).

Prototyping game mechanics or *system/simulation* can easily reveal "how the game is played" (see Guldbrandsen and Storstein, 2010, p. 57) which can improve – for evolutionary or rapid prototyping – the design during development. Games with good mechanics do not have to use the latest graphics technologies to be "fun to play" (Guldbrandsen and Storstein, 2010; Blow, 2007). Guldbrandsen and Storstein already suggest combining rapid and evolutionary approaches to overcome the time between the design phase, having nothing to test (nothing to test dilemma), and the first evaluation which then evolves within much clearer boundaries (see Guldbrandsen and Storstein, 2010, p. 57). Ollilia et al. also list some applications of prototyping from Ballagas and Walz (Ballagas and Walz, 2007), where they use a board game to prototype game mechanics, with findings regarding travel times, spatiality and whether the game is fun (see Ollila et al., 2008, p. 3).

Guldbrandsen and Storstein further cover the level prototyping, creation of visual elements and challenge are key features and are covered by the *world* category. As the thesis covers UI in particular, taking level prototyping into account is out of scope. The relation between the six aspects (*world design*, *system design*, *content design*, *game writing*, *level design*, *user interface design*) and the proposed categories (*world*, *system/simulation*, *interface*) is pictured in figure 3.3.

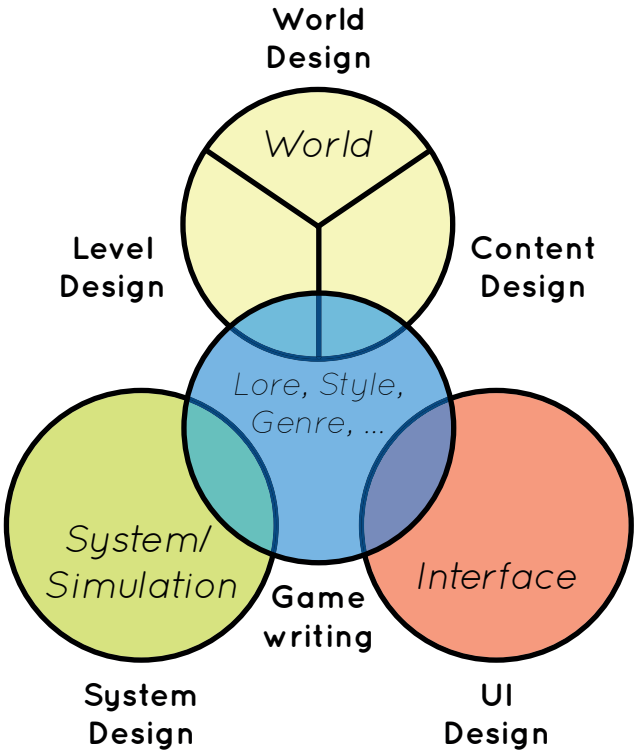


FIGURE 3.3: The six aspects of game design prototyping, categorized by *world*, *system/simulation* and *interface* – connected by the aspect of game writing (see Guldbrandsen and Storstein, 2010, p. 56-60)

3.4 What can possibly go wrong?

3.4.1 Communication breakdown

Misunderstanding the users objectives can be a result of having less knowledge of the target group, lacking empathy during tests or lacking application and knowledge of UI design principles and human cognitive processes. Another pitfall is misjudging the role of prototyping, e.g. expecting features to work which are out of the prototypes scope. While testing or evaluating a prototype the customers or users must be able to provide feedback, that again needs to be collected somehow (through recording, transcribe or taking notes). Likewise the context of the test needs to fit comparable conditions of a real application, the setting influences users and therefore quality and quantity of results (Hartson and Smith, 1991; Rome, 1992).

3.4.2 Imbalanced approach

One should avoid designing strictly bottom-up or top-down and instead mix both: the bottom-up aspects of prototyping and the top-down aspects of software engineering. There are multiple layers in the architecture of the software or game, and in order to connect (e.g. the data of a save-game with the UI representation and additionally the functionality to load it) those in the context of a prototype; bottom-up and top-down will build the bridge in between the layers (see Hartson and Smith, 1991, p. 7-8).

3.4.3 Dedication

Dedicating too much to a prototype may enforce attachment of developers and designers, or furthermore lead into spending too much time with a possible throwaway prototype. When too much effort is put into new features, without testing or evaluation, possible side effects e.g. functional errors, unexpected user behaviour or game imbalance can result. The urge to fix the possible effects is costly in time and budget, and also can lead to a feature creep (a continual expansion or addition of requirements). Nevertheless, yet implementing prototyping processes in companies can be costly and lower the productivity (Rome, 1992).

3.5 Choosing Prototyping Techniques

That "prototyping should be done for nearly every game" (see Federoff, 2002, p. 36-38) is a widely spread opinion, choosing the appropriate techniques is the purpose of this section. The selected prototyping techniques will define the further progress of the project, as they are practised to answer the research questions (see section 1.7). Ollilia et al. offer some guidelines on selecting methods: *choosing by desired result*, *choosing by requirements* and *choosing by game type* (see Ollila et al., 2008, p. 10-15). Those were primarily designed for pervasive games but could "be applied to nonpervasive games as well." (see Ollila et al., 2008, p. 10).

The first step in the project will be, after some game UI research regarding RPGs, the conceptualization and creation of the interface. In order to prove the design(s) regarding completeness, functionality and to reveal insufficiencies, those need to be evaluated. This process is considered to generate ideas, thus requires more expert testers and evaluators, with experience in RPG games. This early design phase requires lower completeness and fidelity, this may also result in generating more ideas, because many aspects regarding the final product are left to the testers imagination (see Ollila et al., 2008, p. 10, Table I). With the *desired result* considered, the *requirements* and *game type* from Ollilia et al. (see

Ollila et al., 2008, p. 11, Table II; p. 13, Table III) do not apply to the project and will not be considered.

For the first stage, physical prototyping (paper prototyping, see 3.3.5) seems suitable, it requires less effort in creation and is highly flexible. The technique can be combined with storytelling and Wizard-of-Oz (see 3.3.8 and 3.3.4) to give the player a context within the game (while paper prototyping). To generate even more ideas, using multiple paper prototypes for different UI designs/concepts could be useful, despite the fact that concepts may be discarded completely; this is clearly an aspect of throwaway prototyping.

Further iterations will occur in the manner of rapid prototyping (see 3.3.1), but may also involve aspects outlined for evolutionary and video game prototyping from section 3.3.3 and 3.3.9. Additionally the concept of team prototyping (see 3.3.7) seems to be an interesting one, it is strongly considered to be used during the prototyping process.

Several prototyping processes were displayed above – concerning UI in particular and games in general; the research questions **RQ1**, **RQ1.1** and **RQ1.2** listed in section 1.7 were discussed and answered with this chapter. The remaining question **RQ1.3** will be answered with the case study, later in chapter 6.

Selected Technique	**Advantages**	**Risks**	**Synergistic Techniques**
Paper Prototyping	early appliance, cheap and fast, low fidelity/complexity needed, prove/disprove a design/concept, experimenting with layouts	too much commitment for a possible throwaway prototype	Storytelling Prototyping, Wizard-of-Oz Prototyping
Team Prototyping	make fast on-the-fly changes, imminent results, enforces communication	needs experts or specific background	Storytelling Prototyping, Wizard-of-Oz Prototyping, Digital Paper Prototyping
Rapid (Game) Prototyping	faster iterations, prove/disprove a design/concept, high execute-ability	missing cooperation, prototype is seen as final system, top-down vs. bottom-up	Physical Prototyping, Evolutionary Prototyping

TABLE 3.1: Selected prototyping techniques, advantages, risks and possible other synergistic techniques

Chapter 4

Prototyping Tools

> *"If you try and take a cat apart to see how it works, the first thing you have on your hands is a non-working cat."*
>
> Douglas Adams

This chapter gives a short overview of research related to prototyping tools, especially for games and game UI. It aims to generally discover features of tools; what they commonly share, despite the fact some are not mainly designed to be used for games or prototyping. The overview will help to gain insight in what is commonly used, what is done well in special cases, what are best practices and thus to finally better understand what must be considered choosing a UI prototyping tool and during evaluation. This includes selecting common features one would expect for the subject of prototyping and therefore selecting the right tool or tool-chain for the prototyping process. The chapter will additionally discuss the selected tools, including the Unreal Engine, because it is the subject of this case study.

4.1 Related Work and Projects

Guldbrandsen and Storstein provide aspects for prototyping tool evaluation, as they focus on the *Unreal Development Kit* (UDK), the predecessor of the projects subject: the *Unreal Engine 4* (UE4) (see Guldbrandsen and Storstein, 2010, p. 79-81). Likewise, this thesis will focus on a qualitative approach in evaluation, for the same reasons (see Guldbrandsen and Storstein, 2010, p. 79). They base they criteria for the engine evaluation on Wang and Wu (Wang and Wu, 2009) and list the following: learning curve, development speed, flexibility, stability, community support and documentation, licensing and competitiveness (see Guldbrandsen and Storstein, 2010, p. 79-81). Other criteria – gleaned from literature

– may also be considered, e.g. Coyette et al. identifies three categories "depending on their fidelity level: high-fidelity tools ; medium-fidelity tools" and "low-fidelity tools" (see Coyette et al., 2007, p. 151-152). Accordingly they propose seven criteria: Amount of fidelity, Fidelity transition, Shape recognition, Gesture recognition, Output re-usability, Multi-platform support and UI types (see Coyette et al., 2007, p. 152-153). Myers et al. present themes they found for tool evaluation: the parts of the user interface that are addressed, threshold and ceiling, path of least resistance, predictability and moving targets (see Myers et al., 2000, p. 4). Furthermore they detect "Issues for Future Tools" (see Myers et al., 2000, p. 23): changing skill and dexterity levels of users, non-overlapping layout or rectangular and opaque interactive components, using (less) fixed libraries for interactive components, a possible interactive setting, reduce required user attention, more support for evaluation, and tools with focus on usability.

4.2 UI Prototyping Tools

4.2.1 Pen-and-Paper Sketches

Using pen and paper is a natural way to sketch ideas, designs and concepts fast and affordable. As mentioned earlier (see 2.2.5 and 2.2.6), this technique is very common amongst HCI and game development and does not need any more explanation at this point.

4.2.2 Adobe Creative Suite

The *Creative Suite* (CS) from Adobe (Adobe Systems Incorporated, 2015) consists of various programs to create media content. This involves audio, video, analogue and digital images/pictures, illustrations, layouts or artworks along with some tools for interactive content. Because the scope of the CS is immense, this section focuses only on the programs that can be considered for UI design and prototyping (see Guldbrandsen and Storstein, 2010, p. 54-55).

Flash

Known for many applications, like browser or web games, advertisement or audio and video players, Adobe Flash is a powerful platform/framework for interactive multimedia content. It provides a object oriented scripting language *Action Script* (AS), which exists in the third version (AS3), and originally was designed for easy access of 2D and 3D manipulation and animation. Flash can be used to create interactive prototypes very fast, if one is skilled

enough. The author has some experience with Flash and AS3 from the internship (see 1.2), which sadly are merely little comparable to the common approach of prototyping with Adobe Flash.

Illustrator & InDesign

With Illustrator more used for vector illustrations (e.g. logos) and InDesign for the general layout of documents or books, the two can be used for digital paper prototyping, either together or separately. In fact, InDesign offers a more general approach to layout and layers, and additionally support vector graphics with sufficient flexibility. This is a comfortable balance, while the results can be printed for paper prototyping or work as base for team prototyping. Additionally the artefacts can be exported easily into vector or pixel formats (like SVG or PNG) for usage in a game engine (e.g. the UE4). Illustrator offers an enormous amount of functions around vector graphics, nevertheless it is not focusing on layout like InDesign does, which can be obstructive.

Photoshop

Laying the focus onto pixel graphics rather than vector graphics, it is widely used for photo manipulation and game artworks or (UI) mock-ups. Together with 3D modelling and animation tools like *3ds Max* from Autodesk (Autodesk Inc., 2015), it can be used to combine 2D and 3D artefacts to showcase the combination of both fast and easy. Like all Adobe (Adobe Systems Incorporated, 2015) tools, there exist exchangeable file formats to exchange content between Flash, Illustrator, InDesign, Photoshop, as well as most of all other programs provided by the *Creative Suite*. Also supporting layers and capable of a sufficient vector graphics support, Photoshop offers a lot one can use for digital prototyping.

4.2.3 Frameworks

For more functional prototypes, but also UI prototypes, programming or scripting languages/frameworks can be used with their provided default component sets. There is a large amount of free and commercial ones available, like *.NET*, *Java*, *WPF*, *Qt*, *Flash* and *GTK*. Most of those are present on may platforms, from embedded system to mobile to desktop. They often have dedicated IDEs, with a UI designer and of course a source code editor, and can be used to implement functionality for more general UI features very quick.

4.2.4 Others

Balsamiq, Lucidchart, Pencil Project & Concept.ly

Those tools are mostly designed for prototyping common software UI apart from games, like desktop, mobile or web applications. They can be used for more traditional UI designs, and no game specific elements are required. While *Balsamiq* (Balsamiq Studios, LLC, 2015), *Lucidchart* (Lucid Software Inc., 2015) and *Concept.ly* (Concept.ly.com, 2015) provide a trail to test them for requirements, *Pencil Project* (Evolus, 2008–2012) is open-source and available for free. All tools provide a different feature-set, which can be viewed on the according website.

Gimp & Inkscape

As contrast to *Photoshop* and *InDesign*, the free alternatives *Gimp* (GIMP Team, 2001–2015) and *Inkscape* (Inkscape Team, 2008–2012b) provide a comparable feature-set and run on more platforms (like several *Linux/Unix* distributions). There also exist many plug-ins for both tools, an active community and documentation (Inkscape Team, 2008–2012a; The GIMP Team, 2001–2014).

4.3 Game UI Prototyping Tools

In this section an overview on other – more game related – tools that can be used for prototyping the user interface of a game. *Unity* (Unity Technologies, 2015), the *Cryengine* (Crytek GmbH, 2014) and the *Godot Engine* (Linietsky and Manzur, 2014) are game engines, where *Scaleform* is a UI framework that can be used with a custom engine and also exists as integration into several 3rd party game engines (Autodesk Inc., 2014b). The *XNA Game Studio* is based on *DirectX* and *.NET*, providing support for several Microsoft (Microsoft Corporation, 2015a) platforms; the Ogre3D engine (OGRE, 2015) – in contrast – is a specific graphics engine targeting multiple platforms.

4.3.1 Unity

This engine has its favour in being easy accessible, flexible and broadly compatible and also is free for no-commercial use in the *Unity Basic* version (Unity Technologies, 2014a; Unity Technologies, 2014c; Unity Technologies, 2014b; Unity Technologies, 2015). It comes with the *MonoDevelop* IDE for scripting with *C#*, *JavaScript* and *Boo*, the development

can occur on most desktop operating systems, while the games can be published to mobile, console or PC environments. The author of the thesis has some experience with *Unity Basic* as well as *Unity Pro* from a students game-project (Arous et al., 2014). Whereas the project was structured in milestones and according prototypes, a UI prototyping was not performed in particular and the implemented UI was in best knowledge of only designers and UI programmers. From the experience in this particular project, the engine has some mighty abilities for UI prototyping also enabled through access on the engines editor, thus providing additional control during prototyping with specific interfaces. Although the UI design can be previewed easily, there is no specific stage for the design, the same editor-view (with the projected UI on top) is used during design mode. Objects can be moved with the world editors functionality, which is often clumsy and the position is best adjusted in the location values in the object details panel.

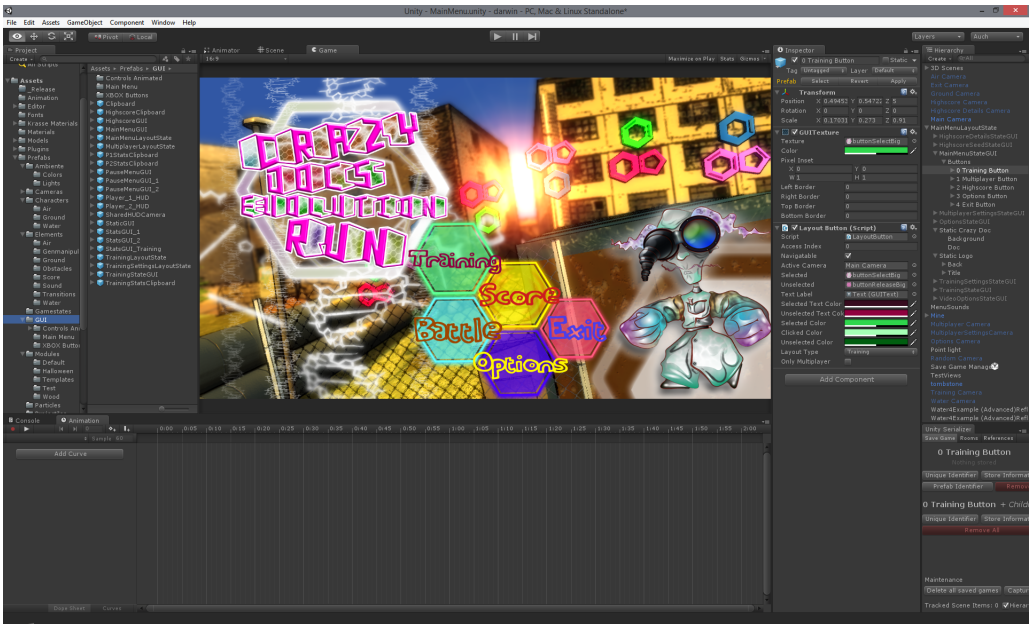


FIGURE 4.1: User interface design with Unity, showcased with the game *Crazy Doc's Evolution Run* (Arous et al., 2014)

4.3.2 Cryengine

This high-end engine from Crytek (Crytek GmbH, 2015a) has a broad feature set and was used for several AAA titles on different platforms (Crytek GmbH, 2014). The author has no experience with the SDK provided by Crytek, but one can imagine that it may be suitable for innovative FPS user interfaces. From the features displayed on the engine website, there is no dedicated UI workflow noticeable (Crytek GmbH, 2015b).

4.3.3 Godot Engine

This engine was found during research and offers some interesting features (Linietsky and Manzur, 2014). It is comparable to *Unity* and improves some features like scene handling and UI design. Like *Unity* it also targets multiple platforms.

4.3.4 Scaleform

This UI framework from Autodesk (Autodesk Inc., 2015) combines the power of Adobe (Adobe Systems Incorporated, 2015) Flash together with a UI programming API. It supports development of interfaces for multiple environments, like PCs, consoles and mobile devices. It can be used in combination with a major game engine like (Unity Technologies, 2015) and also on common platforms (Autodesk Inc., 2014b). With dedicated workflows and tools for UI creation, it can be used for functional prototypes easily, if the previous prototypes and concepts were already made with Flash (Autodesk Inc., 2014a).

4.3.5 XNA Studio

The XNA framework is part of the XNA Studio from Microsoft (Microsoft Corporation, 2015a), which unites several programming interfaces related to games, like *Direct3D* from *DirectX*, *XACT* and *XInput*. It is primarily designed to support Microsoft operating systems and devices like the *XBox* or *Windows Phone* (Microsoft Corporation, 2015b). The author used the XNA framework several year ago, it supported UI prototyping only at a very functional level, requiring lots of programming to produce visual artefacts; to evaluate the current version was out of the projects scope.

4.3.6 Ogre3D

Although this is primarily a graphics engine rather a game engine, it provides concepts for graphical user interfaces, together with some third party libraries (like *CEGUI* (CEGUI, 2015)). Based on *C++*, *OpenGL* and *Direct3D 9 & 11*, the engine is object-oriented designed with the sources available and targets several platforms (and compilers). There is no dedicated IDE for the development, thus no world editor, UI designer or animation editor is provided. The extendability through other libraries makes Ogre3D very flexible, but the lack of tools makes it hard to be used for prototyping (OGRE, 2015).

4.4 Selected Prototyping Tools

The selection of tools requires consideration of the transitions between them. A board-game prototype is different to transfer into a actual game implementation, than a paper prototype. Of course the final platform(s), input devices, and software legacy has to be considered too. The games genre also influences what tools are appropriate, a simple 2D side-scroller does not need extensive 3D features. Experimental and revolutionary approaches for inventory concepts need tools that offer productive options for custom designs.

4.4.1 Pen-and-Paper Sketches

Being able to execute naturally and with low resources, the early usage of pen-and-paper sketches support creativity and flexibility. The concepts should be supportive for the player involvement, and combine with the game world. Design concepts can be digitized and create the base for further digital iterations. During paper prototyping, they are useful to make new design scribbles, and to enhance paper prototypes manually during the session (additionally, see section 4.2).

4.4.2 InDesign

For increasing fidelity, vector based tools are from great usage for multi resolution designs; they scale without any quality loss. *InDesign* and *Illustrator* offer various tools for vector graphics and export to all common formats. The space for design implementations covers only 2D elements; nonetheless, audio can be considered at this point. For more flexibility beyond layout, fonts, icons or colours, *Flash* could provide a early implementation for UI animations and simple functionalities. The author already mentioned, there is insufficient knowledge of this tool and the present knowledge of *InDesign*, in section 4.2.

4.4.3 Unreal Engine 4

Despite the fact, that the engine is used for the game “Caede”, the expectations for the *Unreal Engine* are high. The design flexibility suggested by the *Unreal Motion Graphics* editor seems promising for fast and easy implementation of functionality. As mentioned in section 1.6, the author has no experience with UMG, the feature was added as early access beta in August, 2014 (version 4.4). The current version (4.6.1) is stable, and the UMG editor is in a production worthy condition.

Concepts created with *InDesign* can be exported easily as pixel graphic for usage in the *Unreal Engine*. Changes on designs (e.g. icons, panels or fonts) can be applied by just reimporting the (texture) asset, exported by *InDesign*. The layout created in *InDesign* has to be rebuild in UMG, with the provided structuring elements (e.g. Overlays, Grid-Layouts or Scroll-Boxes) (Epic Games Inc., 2015g). Apart from those features, that are common for prototyping, the UE4 introduces event based creation of behaviour with an easy accessible and learn-able editor: the *Blueprint Editor*. The additional UMG editor blends seamless into the framework; including the connection of UI events, input and functions, from the blueprint editor, with UI elements in the GUI layout. Different input concepts like touch, game and motion controllers, HMDs and the traditional mouse/keyboard are supported with various pre defined events. In general the engine forces “doing things right”,

Chapter 5

Game User Interface Design & Usability Evaluation

> *"A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."*
>
> Douglas Adams

In addition to the prototyping methods from chapter 3, this chapter dedicates to UI design and usability evaluation. Both are connected to the design and development process targeted with the project; a evaluation has to take place for resulting UI prototypes. The chapter concludes with a discussion on potential methods for the project.

5.1 Game UI Design

The user interface (UI) is the communication channel between game and player (Llanos and Jørgensen, 2011). It is responsible, for the way the game is actually played; and therefore, the relevance and role of the user interface, needs to be considered at any time in the game's development process (see Fricker, 2012, p. 20).

5.1.1 Understanding Goals, Purpose and Role

The common way to separate games is defining genres, like it is done for books, music or films. The genres cover a certain set of game mechanics, have differing goals and purposes; a competitive online game has obviously another purpose than a point-and-click adventure.

With this in mind, the role of the UI will be a different one, depending on the genre. But since "all games are different" (see Fricker, 2012, p. 20), there is no formal definition to create, e.g. a role playing game UI from scratch. The genre standards can be used as a guideline, nevertheless the individual game-design and -mechanics overlap with various aspects like feedback, control, aesthetics and cognitive load, which is also covered by the UI (Fricker, 2012; Saunders and Novak, 2012; Pagulayan et al., 2003). Where the immersion is factor of the UI, that has to be considered; player involvement is complex and has consequences for UI design, regardless what form (Llanos and Jørgensen, 2011).

5.1.2 Feedback, Control, Design Aesthetics & Cognitive Load

The purpose of the UI is feedback and control; the feedback gives the player information about the game: what options are available, what is the current state of the player in the game, and what effect actions have. The control is the back-channel, offering access to the game, giving the player the control to influence what is happening in the game world. The control is usually available through a peripheral device; like a keyboard, mouse or game-pad.

Information that is communicated between player and game can be very complex, thus consistency of feedback and control is very important. An obstructive UI and inconsistency or more general: lack of usability, accessibility, functionality and aesthetics, can make a game less fun for the player (e.g. missing information about errors, progress, state, etc.) (Fricker, 2012; Saunders and Novak, 2012; Llanos and Jørgensen, 2011).

Fagerholt and Lorentzon offer three categories for feedback in FPS games: visual, auditory and haptic (Fagerholt and Lorentzon, 2009). The visual category involves 2D overlay graphics (the common HUD), filtered graphics (like blood-splatter graphics if the player character is injured) and in-game world (animations or textures to indicate progress and state). Auditory feedback covers sound-effects (like hit-sounds or the simulated heart-beat on low health), dynamic soundtrack (e.g. indicating enemies are nearby) and dialogues and speech (e.g. NPCs giving hints on goals or secrets) (Fricker, 2012). The haptic feedback, requiring the touch of the player (a game-pad or smart-phone), can be casual feedback (e.g. for a weapon hit) or decision making; like on low health a heart-beat, communicated with vibrations.

Furthermore, Fagerholt and Lorentzon introduce the design-space model (see section 2.2.7), which can be used to design or categorize UI elements according to the fictional and spatial dimensions. For FPS games, they identify six categories in design space: HUD elements, geometric elements, diegetic elements, meta-representations, meta-perception and signifiers (see Fagerholt and Lorentzon, 2009, p. 51-52). They are pictured in figure 5.1 according

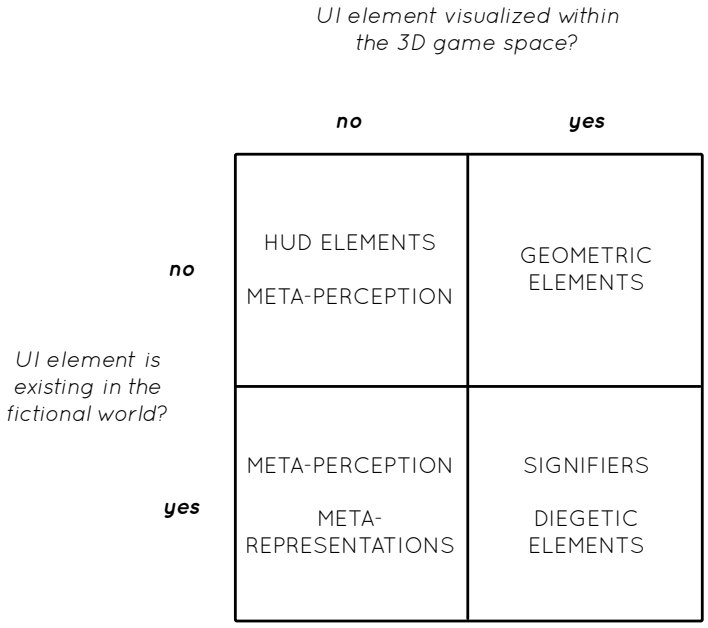


FIGURE 5.1: Categories of UI elements (for FPS games) and how they fit into the design-space (see Fagerholt and Lorentzon, 2009, p. 51)

to the fictional and spatial dimensions. Those categories were found for FPS games, but they apply to other genres as well; obviously feedback and control are independent from the genre.

Since games usually define their own world, the aesthetics of a UI contribute to the experience of this world: "how the game is seen, felt, heard, tastes and smells" (see Fricker, 2012, p. 23). Fricker offers some more details on related work for design aesthetics; those cover aspects like sizing, color-coding, typography, icons, animations and audio. The primary purposes are to differentiate between information and its relevance, to underline the games mood and atmosphere, accessing multiple channels for informations to optimize the cognitive load. They can be used as guidelines for a clear, unobstructed, customizable and consistent UI, that is intuitive and easy to learn and use.

5.1.3 Integrated UI

An integrated UI uses diegetic elements and can improve the experience for more player immersion (Stonehouse, 2014). Diegesis describes the world a story is set in, film theory also refers to the internal world, created by the story the characters experience (see Fagerholt and Lorentzon, 2009, p. 18). Whether integrated UI can increase the player's involvement is still discussed (Llanos and Jørgensen, 2011; Fricker, 2012; Russell, 2011; Papalamprou,

2014; Andrews, 2010; Stonehouse, 2014). But the trend towards more UI integration are ongoing and visible over several genres, some examples can be seen in figure 5.2.

Where diegetic elements can be slow in response to player actions, they provide a much more realistic representation and can be appropriate for certain genres e.g. survival games. Nevertheless they are used in more futuristic narratives like *Splinter Cell*, *Syndicate* or *Mirror's Edge* with success, where they are accepted much easier (Stonehouse, 2014). The game's design, mechanics, narrative, lore and setting can be used to decide if a integrated UI is beneficial; still, feedback and control need to be consistent and intuitive (Andrews, 2010). If the UI is integrated and appears in an aesthetic look and feel, but is barely usable, giving less or no feedback at all, it is counterproductive for the players experience and involvement.

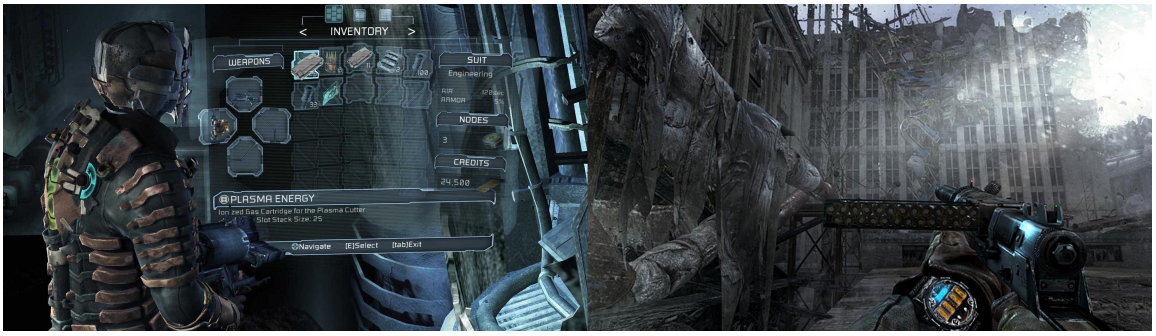


FIGURE 5.2: Diegetic UI elements in *Dead Space 2* from *Visceral Games* (Electronic Arts Inc., 2015) and in *Metro Redux* from *4AGames* (4A Games, 2015; Deep Silver Inc., 2015)

Llanos and Jørgensen provide some more details on integrated UI, and conclude, that they need to display relevant and sufficient informations for the player, in order to interact with the game (Llanos and Jørgensen, 2011). They also relate to player involvement or immersion, as being a "complex and multidimensional phenomenon" (see Llanos and Jørgensen, 2011, p. 10), depending on the game and genre specifics and on the player's preferences and attraction to games. While a minimal and integrated UI may be more aesthetic, a superimposed UI provides information in a more structured view and is still common and achieved with less effort.

5.2 Usability Evaluation

The design and creation of a system has to come with a process of evaluation, to find shortcomings in the design. Those result in changes, that can be evaluated again (Bowman et al., 2001). There are various techniques for usability evaluation, some were discussed in section 2.2 and are further discussed by Lee and Conyer in detail (Lee, 1999; Conyer, 1995). Usability experts Nielsen, Norman and Desurvire et al. offer more on usability

inspection and evaluation (Nielsen, 1994b; Nielsen, 1994c; Desurvire et al., 2004; Desurvire and Wiberg, 2009; Federoff, 2002).

Fricker distinguishes between *Usability Testing* and *User Testing*, where the first requires expert evaluators conducting the test, and user testing focuses on the user, performing the evaluation, during use of the product (Fricker, 2012). Both are used in game development, often in combination with prototyping, where about 5-7 play-testers are used and represent the target-group (Laitinen, 2005; Fricker, 2012). The length of a test-session varies from one to two hours and is beneficial to understand the player, and to understand if he is able to comprehend the game-ideas in early development.

Heuristic Evaluation is researched by a lot of experts, coming from HCI and game development in particular. Nielsen poses usability guidelines for non-entertainment software, but Desurvire et al., Federoff and Pinelle et al. offer heuristics especially for games (Desurvire and Wiberg, 2009; Pinelle et al., 2008; Federoff, 2002). Heuristics are used by evaluators to explore an interface, they are common usability principles and guidelines for better usability. Heuristics need to be customized for genres or even games, they need to focus on playability and engagement as well as usability, experience and immersion (Pinelle et al., 2008). Typical use for heuristics is the *Usability Expert Evaluation*, already described in section 2.2.3, other methods found and studied in literature are outlined below:

Empirical Testing For this method, experimental tests generate data to prove or disprove a hypothesis, which can be measured regarding a set of objective variables. The outcomes are quantitative results, which can reveal cause and effect (Conyer, 1995).

Pluralistic Walkthroughs They aim to review the usability and flow of an interface formally from a users perspective. Evaluators document in detail, how they would accomplish a certain task; in scenarios with users those are evaluated, writing down the steps to complete the tasks. After a discussion between users, developers and HCI professionals, the results are presented and are the base of further design enhancements (Conyer, 1995).

Focus Groups This technique can give user-centred feedback, which is gathered during play-testing and user-discussions. The combination with surveys and usability testing is common and group discussions on game, its features and the final product between participants can give valuable feedback on possible new features, balance or content (see Pagulayan et al., 2003, p. 30).

Formative Playtest Surveys The combination of hands-on play and surveys can be used to gain "specific information about how consumers perceive critical aspects of a game and provide actionable feedback to game designers" (see Pagulayan et al., 2003, p. 26 ff). The survey combines forced choice questions together with open-ended questions in order to find reasons for the players answers.

Formal Design Analysis By decomposing tasks into goals, operators, methods and selection rules, the requirements and behaviour can be understood, to improve the design process. The method aims on expert users, but the designs are rated by algorithms, that generate numeric results. Further designs or design-alternatives can be rated and compared against prior results (Conyer, 1995).

Formative Evaluation Like *Formative Evaluation* or the *Formal Design Analysis*, this method aims on gathering quantitative and qualitative data, to evaluate and improve the designs from early development phases "throughout the entire life cycle." (see Hix et al., 1999, p. 3-4).

Formal Usability Inspections This method is performed by evaluators, to "perceive the information, plan to use the information, decide how to proceed and perform the selected action" (see Conyer, 1995, Formal usability inspection). It is structured in six steps: Planning, Kick-off meeting, Preparation, Review, Rework and Follow-up; to asses the effectiveness of evaluation. The evaluation occurs with user-profiles and pre-defined scenarios, using performance tests and heuristics.

5.3 Method Discussion

It is quite obvious, that prototyping, UI design and usability evaluation go hand in hand, as displayed above and in chapters 3 and 2. The combination was performed with success by Fagerholt and Lorentzon, Guldbrandsen and Storstein or Ferderoff (Fagerholt and Lorentzon, 2009; Guldbrandsen and Storstein, 2010; Federoff, 2002).

Prototyping and usability evaluation methods overlap; the evaluation is a mandatory part of a prototyping process, but can be used on a final product nevertheless. But like the latest trends show, the improvement of a software product continues the whole life-cycle. Programs can auto-update itself, adding new functionality, content or patches, improving the product continuously. Thereby the updates need to be evaluated before the distribution occurs; it can be said, that the prototyping process never ends, and is more a progressive

evolution after a critical point is reached. A good example is *Dota 2* (Valve Corporation, 2015b) by Valve (Valve Corporation, 2015a), the game's development started in 2010, entered open beta in late 2011 and was released in June 2010; but *Valve* steadily improves its features, content, balance and eliminates errors. The monetization of *Dota 2* is done by selling new character models, animations, voices, etc.; the game itself is for free. Since game assets are the source of money for most free-to-play titles nowadays, the development of content and features never stops entirely. Thus, the development of continuously evolving products, needs an accompanying process of (rapid) development, testing/evaluation and improvement.

Game UI design guidelines and the research on design-space are valuable along the process, nevertheless they have most impact in early development, in design and concept phases. Since this project targets the design of UI from the scratch, the development process aims to consider the findings displayed in section 5.1. Different design-space approaches are planned, and the usage of heuristics to evaluate a UI may be an affordable method, regarding time and scope of the project (Fagerholt and Lorentzon, 2009; Federoff, 2002; Fricker, 2012).

The thesis' focus lies more on qualitative evaluation rather than quantitative, to gather ideas and insights on early designs; but quantified data is collected along the evaluation process. The *Formative Playtest Survey* method used by Pagulayan et al.: a combination of *Focus Groups*, *Cognitive Walkthrough* or "playtesting" and *Survey*, seems adequate to use for a first prototype evaluation (Pagulayan et al., 2003).

Chapter 6

Case Study: Game UI Prototyping

> *"I like the cover," he said. "Don't Panic. It's the first helpful or intelligible thing anybody's said to me all day."*
>
> Douglas Adams – The Hitchhiker's Guide to the Galaxy

This chapter will define and document the UI design and prototyping process used in this thesis; it also relates to several other processes found in literature review, which were discussed earlier in chapter 3 and 5. Furthermore it also defines the requirements for each stage in the prototyping process, and dedicates to the documentation of pre-prototype, warm-up, refinement and functional implementation stages, of the process defined first.

6.1 Defining the Prototyping Process

Because a role-playing game (RPG) user interface (UI) is a rather complex system, this thesis focuses only on a specific sub-part: the *character inventory*. Nevertheless, other parts of the UI may also be considered, if they influence the design or functionality. Before the prototyping can start, an overview on RPG UI is required, to understand the different UI approaches, existing in other games. The RPG UI research is part of the idea generation process, and they inspire the overall design process. Early pen-and-paper sketches will illustrate the first UI concepts, which are the base of further digital concepts.

The concepts will be prepared for the almost medium-fidelity paper prototyping in the first session, and the findings will be used to eliminate dead ends in the design, and to improve it, based on the feedback from the first session. The improved version, with medium to high fidelity is discussed and evaluated in a team prototyping session, and leads to a final

design concept. That will be implemented as a functional prototype within the current game project of the Unreal Engine 4.

6.1.1 Towards a Process Definition

A process for a UI prototype of a game has different aspects to consider. The applied prototyping techniques, usability evaluation, user-centred design or player involvement are only some of them; relating theories, methods and techniques were discussed in the prior chapters. As outlined before, a rapid prototyping process with evolutionary aspects suffers the required flexibility in early phases (Guldbrandsen and Storstein, 2010). Evaluating the results of each iteration is crucial, in order to compare prototypes of the same level of iteration and between consecutive prototypes. This is done to filter and discard unpromising designs along the process, to reduce many early approaches to only one final approach. Changing requirements, discovered during evaluation, in surveys, discussions and usability tests, are common in game and software development. Evolutionary and rapid techniques, and user-centred usability testing, were found as supportive for the adjusting requirements (see section 5.3 and 3.5).

Starting the prototyping, requires a subject to prototype with; this dilemma affects prototyping in general. Since, there exist no usable UI designs at the opening of the project, a preceding study and design-phase is necessary. Once, designs are produced in a sufficient amount, evolutionary iteration can start to increase the prototypes fidelity. Is a certain level of fidelity reached, the prototyping has to continue in the UE4, implementing functionality and covering more dimensions (like audio or animation feedback, interaction and controls). From the initial design phase throughout the iteration with the UE4, development of new game features, game design or mechanics, changes in the code-base, new models or animations, takes place continuously. A UI prototype must be able to adapt and integrate changes easily at each phase. It follows a description of requirements, posed for the desired prototyping process:

What is the state-of-the-art for game UI? In order to answer this question, past and current RPGs can be reviewed and studied with focus on the user interface. Surveys, interviews and discussion with developers or players on developed and played games and their UI features can be beneficial too.

Support creative processes during the development; because innovative ideas can emerge better, without restrictions of specific executions. *Paper Prototyping* and *Team Prototyping* allow creative discussions and fast concept creation during the actual testing (see chapter 3). Being unable to implement and evaluate functionality

can be a benefit, instead of a disadvantage, because imagination and ideas can shape experience better than any low-fidelity prototype.

Design the interface with the user and thus, more for the user. User centred design (see section 2.2.5) approaches like *Focus Groups* and *Cognitive Walkthroughs* (see sections 5.2 and 2.2.4) were used in related projects.

Find requirements for further iterations from prototyping results. The findings and feedback of each session are used to refine the designs for the next one.

Find player-relevant UI features, to compress the requirements into significant statements and hypotheses for changes and decisions.

What prototype or feature needs further iteration must be decidable at the end of each prototyping. The decision should arise in collaboration of players and designers where possible.

Find immersion-relevant UI features, to prevent breaking the players immersion or involvement. This covers, where extensive feedback is needed, where "less is more", what type (auditory, visual, haptic) or combination of feedback is appropriate, as well as controls, timing and whether help should be displayed.

What can raise the immersion? Are there game aspects that need to be handled with more care, in order to provide a better experience? The process must be able to identify those and enable a different treatment.

Allow parallel and asynchronous development of content, features and functionality. Game design, UI design and actual implementation are developed independently until a certain degree; interfaces between the different iteration states and fidelities should support this during the prototyping process.

6.1.2 Formal Process Definition

This section defines the process formally, performing the process starts in section 6.4. It is structured in four stages, the *Pre-Prototype Stage* is a creative idea generating step, that produces the first designs that are prepared for initial prototyping. It follows the *Warm-up Stage*, where the first actual prototyping takes place; it eliminates dead-end design approaches, and with the findings, the remaining designs are improved for the next stage, the *Refinement Stage*. In this one, another step of elimination takes place and only one concept should remain for further iteration. During the *Functional Implementation Stage*, a single functional prototype is implemented with the UE, and also followed by a evaluation. The thesis will cover every stage, up to and including the functional prototype evaluation;

nevertheless, the prototyping will continue beyond the scope. The process is described and defined below:

1. Pre-Prototype Stage

Idea generation: Use creativity methods, state-of-the-art examples, interviews, surveys, (game-)reviews.

Develop concepts for the UI: Two to four should suffer, too much concepts will slow down the iteration phases.

- Do not be too specific for the interaction concept
- Feel free to make extreme revolutionary or evolutionary approaches
- Consider multi platform usage if possible (controls, screen sizes, haptic feedback, head mounted, mobile, web-services)
- Use user interface design guidelines
- Keep feedback, controls, aesthetics and cognitive load in mind

Choose your tool and technique: What will be used for prototyping and evaluating the concepts? Decide and prepare the design for the prototyping.

2. Warm-up Stage

Prototype evaluation: Use qualitative methods, like surveys, interviews in combination with usability evaluation or other qualitative techniques.

- Gain insight of your participants thoughts, favours and dislikes
- Level of expertise: we need at least semi-expert players
- Measure experience in gaming and genre
- Identify design and usability issues
- Identify improvements and requirements

User-centred aspects: important along the whole process

- Motivate participants by being open with concepts, design problems known, discuss game design in detail and being generally open to their thoughts and ideas
- Force to "think out loud", assume users as part of the development process for more and (potentially) better results
- Designing for and with the user for better acceptance
- Introduce ideas early and thus explaining them in order to spread new ideas and inspire/provoke
- Discuss player involvement and its relevance depending on features or specific game-design and -mechanics

- Consider: cost and time heavy, same users needed for later stages, strong dependence on testers (QA), best practice needs 5-7 testers

Collect and improve design: Evaluate results, discard concepts, make suggestions, improve and refine the remaining concepts according to findings and user feedback.

Choose your tool and technique: What will be used for prototyping and evaluating the concepts? Decide and prepare the design for the prototyping.

3. Refinement Stage

Prototype evaluation: Use qualitative and collaborative methods (like e.g. team prototyping). Identify design and usability issues, improvements and requirements.

Collect and improve design: Evaluate results, identify only one remaining concept; make suggestions, improve and refine this concept again, according to findings and user feedback.

4. Functional Implementation Stage

Implementation with UE4: Develop a functional prototype for the remaining concept, using the Unreal Engine 4.

Prototype evaluation: Perform a usability evaluation with the functional prototype for 5-7 users. Define requirements from preceding evaluations and perform empirical tests, use heuristics, conduct formative play-tests or formative evaluation – combine qualitative and quantitative evaluation (see section 5.2).

Collect and improve design: Evaluate results, make suggestions, improve and refine the prototype in a evolutionary manner with findings and user feedback.

Improve implementation: enhance and integrate the improved UI design.

Rapidly repeat, until evolutionary goals are reached: iterate the design in a rapid and evolutionary prototyping process. Define requirements/goals for prototype features, iterate them along with the prototype.

6.2 Game UI Prototype Requirements

The requirements for the UI prototype will evolve during the process in detail by design. The currently identified requirements are displayed against the different stages defined before. For the pre-prototype stage, sufficient designs need to result for further prototyping. In the second stage, the first iteration step needs to identify a significant amount of design

hypotheses for the remaining concepts. Whereas the second iteration needs to extend and refine them further. After the first three stages, the hypotheses gathered define the requirements for the functional UI prototype. The fourth stage, the functional implementation stage, will continue beyond the projects scope. Therefore, not all requirements identified after the third stage can be achieved. Still, the current defined requirements will evolve along the rapid evolutionary prototyping process.

TABLE 6.1: Requirements for the prototype and process

Stage	**Requirement**
Pre-Prototype Stage	Sufficient designs need to result: 1a. Create three designs for an inventory UI. 1b. Prepare them for prototyping.
Warm-up Stage	Identify a significant amount of design hypotheses: 2a. Find 20 issues, ideas and suggestions. 2b. Eliminate the least sophisticated design.
Refinement Stage	Extend and refine design hypotheses: 2a. Find additional 20 issues, ideas and suggestions. 2b. Find a final design for prototyping with the UE4 2c. Identify requirements for the functional prototype.
Functional Implementation Stage	2a. Implement requirements defined in refinement stage. 2b. Progressively enhance and adjust requirements.

6.3 Study of Existing UIs

The focus of this section lies upon existing UIs in RPGs and other cross-genre games, in order to get an overview of possible standards, to inspire the design process and to examine how the UIs evolved over time. The whole study can be seen in appendix C.

6.3.1 Considerations

The UI study focuses mainly on RPGs, nevertheless there will also be other games and genres in focus of the study, but are not discussed at this point. The selected games are popular RPG series, but are chosen by the authors personal favours. In the study, not only inventory UIs were reviewed, but also other parts of the chosen games UIs. This was done to leave the UIs in their context and focus not only on certain parts. This seems highly necessary to the author, when looking on games, to get the right experience. Especially

RPGs draw their immersive world out of many aspects, and the design of an inventory has to consider those various aspects.

6.3.2 Game subjects

The selected subjects of the study are some single titles and from game series: Gothic and Risen from *Piranha Bytes* (Pluto 13, GmbH, 2015a), The Witcher from *CD Project RED* (CD PROJEKT, S.A, 2015a) and The Elder Scrolls from *Bethesda Game Studios* (Bethesda Softworks, LLC, 2015). The titles were played on the PC platform, with keyboard and mouse, but some of the titles are available on other platforms, with other input devices as well. It follows a rough analysis of the game titles in respect of UI with focus on the inventories the games provide.

Gothic This series has a long tradition, the first part was released back in 2001 for the *Microsoft Windows* platform and received a lot of credit amongst RPG fans and the press (CBS Interactive, Inc., 2015a). Also the second part followed in this tradition and was first released in late 2002, the add-on *Night of the Raven* followed in 2003.

The fact that *Gothic*, *Gothic II* and its add-on are based on the same game-engine is responsible that all three games share the same user interface. The UI of the *Gothic* series is definitely not among the best in this time and genre, but the game draws its fascination out of dense atmosphere and love for details. The visual interface lacks a dedicated theme and is more functional than appealing, but it is mostly consistent and provides all necessary informations to make progress throughout the game. The concept for interaction is not really intuitive but again consistent. The audio feedback supports the atmospheric aspects of the game and is also used as user feedback (e.g. exhaustion, pain/hits, sleeping, aggression) from the player character as well as NPCs 1. Looking at the inventory in particular reveals a lack of categories or other sorting options; all items sort into a regular grid, in which each cell holds a stack of all items of the same kind, showing the amount with a number. The details displayed for an item contain many text, attributes and a 3D model of the item itself; there is no usage of icons, and a rare usage of colour within the UI.

In *Gothic III* the interface gets a new face and is restructured. For the inventory in special: there are categories as sorting option for the characters items, there is a consistent colouring for e.g. items one can or cannot use, and there are icons to show additional properties, like the poison droplet. Still being consistent, the interface in a whole stays more functional, than appealing. To classify the interface according to the model of design space by Fagerholt and Lorentzon, the UI consists of non-diegetic element-types for mostly everything visual

(Fagerholt and Lorentzon, 2009). However, if the audio feedback is much more diegetic like the exhaustion mentioned above, taking place in the fictional world as well as being feedback for the player (Pluto 13, GmbH, 2015c; Pluto 13, GmbH, 2015b).

Risen With *Gothic* in mind, this new series from *Piranha Bytes* is very close to its predecessor, also in terms of the UI. One can probably comprehend it as another iteration in their tradition of creating role-playing-games. From this point, there are indeed many elements, from especially *Gothic III*, appearing in a new look, which is much more themed, has more contrast and vividness.

The player's main HUD is restructured, the weapon slots and other item quick-slots are separated, in contrast to the slots from *Gothic III*. In the latest release, the third part of the series: *Risen 3: Titan Lords*, the UI is contributing to the game's theme, is well structured and uses high-resolution elements with some more details, without being too obstructive. Feedback is gathered in a log, in the player's HUD, and in a dedicated view onto the current objectives/quests. Input options, e.g. during a fight, are displayed in the according context. The compass from the first two parts disappears and is replaced with a mini-map.

From the first game from *Piranha Bytes*, up to the latest, the continuous improvement of RPG UI elements can be observed over quite a long period. And more than a decade later, the clumsy and monotonous UI from *Gothic I* shows pretty clear, how the standards evolved (Pluto 13, GmbH, 2015c).

The Witcher This series from *CD Project RED* (CD PROJEKT, S.A, 2015a) is based on the novels from *Andrzej Sapkowski* and the first part: *The Witcher* was released in 2007 (CD PROJEKT, S.A, 2015b). The successor *The Witcher 2: Assassins of Kings*, released in 2011, both titles were successful and received good critique amongst players and press (CBS Interactive, Inc., 2015b; CBS Interactive, Inc., 2015c).

Despite the fact, that the first and second part nearly contain the same game mechanics, there are differences in the UI. Like the *Gothic* and *Risen* series, it is more an iteration than a completely new approach. Compared to *Gothic III* the UI of *The Witcher* is much more loose, is less opaque and has more decorative, thematic and story-related elements. It can appear more obstructive having too much details, but the interface is consistent in colours, sizing, typography and structure nonetheless.

Both titles use animations to support the players actions, e.g. the protagonist goes onto its knees while crafting items or meditation. Where *The Witcher* uses multiple icons within the HUD, to visualize the characters current state, the second part reduces the amount of

FIGURE 6.1: RPG evolution – from top-left to bottom-right: Gothic I, Gothic III, Risen and Risen 3

first level information. The currently selected weapon was displayed in the HUD of *The Witcher* as icon; in the successor, the representation is the drawn weapon in the game world, a much more diegetic approach.

The interface of *The Witcher 2: Assassins of Kings* contains less thematic details and appears less obstructive, not much interfering with the game world. The inventory, an obviously relevant part of any RPG, is structured intuitively, the menu structure is flat, most information is present at the first level. Inventory items can be filtered by category and sorted (e.g. by weight or value); the character can be equipped with cloth and weapons. Each item has a weight and value, also the current total inventory weight and money is displayed, and details onto selected items and equipped, in order to compare them.

The Elder Scrolls The series is currently on its fifth part: *Skyrim*, released in 2011, and is one of the oldest RPG series that continues till today. Where *Oblivion*, the fourth title from 2006, offered a very thematic interface in old parchment style; the latest part is designed much clearer, minimal and reduced.

The inventory of *The Elder Scrolls: Skyrim* grants access on items and character equipment. Along with item informations and character informations, items and the effects can be compared; the inventories total weight and the current in-game money is additionally

displayed. The game presents the current interaction options at any time, e.g. in a dialogue, the inventory or world map.

Because *The Elder Scrolls* games can be played in first and third person, in contrast to the titles above, the UI is mostly non-diegetic, presenting informations on the 2D HUD plane. Nevertheless the map of *Skyrim* is impressive, it is not a 2D representation but the 3D game world from another perspective.

6.4 Pre-Prototype Stage: Game UI Concept Creation

The prototyping process starts with the creation of three different UI approaches. When all concepts are finished roughly on paper, the further design will take place digital, with *InDesign*. Then the three concepts are prepared for paper prototyping, to evaluate the designs. After this session, one of the three designs is discarded; a throwaway attempt, to eliminate dead ends early.

6.4.1 Early Pen-and-Paper Sketches

The first scribbles were done in about 20 hours time and three different UIs resulted. Those are further referred to as *Table UI*, *Circle UI* and *In-game UI*.

The table UI is structured horizontal and vertical, like the name suggest in a table style. The first level shows the categories, also described as different inventory types (e.g. a herbal bag for the herbs category). The second level displays the appropriate item content of the inventory/category; offering item details in the third level.

The circle UI is inspired by the UI of *The Witcher 2: Assassins of Kings*, where the item quick selection is also designed in a circle. The inventories are arranged around the screen centre, the content is shown on the right side, where the currently selected is always rotated to the right.

The in-game UI is, contrary to the other two, not only placed on the HUD plane, but also in the 3D game world and follows a more diegetic concept. When in inventory, the displayed inventory- and item-details are selected by clicking onto the 3D world representation.

Where interaction concept was considered at this time, is was not designed with much detail. The concepts found in the game UI study were diverse, and the concept for interaction varies for all three designs. Finding an appropriate control scheme will take place after the first prototyping, since the results can help to decide what works and what the player desires. The concepts are displayed in detail in Appendix D.

6.4.2 Digital Iteration and Paper Prototype

In the digital implementation, the designs were revised and enhanced. Icons, fonts and colours were considered in addition to the layout from the pen-and-paper scribbles. The informations communicated with the inventory was iterated further; how to display amount, value, type and category, weight or quality. A set of icons for usage in all three UI concepts was created, some experiments with colours were performed.

The design aims for a more minimal UI, the usage of colours is limited and is used to indicate certain states, e.g. a green item-amount bar or text indicates the inventory is barely filled. The font used, called Bitter (Huerta Tipografica, 2015), is easy to read and is chosen to support the medieval setting. The icons are reduced to black and white only, having more contrast and little detail (see figure D.8).

While a paper prototype offers possibilities to test feedback, controls can be difficult to implement at this point. The concepts created in *InDesign* are easily transformed in a paper-prototype ready document to print and prepare for the session. In order to offer some more material to discuss, the character HUD was also included to the prototype, and a screen-shot of the current in-game perspective was prepared. The HUD was added, because the inventory is obviously not visible all the time, and the transition was meant to be captured by the prototype as well. The screen-shot is necessary for the in-game inventory concept and presents a scene the players can relate to. The digital designs are displayed in the figures D.5, D.6 and D.7.

6.5 Warm-up Stage: First Iteration

In the first evaluation, the three concepts are tested by expert users to compare the designs and to find out how the users interact with them. The observation of usage, generation of design hypotheses, discussions regarding the three implementations, and how the concepts can integrate into the game, are the goals of this stage. Methods, discussed in section 5.3, like the *Formative Playtest Survey*, will be used in combination with a paper prototype, see section 3.5 for details.

6.5.1 Conducting the Evaluation

The evaluation is done with a short survey on the players habits and likings. It takes place in a living room, for a realistic environment. The author is in the role of the interviewer, observer, scribe and conducts the paper prototyping session. The participants need to be expert RPG players, and the questions are designed to give information on the level of

expertise. Additionally it follows a semi-structured interview-survey combination, where the players have to rate statements relating to game UI and are free to debate them. Discussions will be captured with notes and aim to rate the preferences of the participants. If a player proves not to be an expert user, the session will end without any prototyping. The complete survey and the statements can be found in appendix D.

When the statements are all rated, the paper prototyping starts; the order in which the three prototypes are presented is randomized to eliminate a possible learn-effect and bias. In order to simulate the input, a mouse and a keyboard are additionally present. The players are confronted with the paper prototype; they initially find themselves with a screen-shot of the game. They are encouraged to think aloud, their actions are questioned to find reasons for behaviour. The actions involve simple tasks, like opening the inventory, navigating through it, selecting items, or retrieving certain item or inventory informations. Usability issues, new ideas, improvements and other feedback on the game and design, are noted.

Table 6.2: Session 1 – Participant overview

Subject	**1**	**2**	**3**	**4**	**5**	**6**
Age	22	25	24	27	27	25
Gender	male	male	male	male	male	male
Max. Playtime/ Day (in hours)	3	1	3	2	3	4
Top 3 Genres	RPG, Action Adventure, FPS	RPG, Action Adventure, Adventure	Action Adventure, RPG, MMO	Strategy & Tactics, RPG, Simulation	FPS, Competitive, RPG	RPG, Strategy & Tactics, MMO
Session Duration	2:31	3:12	1:47	2:26	1:43	2:17

6.5.2 Findings & User Feedback

The evaluation was performed with six participants, all were male and between 22 and 27 years old. The average playtime per day was 2.7 hours, all of them primary play on the PC platform and the RPG genre was always in the top three of favourite genres (see table 6.2). While the statements were rated differently, there was a consensus that "A good UI explains itself." and "Less UI is better UI.". Also consistency of colours, fonts, sounds and icons was rated for high importance. The statement discussion was very instructive, giving insights on the different requirements, desired by potential players and how they get

involved by (RPG) games. This was valuable for the further paper prototyping, because actions can be compared with appropriate statements for more qualitative results. The complete quantitative data can be found in appendix D, as well as the scribbles created along the session (see figure D.9).

The notes taken during the session were compressed, the resulting findings are listed in the tables D.1, D.2, D.3 and D.4. They are issues, insufficiencies, improvements, suggestions and ideas, that were stated by the players during the survey and paper prototyping. Where the according participants, relating to the finding, are displayed before each of it. Where 35 findings concern all three UI prototypes, shown in the general findings table D.4, and relate to different categories: user interface, game design or game mechanics.

The results for the table, circle and in-game UI are all UI specific; together with the general findings they form new hypotheses for further designs and prototypes. The amount of findings is immense (nearly 50 suggestions, issues and thoughts), compared to the time and resources available; in order to deal with them, refined set of hypotheses is required.

6.5.3 First Refinement

All participants rated the table UI prototype as the least desirable, the layout was seen as inefficient, navigation is clumsy and interactions are not intuitive. Circle and in-game UI were both seen as promising, the combination of both was also suggested. The findings for the table UI are valuable nonetheless; but the actual re-design can get very extensive, and testing, if it will cope the issues, requires a functional implementation. Thus, the decision, which prototype is thrown away, was not complicated. The design hypotheses and suggestions generated by the prototyping session are refined further, separated into specific ones for the circle UI and in-game UI, and both:

Circle Inventory Hypotheses

H1.1.1: Re-design as static circle, with only 2nd level information.

H1.1.2: Provide customizable slots for inventory categories and items/weapons.

H1.1.3: Use direct mouse interaction, providing drag-and-drop, hovering and different focuses.

In-game Inventory Hypotheses

H1.2.1: Remove 2nd level information.

H1.2.2: Combine and reuse approaches from circle inventory where possible.

H1.2.3: Provide a interaction concept.

General Inventory Hypotheses

H1.3.1: Offer current interaction options.

H1.3.2: Use numeric values for item amount, plus icons for item consumption feedback.

H1.3.3: Remove or at least reduce colours used in UI, add appropriate in later iteration.

H1.3.4: Add item descriptions.

H1.3.5: Use more icons.

H1.3.6: Highlight current selected inventory-item.

H1.3.7: Display current hand item.

The items listed above, will be evaluated again, after a further iteration. The scope reduction after each stage was definitely a good decision; developing all three concepts, would go beyond the scope of this thesis. The designs for the circle and in-game UI are enhanced according to the design hypotheses, the progress is displayed below, in the figures 6.2 and 6.3.

6.6 Refinement Stage: Second Iteration

The improved design is shown in the figures E.2 and E.3, and additional infos can be found in appendix E. Improvements were made for the circle and in-game UI, according to the hypotheses from the preceding evaluation in about ten to 15 hours. Some of them were considered as being not relevant at the early stage, and will be taken into account in later prototype designs. The selected prototype evaluation is the mentioned *Team Prototyping* technique (see section 3.3.7). It requires experts, the testers from the previous session, are familiar with the game, its concept and design, and the author assumes, they are experts, for the sake of the evaluation. The participants, according to table 6.2: number 2 and 3, were invited for another session. The prototyping aims for a decision between the two remaining concepts, which proves as the most sophisticated, and what suggestions for future improvements and current issues will result.

6.6.1 Conducting the Evaluation

The location and setting is the same as in the first session: a living room, this time a beamer, wireless mouse and keyboard are used together; the session lasted about two and a half hour. The combination of proper tools has proven as very efficient: *InDesign*, the beamer

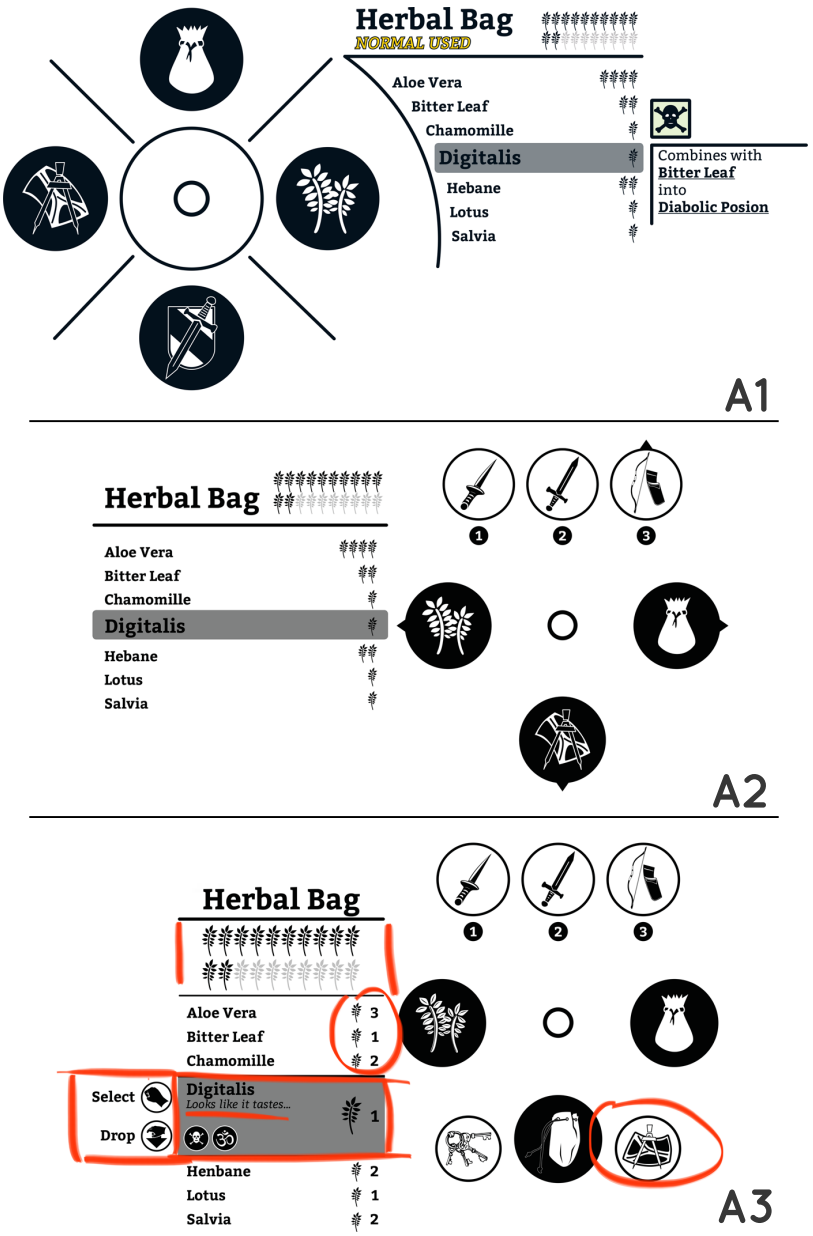


FIGURE 6.2: The refined circle inventory UI design after session one

and a recording software made discussions, on-the-fly changes and enhancements fast and easy feasible. Together with the two participants, the author acts again as interviewer, observer and designer; the session is audio-recorded and resulting UI design changes are saved. Due to previous participation, both have a detailed knowledge and need no extensive introduction. During the team prototyping, the UI elements are moved, toggled in visibility, rearranged, adjusted in size, colour or font. Both designs can be compared on the screen, and discussions on combinations, advantages and disadvantages of designs, can arise.

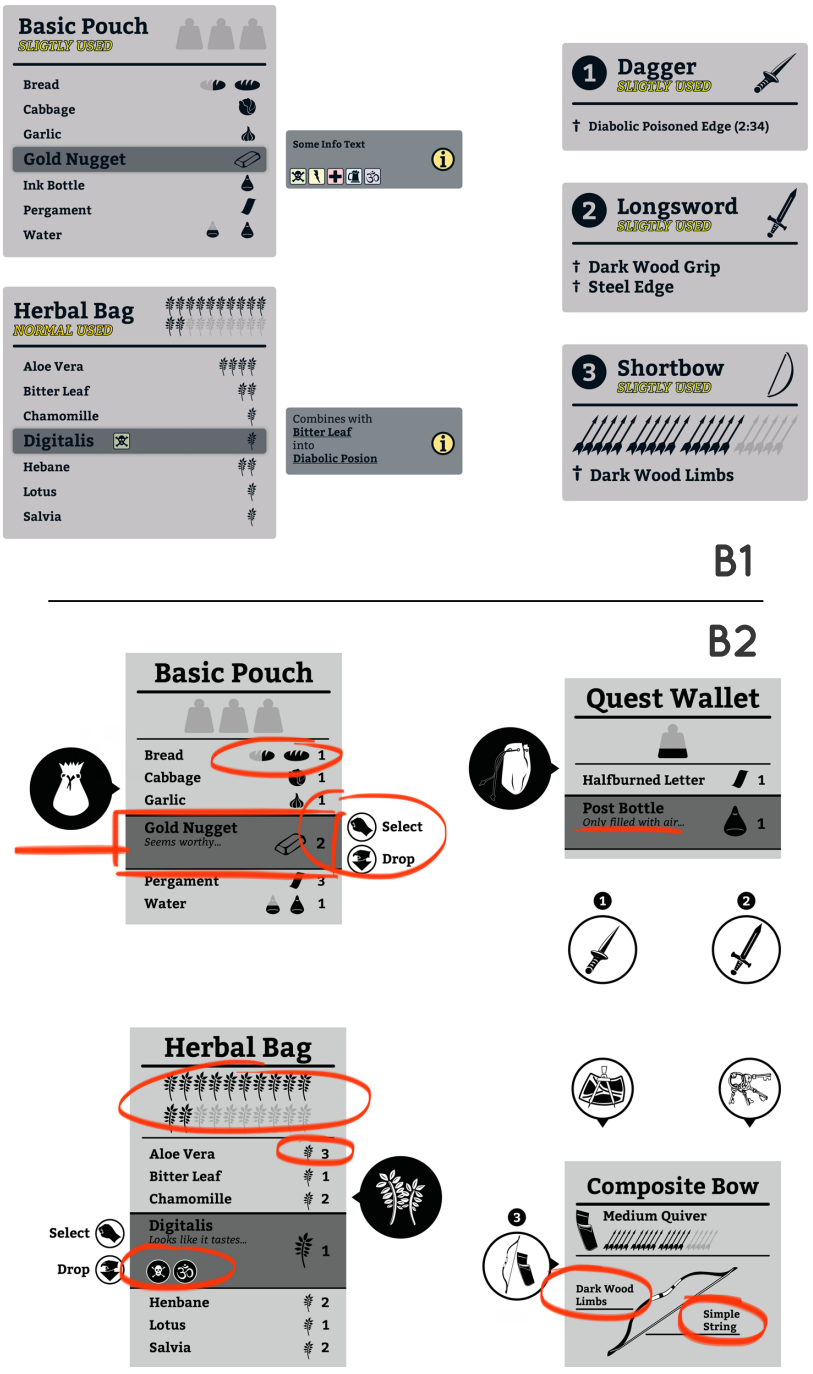


FIGURE 6.3: The refined in-game inventory UI design after session one

6.6.2 Findings & User Feedback

Both participants prefer the circle UI, the six issues found for the in-game UI are listed in table E.2. Crucial statements were, that "interaction is to slow for power users and can get annoying with frequent usage", there are "to much actions needed to achieve goals" and there is "no improvement or advantage using this design". Positive feedback regarding the in-game UI was stated, despite the fact it was rejected; again, the possible combination

with the circle UI, and a modified version, for a different use case, were suggested. The circle UI was enhanced during the session, the result is displayed in figure E.4.

Likewise, the context how the inventory is used was questioned: whether the game should pause, stays real-time, or uses a slow-motion effect. The main purpose was identified as giving fast access on belongings, thus a minimal, unobstructed concept with flat menu structure is more desirable. The circle UI was found more satisfying for this, a complete list of the 29 findings is shown in table E.1. Where the in-game UI offered less potential for discussion, the circle UI was debated comprehensively.

The hypotheses from session 1 (see section 6.5.3), concerning the circle UI were partial met; **H1.1.3** can only evaluate with a functional prototype, the other hypotheses have proven to be valid. From the hypotheses relating to the in-game UI, **H1.2.1** and **H1.2.2** were realized and evaluated with convincing results; nonetheless, **H1.2.3** revealed a insufficient interaction concept. Regarding the general suggestions raised in session 1, six out of seven were met, only the realization of **H1.3.1** was inadequate.

6.6.3 Second Refinement

Like the first refinement, the evaluation results are used to pose new design hypotheses. They are the additional requirements for the next prototype, developed with the UE4. With the functional implementation, interaction concepts can be tested with realistic conditions; the consideration of detailed interaction was beyond the scope of team prototyping or paper prototyping. Supplementary suggestions and hypotheses are described below, the enhanced design of the circle UI is shown in figure 6.4.

Immediate Circle Inventory Hypotheses

H2.1.1: Display numeric inventory fill-level at bottom right.

H2.1.2: Combine icon and number for items, use only a representative to display amount of item consumption.

H2.1.3: Inventory must offer a scroll-bar if content is too large.

H2.1.4: Display interaction options for items in an overlay.

Incremental Circle Inventory Hypotheses

H2.2.1: Add tool-tips on hovering items, attribute icons or weapon enhancements.

H2.2.2: Add intuitively customizable hot-keys for items.

H2.2.3: Add colours for feedback, warnings or errors.

H2.2.4: Add colours for certain icons or text.

H2.2.5: Make the UI contrasting to the game world, for readability, aesthetics, cognitive load, etc.

H2.2.6: Use less numbers for attributes, use a description on what advantages enhancements/components will give.

H2.2.7: Add a weapon or item preview – a 3D model or drawing.

H2.2.8: Experiment with a character/person shaped layout.

H2.2.9: Support learning processes, use tutorials, story and player character to introduce mechanics and new game elements.

H2.2.10: Experiment with simplicity vs. informations and aesthetics.

H2.2.11: Use the centre for additional feedback or options.

H2.2.12: Reduce mouse movement distances, make inventory more compact.

6.7 Functional Implementation Stage: Third Iteration

6.7.1 User Interface Implementation

After the last session, the suggestions and findings can be implemented directly inside the UE4. Some of the required assets and code were created along the process; UI elements from the inventory can be exported from *InDesing* for use inside the UE4. The circle UI is build with the engine, using UMG and the code-framework provided. The functional components for the inventory, like items, the inventory itself or events relating to inventory behaviour (e.g. pick-up or drop), were partly present before the project started. During the sessions, lacks of the current implementation were found and improved. This took about 80 hours, and covered specific game mechanics for items, inventory and introduced a new element: the item-slot. Item-slots behave comparable to inventories, but can only contain one item (e.g. a sword sheath). Inventories were adjusted to work like items with space for more items inside them and are bound to one item and not – like before – to a character.

Code-base In addition to the changes made for functionality, new classes for UI elements (further called *widgets*), were produced. The workflow of the Unreal Engine supports the creation of fundamental gameplay in C++, which is the base for blueprints. Those allow faster iteration inside the engine, where the specific behaviour is created with a visual scripting system, " which enables classes to be created in-editor through wiring together function blocks and property references " (Epic Games Inc., 2015e). Changes made on the gameplay code can easily recompile and get reflected by the Unreal Editor in real time, also referred to as "Hot Reload" (Epic Games Inc., 2015j).

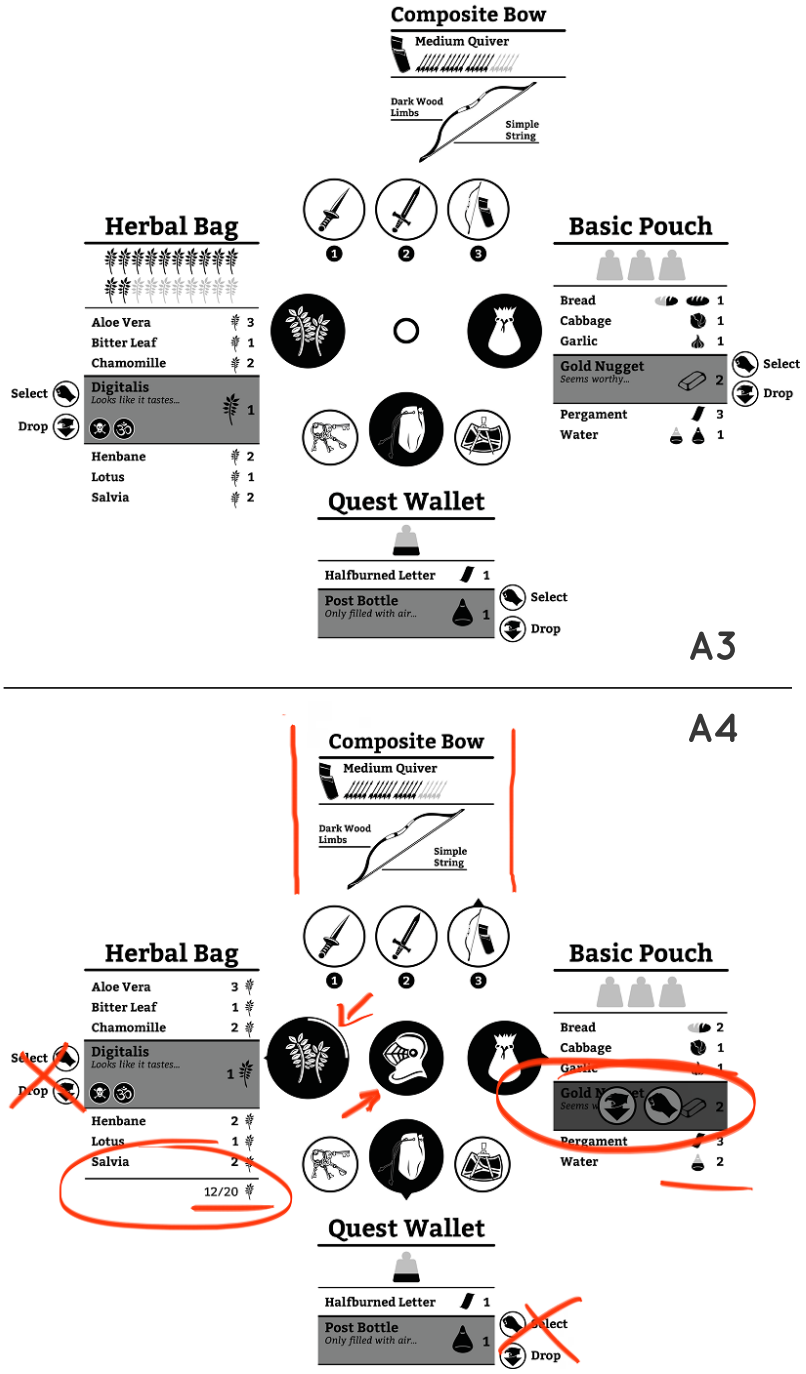


FIGURE 6.4: The refined circle inventory UI design after the second session

Slate This is the Unreal Engine's custom UI programming framework; the UI of the Unreal Engine 4 itself was created using Slate. The motivation of the framework is, to improve UI design and interaction design, to control data flow (model-view) and to enhance learning a new UI. Slate is a pure programming framework, characteristics are: easy access to model's code and data, procedural UI generation, UI description is hard to screw up, and supports animation and styling. Nonetheless, it is still a pure programming interface, therefore Unreal Motion Graphics (UMG) and a proper UI editor were introduced, to

provide a visual authoring tool for the UI creation (Epic Games Inc., 2015f).

UMG The UI designer can be used to create UI elements (widgets) and complete HUDs, menus or other interface related content (see figure 6.5). Basic widgets and layouts/container are build-in, like grid-layouts, overlays, text-fields or buttons. Widgets – either custom or pre-defined – can be reused on UI creation time and dynamically instantiated during the game's run-time. The editor for UI creation is structured the same way, like the rest of the engine is; if one has experience with the blueprint editor, only the designer view has to be learned. Property-binding of relating game-object attributes enables e.g. to connect the health of a character with the according health-bar. Within the editor, the behaviour of UI elements can be established with events, e.g. on-click, drag or hover for each component of a widget or the widget itself. Animations are created in an extra panel, and can involve multiple widget components. Styles define the look of the pre-defined (like buttons, panels, progress-bars, etc.) and the custom widgets and their components. The fonts used by the game are collected in Composite Fonts, they describe the font-family and certain sub-fonts (Epic Games Inc., 2015g).

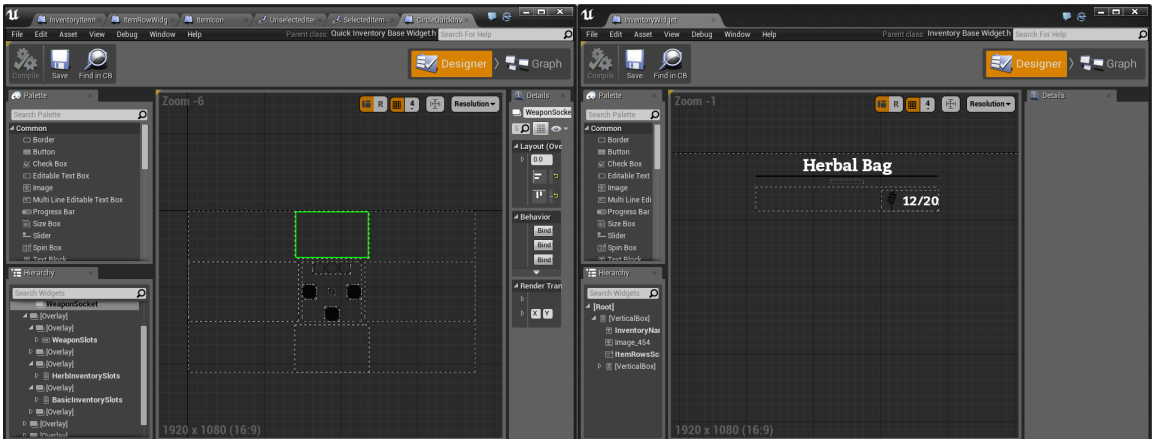


FIGURE 6.5: The UMG UI Designer view

Prototype scope The amount of suggestions and requirements, resulted from session 1 and 2, is extensive. Where the four suggestions and hypotheses, immediate changes during the prototyping, produced by session 2 (see section 6.6.3), are covered by the first functional prototype. The remaining twelve hypotheses will get processed in future iterations, due to the thesis' scope. The design hypotheses from session 1 (see section 6.5.3) are still valid; and can be considered as progressive requirements of the prototyping process. The prototyping process in the UE4 consumed about 40 hours for pure implementation. Additional 40 hours were required to test several features and to implement basic widget classes in C++ for later development inside the engine. The implementation level achieved can be seen in figure 6.6, the functionalities and actions provided are listen in table 6.3.

TABLE 6.3: The functionality covered by the prototype created with the UE4 and evaluated in session 3

ID	Category	Provided Functionality
F1	Game Mechanics & UI	open and close inventory
F2	Game Mechanics & UI	pick-up items
F3	Game Mechanics & UI	drop items (from a inventory or slot and from hands)
F4	Game Mechanics & UI	store items in a inventory (e.g. a apple in a pouch)
F5	Game Mechanics & UI	store weapons in a weapon slot (e.g. shield on the back)
F6	Game Mechanics & UI	store/equip a inventory item into a slot (e.g. a pouch on the belt)
F7	Game Mechanics & UI	move items between inventories
F8	Game Mechanics & UI	use items (e.g. fight with a weapon or drink water)
F9	Game Mechanics & UI	select items (move them out of inventory into the characters hands)
F10	UI	display slots and their content
F11	UI	display inventories and their content
F12	UI	display weapons and their description
F13	UI	sort/move inventory slots and weapon slots
F14	UI	highlight selected item
F15	UI	display interaction options for selected item
F16	UI	display item description, amount of items per type and the amount of consumption

6.7.2 Conducting the Evaluation

The prototype created with the UE4 is evaluated with selected usability heuristics; whereby all the five testers participated in session 1, additional including the players from session 2. The evaluation again takes place in the living room, using a beamer, wireless mouse and keyboard. The testers are introduced to the heuristics (see appendix F), they have to rate whether the heuristics where implemented adequately. After the short introduction, the players start in a test-level, providing all assets they might need to test the inventory UI. They are left to their own, in order to explore the UI; during the whole session, the author acts as observer and interviewer, taking notes on feedback and behaviour. The five tests lasted about 1 and a half hour, during this time the participants were free when to rate which heuristic. The full set of heuristics used is listed in the tables F.1, F.2, F.3 and F.4.

6.7.3 Findings & User Feedback

The players rated certain features more positive, than the actual implementation level. They might have considered how the features will evolve, after they are realized in more detail. Nonetheless, most of the requirements displayed by the design hypotheses from session 1 and 2 (see sections 6.5.3 and 6.6.3), were met in sufficient degree. The heuristics

FIGURE 6.6: The circle inventory UI, implemented with the UE4, for the evaluation in session 3

TABLE 6.4: Session 3 – Participant overview

Subject	1	2	3	4	6
Participant in Session 1	yes	yes	yes	yes	yes
Participant in Session 2	no	yes	yes	no	no
Age	22	25	24	27	26
Gender	male	male	male	male	male
Session Duration	1:17	1:02	1:25	1:32	1:22

relating to instructions, training, warnings, help, means for errors and recovery options, have proven to be covered least by the prototype (see table F.2). While the controls are not properly customizable yet, the display, audio and video settings cannot be changed too. The control scheme of the prototype uses industry standards and is minimized in the available options; thus the testers seemed to be satisfied with the provided controls (see table F.1). Heuristics on aesthetics and immersion were rated sufficiently, this can be seen as the result of the preceding prototyping sessions (see table F.3). Finally the heuristics

on feedback and cognitive load were only rated average, due to the fact, feedback was not implemented in detail, this was no surprise (see table F.4). Detailed statistics can be found in appendix F and table F.6. The qualitative results for the prototype evaluation are displayed in table F.5 and will be used for future improvements.

6.7.4 Future Refinements

There still exist plenty of requirements that are not met sufficiently. The list of *Incremental Circle Inventory Hypotheses* in section 6.6.3 needs to be implemented and evaluated again. The 24 suggestions from the heuristic evaluation concern similar requirements, and will find consideration in the next iterations (see table F.5. From the 29 defined hypotheses, 17 were accomplished with the first functional prototype, created in the UE4. The remaining twelve were partly prepared or implemented in low-quality; they are targeted for the next iteration.

Chapter 7

Tool Evaluation

> *"We notice things that don't work. We don't notice things that do. We notice computers, we don't notice pennies. We notice e-book readers, we don't notice books."*
>
> Douglas Adams – The Salmon of Doubt

This chapter aims to answer **RQ2**, prototyping and tools were discussed in the chapters 3 and 4. After the used criteria are described, a discussion on how those are met for the Unreal Engine 4 follows. The chapter concludes with a classification of tools used during the UI prototyping process.

7.1 Prototyping Tool Evaluation

7.1.1 Prototyping Tool Criteria

Like already mentioned in chapter 4, the criteria used by Guldbrandsen and Storstein are applied (see Guldbrandsen and Storstein, 2010, p. 79-81). The evaluation is qualitative, the judgement for each criteria is done by the author. The engines effectiveness as a prototyping tool can also be judged by the result of the functional UI prototype; and by issues, workflows, routines and dependencies during the actual usage of the UE4 – performed in the case study in chapter 6

TABLE 7.1: Criteria for a prototyping tool evaluation, borrowed from Guldbrandsen and Storstein (see Guldbrandsen and Storstein, 2010, p. 79-81)

ID	Criteria	
1	Learning curve	How is the ease of use, exist a intuitive and consistent structure and actions? Is it possible to get results right away? What requires extensive training? Is the engine accessible to users without professional experience in game development? What programming language used with the toolkit? Is it common or the syntax familiar? Identify unconventional GUI-elements in the tools. What background is provided/required by the user?
2	Development Speed	How is the compile and deploy time when altering the prototype? Does the framework support changes without recompiles? What effort is required to add or change a game element?
3	Flexibility	Are there any restrictions on the user? Can any type of game be created? Single and multiplayer game experiences?
4	Stability	How stable is the framework? Is work lost from crashes? Are there major bugs in the tools or engine? Are the developers actively maintaining the engine?
5	Community Support and Documentation	What amount of documentation is available? Is there support to and from users? Is it easy to find and understand? Are there undocumented features? Are the developers actively publishing and updating the documentation? Is there an active user community?
6	Licensing	licensing terms, price, feature restrictions, ownership, closed or open source, participation fee, platforms and additional third party licenses
7	Competitiveness	Compare subject to other engines or comparable software on the market. What feature sets are provided, is it sufficiently sophisticated to create state-of-the art game prototypes? Are there functionality not present in the development kit or commonly present in others?

7.2 Evaluation of the Unreal Engine 4

It follows a discussion on the criteria listed in table 7.1. It is to consider, that the author has previous knowledge of the Unreal Engine 4. The evaluation assumes that basic understanding regarding games (e.g. what is a RPG) and interfaces (e.g. what is a button) is provided.

7.2.1 Learning Curve

The engine provides its own GUI, the whole editor and tools use *Slate* (see section 6.7.1) to draw the interface. Thus the complete framework has the same structure, style and wording. The editor starts with a tutorial at the fist use, new tools provide additional tutorials; a reference to detailed documentation can be found next to every tutorial. Additionally the *Epic Games Launcher* provides learning resources, new versions and changes. There are multiple resources where help, information, examples, tutorials and answers can be found (Epic Games Inc., 2015h; Epic Games Inc., 2014e; Epic Games Inc., 2014d; Epic Games Inc., 2015i).

The engine starts with several build in game prototypes, for e.g. a FPS, a top-down shooter, a vehicle game or a side-scroller. It also offers some basic content for functional testing. Nonetheless, there are features that require some more experience: e.g. the blueprint editor, material editor or animation (blueprint) editor.

Due to the usage of C++ instead of Unreal Script, the acceptance is much higher and C++ is still the favourite language for excellent performance and feature-heavy games. If expertise in C++ programming, 3D modelling or animating is provided, the learning process will positively adjust.

7.2.2 Development Speed

With the options for a cooked preview (cooking is a pre-process were game assets are prepared for final usage) and the packaging onto several platforms, a prototype can be tested right away. During the cook and package process, warnings and errors can occur; they have to be fixed for further progression. Depending on the size of the project, or amount of changes, the cook and package process length can vary.

Sufficient hardware provided the engines overall performance is excellent. Small changes and even new features, not concerning source-code, can be implemented and tested after fast recompile of the affected blueprints. Recompiling source-code takes more time, using

blueprint for iterating the major functionality is recommended. Simulation and live debugging of blueprints and code, improve the iteration processes (Epic Games Inc., 2015j; Epic Games Inc., 2015e).

7.2.3 Flexibility

The overall compatibility is sufficient, nevertheless there exist some restrictions. The best experience is achieved on *Windows* or *MacOS*, *Linux* is still under development. While C++ is common amongst game programmers, there exist languages that are easier to read, learn and use (like JavaScript or C# for *Unity*). But the blueprint scripting framework offers a much more intuitive way for the creation of game-play, -mechanics and UI. The engine offers several multi- and single-player templates, for both C++ and Blueprint base games. The engine offers *Perforce*, *SVN* and a beta *Git* version control system integration. Engine associated assets can be created from various other resources (e.g. PNG graphics, FBX models or WAVE files). To some extend, content create with the engine can be exported into common file formats (Epic Games Inc., 2015j).

7.2.4 Stability

The engine is still developed and improved, currently targeting version 4.7. During the time the author used the engine, stability increased steadily; additionally the performance can be adjusted dynamically. If a crash occurs, a error description and a additional bug report form is displayed. The development of C++ game-code may lead to a unexpected crash; to prevent loss, the engine has a cycling autosave function. Altogether the performance stays excellent over time and major bugs or even crashes are rare.

7.2.5 Community Support and Documentation

As already mentioned the documentation is large, a quite active community is visible in the forums, answer-hub and engine wiki. There exists a regular twitch stream were developers discuss new features, perform tutorials or advertise the community and events (Epic Games Inc. and Twitch Interactive, Inc., 2015; Epic Games Inc., 2015k; Epic Games Inc., 2015i). The developers of the engine are present in the forums as well as the answer-hub; the documentation is improved continuously, and there exist several video-tutorials onto nearly every feature or workflow. There is also the option to sell and buy content on the *Unreal Engine Marketplace* (Epic Games Inc., 2015l).

7.2.6 Licensing

The licensing terms of the engine are quite simple, they can be found in the "UNREAL® ENGINE END USER LICENSE AGREEMENT" (Epic Games Inc., 2014f). For each product made with the engine and sold, a royalty of 5% has to be paid; if the gross revenue is below 3.000 $ per calender quarter no royalty is required. Additional third party licenses may be required for *iOS* or *Android* development and deployment. The engine itself is available for 19 $ per month, this includes updates, sample content, and access to the Git repository with the complete engine source-code. The documentation and other help and learning resources are for free.

7.2.7 Competitiveness

Competitors of the Unreal Engine obviously are other game engines like the CryEngine or Unity (Crytek GmbH, 2014; Unity Technologies, 2014a) . The features of the Unreal Engine and CryEngine are comparable in quality and amount (see section 4.3). The lighting techniques offered by the CryEngine are quite a bit better, but latest iterations of the UE4 provide comparable features, like "Lightmass Global Illumination" and "Distance Field Ambient Occlusion" (Epic Games Inc., 2015i). The price of the CryEngine is 9.90 $ per month, but offers no source-code access. Unity Pro costs 75 $ per month, and offers free updates; additional packages include iOS and Android support for 75 $ per month and support for collaborative development. The Unity Engine is easy affordable for simple games as well as more complex ones. The feature-set is adequate, but the Unity Engine is inferior in overall performance. Compared to some of the rivals, the Unreal Engine provides a excellent state-of-the-art game development tool, a complete framework for fast and easy content creation, testing and improvement; offering new useful features continuously.

7.3 Classification of UI Prototyping Tools

During the project, pen-and-paper sketches, digital paper-prototypes and functional prototypes were created and evaluated. The tools used show familiar characteristics like fidelity, functionality, resources and effort (see chapter 3 for more on fidelity and functionality). Most likely the fidelity and functionality of a prototype is low at the beginning. As the prototyping process progresses, the fidelity increases along with the functionality; tools have to comprehend this transition. Different stages need the proper tool to accomplish the design or implementation. The classification proposed is a model where fidelity and functionality are separate dimensions:

Fidelity: The quality and quantity of components, features and requirements, implemented by the prototype. Where low-fidelity prototypes are most likely pen-and-paper sketches, paper prototypes or board game prototypes; not providing persistence, only core functionalities, design ideas, visual layout and general mechanics. Medium-fidelity prototyping, like e.g. Wizard-Of-Oz or Horizontal Simulation, can either focus on specific features or generally outline the overall look-and-feel. High-fidelity prototypes usually execute immediately, while they can vary in functional completeness. One could simply overlay a conceptual UI mock-up with the game world, without providing any functionality or valid feedback; nevertheless, the execution context is already similar to the final product's.

Functionality: The functional completeness of features or requirements, regarding the completeness of (inter-)action, feedback, control options, design and mechanics. Prototypes and tools used for creation can provide functionality independent from fidelity. A digital prototype can provide no functionality at all and focus only on design, while a board game prototype can cover mechanics, interactions and feedback.

The tools used in the case study in chapter 6 can be placed in the model, according to fidelity and functionality. The model, applied to the tools used in the prototyping process: Pen-and-Paper, InDesign and the Unreal Engine 4, are pictured in figure 7.1.

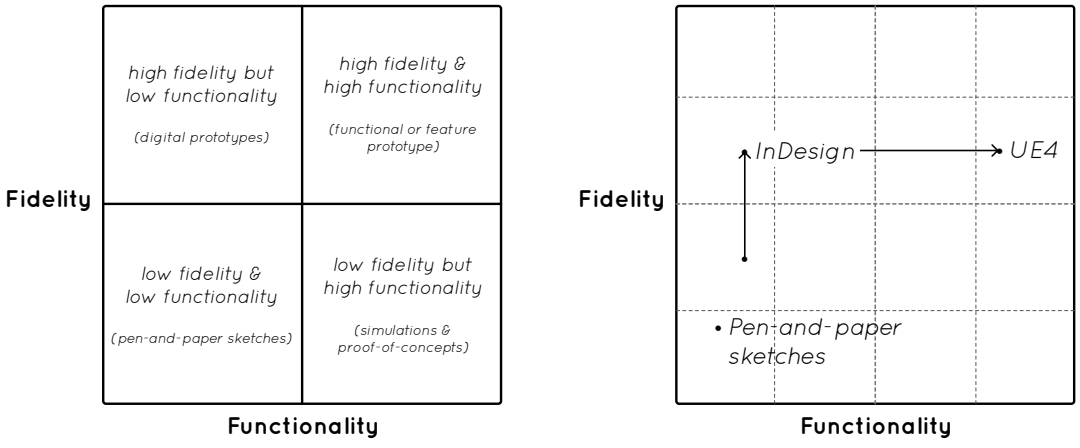


FIGURE 7.1: The model for a prototyping tool classification space, spanned by fidelity level and realized functionality

Chapter 8

Discussion

"In order to fly, all one must do is simply miss the ground."

Douglas Adams – Life, the Universe and Everything

In the preceding literature study, methods and techniques from related work on prototyping, usability evaluation and user interface design, were displayed; the selections made for techniques, tools and process have proven effective compared to requirements and outcome.

Great results fruited from user-centred design approaches, like the *Interviews* and *Surveys*, conducted during the prototyping case-study (see chapter 6). Evaluating the first usage of a design can reveal crucial aspects in interaction concepts, structure and insight, on the goals and tasks, that are desired by the user. Therefore, *Focus Group* and *Cognitive Walkthrough* were used in combination with interview and survey during a *Formative Playtest Survey*. The synergy of the techniques, was experienced during the prototype evaluation; and the amount of design hypotheses gathered throughout the whole case study, is immense. Whether this combination is adequate, still depends on the individual project and context; for agile and rapid development, with unknown and changing requirements, it seems indeed useful.

A successful method has proven to be *Team Prototyping*, with three evaluators generating more than 30 findings in two and a half hour (see section 6.6). This seems pretty effective compared to the paper prototype evaluation, which revealed about 50 findings, with six participants plus one observer, in overall 14 hours of test conduction (see section 6.5). It has to be considered, that the team prototyping session was performed after the paper prototyping; which had a additional preceding introduction, with a duration between half and one hour. The participants therefore were familiar with the subject; team prototyping

is most useful, if conducted with involved developers, expert usability testers or expert users.

The *Expert Heuristic Evaluation*, conducted in the last stage of the prototyping process, has shown a sufficient overview on the prototype's current implementation level (see section 6.7). Continuously using heuristics to measure the degree of completion, in regard of progressing design hypotheses, appears to cope with the risk of having to much features and requirements. Having the same end-users accompanying the development seems beneficial, but first usage tests should be performed together with progressive user testing.

The *Unreal Engine* has proven, to support game and UI prototyping in particular, with broad flexibility for e.g. layout, controls, feedback or interaction design. The interface created with the UE4 evolved in fidelity and functionality; and rapidly will continue to do so, till a certain level of completion is reached in the future. The scalability offered by the engine for iterating features and designs, turn up to be from great use for further implementations. Regarding scalability, the sequence of pen-and-paper, digital refinement and functional implementation, was experienced as appropriate. The concepts were transferred, without any loss or additional extensive processes, between the stages and tools (see chapter 7).

Chapter 9

Conclusion

> *"We have normality. I repeat, we have normality. Anything you still can't cope with is therefore your own problem."*
>
> Douglas Adams – The Hitchhiker's Guide to the Galaxy

Performing prototyping along with game development is already common in the industry and across independent developers (see chapter 1). Recent scientific research contributed by Guldbrandsen and Storstein (Guldbrandsen and Storstein, 2010) onto evolutionary prototyping with a game engine, shows the increasing importance for interdisciplinary exploration of game development processes. Therefore, methods and techniques from software engineering, human-computer interaction and game related research, can be combined when developing a game (Fagerholt and Lorentzon, 2009; Breda, 2008; Laitinen, 2005).

This combination has been applied in the case study, in order to document definition and execution of a game UI prototyping process. The process revealed, despite having some idle time ahead, that the combination of *paper prototyping*, *team prototyping* and *rapid evolutionary game prototyping* can be used together with adequate prototyping tools, for early game UI prototypes. Pen-and-paper sketches, digitally refined with InDesign, and the functional implementation within the UE4 work well together, and can be executed by beginners and experts alike.

User interfaces act in a key-position, as communication channel between the player and the game. They are essential for the way, the game is played; influencing experience and involvement of the player. In order to make the game "more fun", UI design guidelines, usability evaluation or game heuristics can be used to progressively enhance game and interface (Conyer, 1995; Fricker, 2012; Pinelle et al., 2008).

User-centred design can increase the design outcome immense, it helps to compare designer expectations and the users actual behaviour. In dialogue with players, UI and game issues, design ideas and further suggestions can be discussed, if resources and participants for evaluation are available (Hix et al., 1999; Pagulayan et al., 2003).

The research questions, stated in section 1.7, are solved from the authors point of view. The processes defined and applied had fruitful results, and the Unreal Engine has proved to support UI design, development and prototyping.

Chapter 10

Future Work

> *"Do you want me to sit in a corner and rust, or just fall apart where I'm standing?"*
>
> Douglas Adams – The Hitchhiker's Guide to the Galaxy

Tasks for future development of the prototype resulted by the case study have been already identified in the end of chapter 6. Finishing the current prototype seems obvious for the author, as well as prototyping other parts of the UI and game, such as a crafting interface or artificial intelligence. A re-evaluation of the process used will occur accordingly, as well as a potential process improvement.

How further iterations of the *Unreal Engine*, might improve its existing features and introduce new, would be interesting to discover in future research. Comparing the results of the case study with others, performed with different tools, like *board-game prototypes*, *Flash*, the *CryEngine* or *Unity*, using the same process and evaluation, seems promising to explore in future projects.

Researching the usage of the UE4 and other game engines "in the field": either in the games industry or independent and academic context, may be appropriate to discover and evaluate techniques used in game development.

Bibliography

[3] Michael Agustin et al. "Game sketching". In: *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*. ACM. 2007, pp. 36–43.

[4] Marcus Andrews. "Game UI Discoveries: What Players Want". In: *Gamasutra* (2010). URL: http://www.gamasutra.com/view/feature/4286/game_ui_discoveries_what_players_.php.

[5] Ahmed Arous et al. "Crazy Doc's Evolution Run". 2014. URL: http://gitlab.void-ptr.org/opensource/crazy-docs-evolution-run.

[6] Autodesk Inc. "Autodesk Scaleform". 2014. URL: http://gameware.autodesk.com/scaleform.

[7] Autodesk Inc. "Autodesk Scaleform Integrations". 2014. URL: http://gameware.autodesk.com/scaleform/integrations.

[9] Rafael Ballagas and Steffen P Walz. "REXplorer: Using player-centered iterative design techniques for pervasive game development". In: *Pervasive Gaming Applications* 2 (2007).

[10] Laura Ballay. "Prototyping as Storytelling". 2012. URL: http://www.uxdesignstrategy.com/prototyping-as-storytelling/.

[11] Balsamiq Studios, LLC. "Balsamiq". 2015. URL: http://balsamiq.com/.

[12] Victor R Basili. "The experimental paradigm in software engineering". In: *Experimental Software Engineering Issues: Critical Assessment and Future Directions*. Springer, 1993, pp. 1–12. URL: https://www.cs.umd.edu/~basili/publications/chapters/C17.pdf.

[13] Victor R Basili. "The role of experimentation in software engineering: past, current, and future". In: *Proceedings of the 18th international conference on Software engineering*. IEEE Computer Society. 1996, pp. 442–449.

[14] Billy Berghammer. "The History Of Team Fortress 2". In: *Game Informer Magazine* (2007). URL: http://web.archive.org/web/20070403160515/http://www.gameinformer.com/News/Story/200703/N07.0326.1849.05812.htm.

[16] Åsa Blomquist and Mattias Arvola. "Personas in action: ethnography in an interaction design team". In: *Proceedings of the second Nordic conference on Human-computer interaction*. ACM. 2002, pp. 197–200. URL: http://courses.washington.edu/dmedia/2005a/pub/acm-p197-blomquist.pdf.

[17] Jonathan Blow. "IGS Video: Jon Blow On Indie Prototyping, Braid". In: *Gamasutra* (2007). URL: http://www.gamasutra.com/view/news/106242/IGS_Video_Jon_Blow_On_Indie_Prototyping_Braid.php.

[19] Doug A Bowman et al. "An introduction to 3-D user interface design". In: *Presence: Teleoperators and virtual environments* 10.1 (2001), pp. 96–108. URL: http://eprints.cs.vt.edu/archive/00000548/01/3dui_presence.pdf.

[20] Luca Breda. "Invisible Walls". In: *Game Career Guide* (2008). URL: http://gamecareerguide.com/features/593/invisible_.php?page=1%3E(2009-01-16)Bidwell.

[21] CBS Interactive, Inc. "Metacritic - Gothic". 2015. URL: http://www.metacritic.com/search/all/gothic/results.

[22] CBS Interactive, Inc. "Metacritic - The Witcher". 2015. URL: http://www.metacritic.com/game/pc/the-witcher.

[23] CBS Interactive, Inc. "Metacritic - Witcher 2". 2015. URL: http://www.metacritic.com/game/pc/the-witcher-2-assassins-of-kings.

[25] CD PROJEKT, S.A. "The Witcher". 2015. URL: http://thewitcher.com/witcher1/.

[27] Georgios Christou. "Usability evaluation for video games: Formal evaluation methods". In: *Gamasutra* (2010). URL: http://www.gamasutra.com/view/news/39633/Usability_evaluation_for_video_games_Formal_evaluation_methods.php.

[28] Concept.ly.com. "concept.ly". 2015. URL: http://www.concept.ly/.

[29] Merle Conyer. "User and usability testing-how it should be undertaken?" In: *Australian journal of educational technology* 11 (1995), pp. 38–51. URL: http://www.ascilite.org.au/ajet/ajet11/conyer.html.

[30] Adrien Coyette et al. "Multi-fidelity prototyping of user interfaces". In: *Human-Computer Interaction–INTERACT 2007*. Springer, 2007, pp. 150–164. URL: https://lilab.isys.ucl.ac.be/bchi/publications/2007/Coyette-Interact2007.pdf.

[31] Jim Crawford. "FINISH YOUR JAM GAME!! And make sure you get the most out of rapid prototyping". In: *Game Carrer Guide* (2014), pp. 13–18.

[32] Crytek GmbH. "Cryengine Website". 2014. URL: http://cryengine.com/get-cryengine.

[34] Crytek GmbH. "Cryengine Features". 2015. URL: http://cryengine.com/features.

[35] Liz Danzico. "User Interview Techniques". 2010. URL: http://de.slideshare.net/edanzico/user-interview-techniques.

[37] Heather Desurvire and Charlotte Wiberg. "Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration". In: *Online Communities and Social Computing*. Springer, 2009, pp. 557–566. URL: http://userbehavioristics.com/downloads/DesigningBetterGames-09HCI-Desurvire.pdf.

[38] Heather Desurvire et al. "Using heuristics to evaluate the playability of games". In: *CHI'04 extended abstracts on Human factors in computing systems*. ACM. 2004, pp. 1509–1512. URL: http://210.240.189.214/gamedesign/resources/02_class/02_class2/00_game_paper/BIT094101/BIT094101_Late%20breaking%20result%20papers_Using%20heuristics%20to%20evaluate%20the%20playability%20of%20games_%E4%BE%AF%E6%84%B7%E5%9D%87.pdf.

[39] Heather Desurvire et al. "Evaluating fun and entertainment: Developing a conceptual framework design of evaluation methods". In: *Facing Emotions: Responsible experiential design INTERACT 2007 conference, Rio, Brasil*. 2007. URL: http://userbehavioristics.com/downloads/Desurvireetal-INTERACT08.pdf.

[40] Joseph S Dumas and Janice Redish. *A practical guide to usability testing*. Intellect Books, 1999.

[41] Steve Easterbrook et al. "Selecting empirical methods for software engineering research". In: *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311. URL: https://www.cs.cmu.edu/~Compose/ftp/shaw-fin-etaps.pdf.

[43] Daniel Engelberg and Ahmed Seffah. "A framework for rapid mid-fidelity prototyping of web sites". In: *Usability*. Springer, 2002, pp. 203–215. URL: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/DG308%20DID/MFP-Prototypying.PDF.

[44] Epic Games Inc. "UNREAL DEVELOPMENT KIT". 2014. URL: https://www.unrealengine.com/products/udk/.

[45] Epic Games Inc. "UNREAL ENGINE 4 FREE FOR STUDENTS THROUGH GITHUB". 2014. URL: https://www.unrealengine.com/blog/unreal-engine-4-free-for-students-through-github.

[46] Epic Games Inc. "UNREAL ENGINE 4 GOES FREE FOR ACADEMIC USE". 2014. URL: https://www.unrealengine.com/blog/unreal-engine-4-goes-free-for-academic-use.

[47] Epic Games Inc. “Unreal Engine 4 Website”. 2014. URL: https://www.unrealengine.com.

[48] Epic Games Inc. “Unreal Engine 4 Wiki”. 2014. URL: https://wiki.unrealengine.com/Main_Page.

[49] Epic Games Inc. “UNREAL ENGINE END USER LICENSE AGREEMENT”. 2014. URL: https://www.unrealengine.com/eula.

[51] Epic Games Inc. “Epic Games on GitHub”. 2015. URL: https://github.com/EpicGames.

[52] Epic Games Inc. “Linux Support”. 2015. URL: https://wiki.unrealengine.com/Linux_Support.

[53] Epic Games Inc. “Platform Development”. 2015. URL: https://docs.unrealengine.com/latest/INT/Platforms/index.html.

[54] Epic Games Inc. “Programming Guide”. 2015. URL: https://docs.unrealengine.com/latest/INT/Programming/index.html.

[55] Epic Games Inc. “Slate Architecture”. 2015. URL: https://docs.unrealengine.com/latest/INT/Programming/Slate/Architecture/index.html.

[56] Epic Games Inc. “UMG UI Designer”. 2015. URL: https://docs.unrealengine.com/latest/INT/Engine/UMG/.

[57] Epic Games Inc. “Unreal Engine 4 Answer Hub”. 2015. URL: https://answers.unrealengine.com/index.html.

[58] Epic Games Inc. “Unreal Engine 4 Documentation”. 2015. URL: https://docs.unrealengine.com/latest/INT/.

[59] Epic Games Inc. “Unreal Engine 4 FEATURES”. 2015. URL: https://www.unrealengine.com/products/unreal-engine-4.

[60] Epic Games Inc. “Unreal Engine 4 Forums”. 2015. URL: https://forums.unrealengine.com/.

[61] Epic Games Inc. “Unreal Engine 4 Marketplace”. 2015. URL: https://www.unrealengine.com/marketplace.

[62] Epic Games Inc. and Twitch Interactive, Inc. “Unreal Engine Twitch Stream”. 2015. URL: http://www.twitch.tv/unrealengine.

[63] Evolus. “Pencil Project”. 2008–2012. URL: http://pencil.evolus.vn/.

[64] Erik Fagerholt and Magnus Lorentzon. “Beyond the hud-user interfaces for increased player immersion in fps games”. In: (2009). URL: http://publications.lib.chalmers.se/records/fulltext/111921.pdf.

[65] Melissa A Federoff. "Heuristics and usability guidelines for the creation and evaluation of fun in video games". PhD thesis. Citeseer, 2002. URL: http://ocw.metu.edu.tr/pluginfile.php/2510/mod_resource/content/0/ceit706_2/10/MelissaFederoff_Heuiristics.pdf.

[66] Forbes.com, LLC^{TM}. "Indie Game Developers Rise Up". 2008. URL: http://www.forbes.com/2008/11/20/games-indie-developers-tech-ebiz-cx_mji_1120indiegames.html.

[67] Theodore Frick et al. "Software developers' attitudes toward user-centered design". In: *24th National Convention of the Association for Educational Communications and Technology*. 2001. URL: https://www.indiana.edu/~tedfrick/aect2001/theodorefrick_softwaredevelopersattitudes.doc.

[68] Helen Fricker. "Game User-Interface Guidelines: Creating a set of Usability Design Guidelines for the FPS Game User-Interface". MA thesis. University of Huddersfield, 2012. URL: http://eprints.hud.ac.uk/17809/1/Helen_Fricker_-_Final_Thesis_-_Jan_2013.pdf.

[69] Markus Friedl. *Online game interactivity theory with cdrom*. Charles River Media, Inc., 2002.

[70] Oy Frozenbyte. "A Day in the Life of Frozenbyte QA". 2014. URL: http://www.frozenbyte.com/2014/09/a-day-in-the-life-of-frozenbyte-qa/.

[71] Tracy Fullerton. *Game design workshop: a playcentric approach to creating innovative games*. CRC Press, 2014.

[72] MIT Game Lab. "Game Development". 2015. URL: http://gamelab.mit.edu/about/game-development/.

[75] Git Project Team. "Git Website". 2015. URL: http://git-scm.com/.

[77] Google Inc. "Google Drive Website". 2014. URL: https://drive.google.com.

[79] Jacob Gube. "What Is User Experience Design? Overview, Tools And Resources". In: (2010). URL: http://www.smashingmagazine.com/2010/10/05/what-is-user-experience-design-overview-tools-and-resources/.

[80] Kjetil Guldbrandsen and Kjell Ivar Bekkerhus Storstein. "Evolutionary Game Prototyping using the Unreal Development Kit". In: (2010). URL: http://www.diva-portal.org/smash/get/diva2:356726/FULLTEXT01.pdf.

[81] H Rex Hartson and Eric C Smith. "Rapid prototyping in human-computer interface development". In: *Interacting with computers* 3.1 (1991), pp. 51–91. URL: http://eprints.cs.vt.edu/archive/00000179/01/TR-89-42.pdf.

[82] Deborah Hix et al. "User-centered design and evaluation of a real-time battlefield visualization virtual environment". In: *Virtual Reality, 1999. Proceedings., IEEE*. IEEE. 1999, pp. 96–103. URL: http://www.sv.vt.edu/future/cave/pub/hixvr99/vr99.pdf.

[84] Huerta Tipografica. "Bitter Font". 2015. URL: http://www.huertatipografica.com/fonts/bitter-ht.

[85] Inkscape Team. "Inkscape Addons". 2008–2012. URL: https://inkscape.org/en/download/addons/.

[86] Inkscape Team. "Inkscape Project". 2008–2012. URL: https://inkscape.org/en/.

[87] Daniel Johnson and Janet Wiles. "Effective affective user interface design in games". In: *Ergonomics* 46.13-14 (2003), pp. 1332–1345. URL: http://eprints.qut.edu.au/6693/1/6693.pdf.

[88] Sauli Laitinen. "Better Games Through Usability Evaluation and Testing". In: *Gamasutra* (2005). URL: http://www.gamasutra.com/view/feature/130745/better_games_through_usability_.php.

[89] James A Landay and Brad A Myers. "Sketching interfaces: Toward more human interface design". In: *Computer* 34.3 (2001), pp. 56–64.

[90] Sung Heum Lee. "Usability testing for developing effective interactive multimedia software: concepts, dimensions, and procedures". In: *Educational Technology & Society* 2.2 (1999), pp. 1–13. URL: http://www.ifets.info/journals/2_2/sung_heum_lee.html.

[91] Clayton Lewis and Cathleen Wharton. "Cognitive walkthroughs". In: *Handbook of human-computer interaction* 2 (1997), pp. 717–732. URL: http://userportal.iha.dk/~sw/kurser/brga/Litteratur/brga-6.pdf.

[92] Juan Linietsky and Ariel Manzur. "Godot Engine". 2014. URL: http://www.godotengine.org/wp/.

[93] Stein C Llanos and Kristine Jørgensen. "Do players prefer integrated user interfaces? A qualitative study of game UI design issues". In: *Proceedings of DiGRA 2011 Conference: Think Design Play*. 2011. URL: http://www.digra.org/wp-content/uploads/digital-library/11313.34398.pdf.

[94] Lucid Software Inc. "Lucidchart". 2015. URL: https://www.lucidchart.com/.

[95] Thomas W Malone. "Heuristics for designing enjoyable user interfaces: Lessons from computer games". In: *Proceedings of the 1982 conference on Human factors in computing systems*. ACM. 1982, pp. 63–68.

[96] Julian Meltzoff. *Critical thinking about research: Psychology and related fields*. American psychological association, 1998.

[97] Microsoft Corporation. "Skype Website". 2014. URL: http://www.skype.com/.

[99] Microsoft Corporation. "XNA Game Studio 4.0". 2015. URL: http://msdn.microsoft.com/de-de/library/bb200104(v=xnagamestudio.40).aspx.

[100] Brad Myers et al. "Past, present, and future of user interface software tools". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 7.1 (2000), pp. 3–28. URL: http://luci.ics.uci.edu/websiteContent/weAreLuci/biographies/faculty/djp3/LocalCopy/MyersHudsonPausch-UserInterfaceTools-TOCHI.pdf.

[101] Jakob Nielsen. "Heuristic evaluation". In: *Usability inspection methods* 17.1 (1994), pp. 25–62.

[102] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[103] Jakob Nielsen. "Usability inspection methods". In: *Conference companion on Human factors in computing systems*. ACM. 1994, pp. 413–414. URL: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/0H420/Nielsen[1994].pdf.

[105] Elina MI Ollila et al. "Using prototypes in early pervasive game development". In: *Computers in Entertainment (CIE)* 6.2 (2008), p. 17. URL: http://salvazquez.googlecode.com/svn/trunk/Doc/ComputersInEntertainment/Using%20Prototypes%20in%20Early%20Pervasive%20Game.pdf.

[106] Randy J Pagulayan et al. "User-centered design in games". In: *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (2003), pp. 883–906. URL: http://www.studiosuserresearch.com/hci_handbook_chapter.pdf.

[107] Stefanos Papalamprou. "DIEGESIS AND INTERFACE DESIGN". 2014. URL: https://gurusability.wordpress.com/2014/03/05/diegesis-and-interface-design/.

[108] Fabio Paternò and Marco Volpe. "Natural modelling of interactive applications". In: *Interactive Systems. Design, Specification, and Verification*. Springer, 2006, pp. 67–77. URL: http://giove.isti.cnr.it/attachments/publications/2006-A1-08.pdf.

[110] David Pinelle et al. "Heuristic evaluation for games: usability principles for video game design". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2008, pp. 1453–1462. URL: http://www.irit.fr/recherches/ICS/projects/twintide/upload/447.pdf.

[111] Beryl Plimmer and Mark Apperley. "INTERACTING with sketched interface designs: an evaluation study". In: *CHI'04 extended abstracts on Human factors in computing systems*. ACM. 2004, pp. 1337–1340.

[112] Luke Plunkett. "I Never Saw The Moment Video Games Hit The Mainstream. It Just Kind of Happened." 2012. URL: http://kotaku.com/5959932/i-never-saw-the-moment-video-games-hit-the-mainstream-it-just-kind-of-happened.

[114] Pluto 13, GmbH. "Gothic 2 Website". 2015. URL: http://www.pluto13.de/index.php?navtarget=3&lang=de.

[115] Pluto 13, GmbH. "Piranha Bytes Website". 2015. URL: http://www.pluto13.de/index.php?navtarget=6&lang=en.

[116] NY Rome. "Software Prototyping and Requirements Engineering". In: (1992). URL: https://sw.thecsiac.com/get_pdf/CSIAC-347214-000.pdf.

[117] Dave Russell. "Video game user interface design: Diegesis theory". 2011. URL: http://devmag.org.za/2011/02/02/video-game-user-interface-design-diegesis-theory/.

[118] Kevin Saunders and Jeannie Novak. *Game development essentials: Game interface design*. Cengage Learning, 2012.

[119] Ian Schreiber. *Challenges for game designers*. Cengage Learning, 2009.

[120] Carolyn B. Seaman. "Qualitative methods in empirical studies of software engineering". In: *Software Engineering, IEEE Transactions on* 25.4 (1999), pp. 557–572.

[121] Andrew Sears. "Heuristic walkthroughs: Finding the problems without the noise". In: *International Journal of Human-Computer Interaction* 9.3 (1997), pp. 213–234.

[122] Bruce Shelley. "Guidelines for Developing Successful Games". In: *Gamasutra* (2001). URL: http://www.gamasutra.com/view/feature/3041/guidelines_for_developing_.php.

[123] Smithsonian American Art Museum. "The Art of Video Games". 2012. URL: http://americanart.si.edu/exhibitions/archive/2012/games/.

[124] Carolyn Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Newnes, 2003.

[125] Rick Spencer. "The streamlined cognitive walkthrough method, working around social constraints encountered in a software development company". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2000, pp. 353–359. URL: http://wiki.fluidproject.org/download/attachments/1704853/p353-spencer.pdf.

[126] Dan Stapleton. "Duke Nukem Forever release date disparity demystified". In: *PC Gamer* (2011). URL: http://www.pcgamer.com/duke-nukem-forever-release-date-disparity-demystified/.

[127] Anthony Stonehouse. "User interface design in video games". In: *Gamasurtra* (2014). URL: http://gamasutra.com/blogs/AnthonyStonehouse/20140227/211823/User_interface_design_in_video_games.php.

[128] Jari Takatalo et al. "Presence, involvement, and flow in digital games". In: *Evaluating user experience in games*. Springer, 2010, pp. 23–46. URL: http://www.researchgate.net/publication/226075575_Digital_Games_the_Aftermath_Qualitative_Insights_into_Postgame_Experiences/file/3deec52c81fba41ecf.pdf#page=53.

[129] The GIMP Team. "Gimp Plugin Registry". 2001–2014. URL: http://registry.gimp.org/.

[131] TortoiseGit Project Team. "TortoiseGit Website". 2015. URL: https://code.google.com/p/tortoisegit/.

[133] Unity Technologies. "Unity 4.5 Free". 2014. URL: http://unity3d.com/unity/download.

[134] Unity Technologies. "Unity For Education". 2014. URL: https://store.unity3d.com/education.

[135] Unity Technologies. "Unity Pricing". 2014. URL: https://store.unity3d.com/products/pricing.

[137] Association User Experience Professionals. "What is User-Centered Design?" 2014. URL: http://domus.imag.fr/sites/default/files/what_is_user.pdf.

[140] Valve Corporation. "Dota 2 Blog". 2015. URL: http://blog.dota2.com/.

[141] Patrick Walker. "Early Access popularity growing, but only 25a full game". In: (2014). URL: http://www.gamesindustry.biz/articles/2014-11-13-early-access-popularity-growing-but-only-25-percent-have-released-as-a-full-game.

[142] Alf Inge Wang and Bian Wu. "An application of a game development framework in higher education". In: *International Journal of Computer Games Technology* 2009 (2009), p. 6. URL: http://www.hindawi.com/journals/ijcgt/2009/693267/.

[143] Mark Weiser. "Scale models and rapid prototyping". In: *ACM SIGSOFT Software Engineering Notes*. Vol. 7. 5. ACM. 1982, pp. 181–185.

Company and Organization Register

[1] 4A Games. 2015. URL: http://www.4a-games.com/.

[2] Adobe Systems Incorporated. 2015. URL: http://www.adobe.com/.

[8] Autodesk Inc. 2015. URL: http://www.autodesk.com/.

[15] Bethesda Softworks, LLC. 2015. URL: http://bgs.bethsoft.com/.

[18] Blue Byte GmbH. 2014. URL: http://bluebyte.de.ubi.com/en/inside-bb/studio-mainz/.

[24] CD PROJEKT, S.A. 2015. URL: http://en.cdprojektred.com/.

[26] CEGUI. 2015. URL: http://cegui.org.uk/.

[33] Crytek GmbH. 2015. URL: http://www.crytek.com/.

[36] Deep Silver Inc. 2015. URL: http://www.deepsilver.com/us/contact/.

[42] Electronic Arts Inc. 2015. URL: http://www.ea.com/1/legal-notices.

[50] Epic Games Inc. 2015. URL: http://epicgames.com/.

[73] GIMP Team. *Gimp*. 2001–2015. URL: http://www.gimp.org/.

[74] Git Project Team. 2015. URL: http://git-scm.com/.

[76] GitHub Inc. 2014. URL: https://github.com/.

[78] Google Inc. 2015. URL: http://google.com/.

[83] Host Europe GmbH. 2014. URL: https://www.hosteurope.de/.

[98] Microsoft Corporation. 2015. URL: http://www.microsoft.com/.

[104] OGRE. 2015. URL: http://www.ogre3d.org/.

[109] Perforce Software Inc. 2015. URL: http://www.perforce.com/.

[113] Pluto 13, GmbH. 2015. URL: http://www.pluto13.de/index.php?navtarget=6&lang=en.

[130] Thrown Together Studios. 2015. URL: http://www.throwntogether.jshubner.de/.

[132] Trello Inc. 2015. URL: https://trello.com/.

[136] Unity Technologies. 2015. URL: http://unity3d.com/.

[138] User Experience Professionals Association. 2014. URL: https://uxpa.org/.

[139] Valve Corporation. 2015. URL: http://www.valvesoftware.com/.

Appendix A

Caede Game Project Details

This is a list of all team members, that are currently (February 2015) part of the developer team. The project website gives more details on the updated process, members and releases (Thrown Together Studios, 2015).

- Johannes Hubner (Germany)
- Lewis Bean (UK)
- Philipp Nasal (Austria)
- Ruben Buchholz (Germany)
- Sophie Knowles (UK)
- Ulrich Tümmler (Germany)
- Hans Ferchland (Germany)

Repository Link: https://github.com/gormed/bachelor-thesis-project

Project Website Link: http://www.throwntogether.jshubner.de/

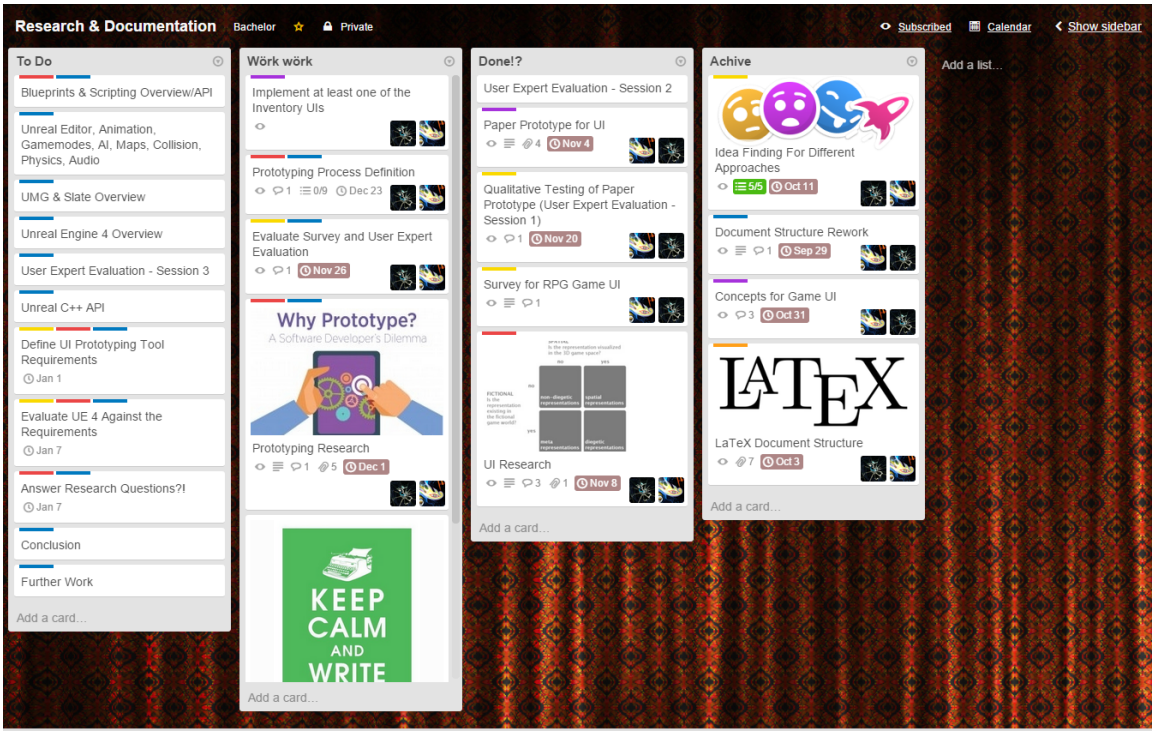


FIGURE A.1: Project management using Trello (Trello Inc., 2015)

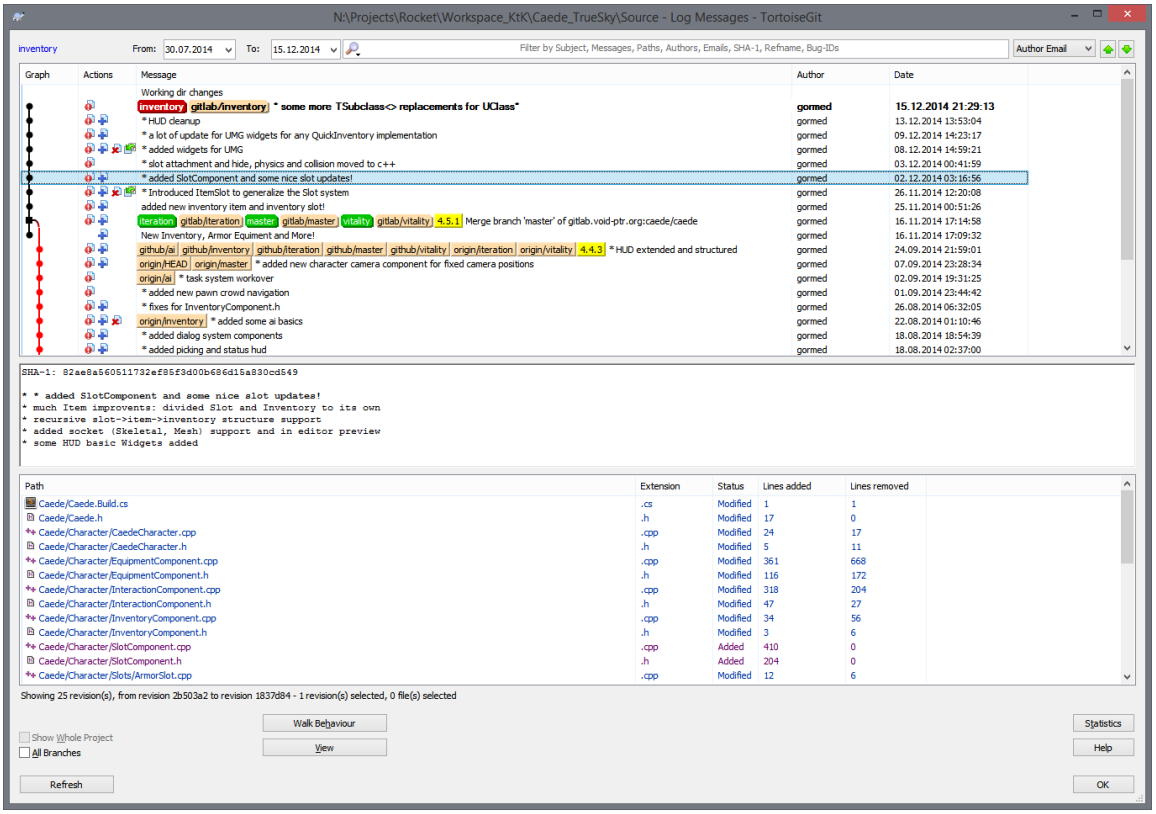


FIGURE A.2: Version control using Git (Git Project Team, 2015b) and TortoiseGit (TortoiseGit Project Team, 2015)

Appendix B

Caede Game Design Document Prototype

Looking at it superficially the game should look serious, but if you take a close look the game will be full of black humour and jokes. You might even learn a thing or two playing it. To summarize it right from the start: the target will be a realistic role playing-game set in the medieval age. The core features are a complex fighting system, realistic NPC behaviour, beautiful environments, non-linear quests and dialogues, survival aspects, much humour, a consistent and immersive world and story.

B.1 Story

B.1.1 Prologue

Today there exist 3 kingdoms around the big mountain Contumacia, the northern realm: Collis, the western realm: Pratum, and the realm at the sea: Litus, but it wasn't always this way. Some time ago the reals where unified, if you remember that time, you don't think of a glorious time, no, it was the time where the people where suppressed by the mad king and a tie full of war. The maddens hold on till the 2 Realms Collis and Pratum split from Litus and declared war against the Mad King, in this war many thing in Litus where destroyed, even the main capital, till they finally cached the king and cut of his head. Litus got a new king, and from this day on the 3 realm should exist side by side, while Pratum and Collis recovered pretty well, was Litus to much damaged as it would ever recover on its own, the burned down capital, Bandits and Raiders, the absence from merchant ships, and the only ships that arrive will only deal with the northern men, the people in Litus just work for their own surviving.

B.2 Characters

There is a general character definition that nails everything what a character can have. The different components of a character can be restricted at need and adjust independently from different individuals (including humans as well as animals).

This section will only display an overview over the characters components and general interaction/communication in between them, the B.3 section will cover the specifics.

B.2.1 General Character

The general character consists of a simple interface to its components and various meta informations and placeholder for specialized character behaviour (e.g. let a bird fly and a human walk/run). It follows a in depth description of a characters components.

Vitality

This covers all attributes like the current and maximum live or stamina of a character. It also defines the interaction between the attributes of the vitality-component itself (e.g. loose health if poisoned) and to the parent character (e.g. slower movement if on low stamina, different animations). With the ability to have live also comes the ability to die or to get exhausted.

Fighting

The component is responsible for fighting, managing the dependencies between vital and fighting components of other characters in a fight. This gives the ability to do damage to any kind of other character in the game using different weapons (or none at all). In the best case all (human) characters share the same fighting animations, their skills will differ in accuracy, speed and damage-amount.

Inventory

All characters have an inventory to store items of all kind. The inventory consists of different types to store the different item-kinds like weapons, herbs, normal items and quest-items. Those are sub-inventories with each having its own limit in weight or amount. Inventories are bound to items such as a bag, a backpack or weapon slots and are also considered as equipment.

Equipment

This component covers all types of items that can be equipped by characters. It contains the mentioned bags and pouches as well as armour, map (& compass, sun-clock), quest notes and recipe-books.

Alignment

This component controls the relation of one character with the world and it's fractions and their characters.

B.2.2 Player Character

The player character has all general character properties – but it is the most specialized character in the game, just because it is controlled by the player. There are several events linked to the actions and effects the player can make through the character in game (e.g. dying restarts/ends the game). That obviously involves the ability to control the character with an input-device.

B.2.3 Non Player Characters (NPCs)

Characters should have a day-night cycle and react to the players actions. They offer different personalities with different problems, work, secrets, intrigues and of course quests and loot.

B.3 Gameplay

Here lays the core game definition with all details and according examples. It defines the set of rules and possibilities underlying the choices the player can make. In the last subsection this also covers the difficulty and ways to scale them to find the optimum between the challenge/realism and the flow/fun/immersion.

The core features include the most general set known in the role-playing game genre. But they are often largely extended or more realistic.

B.3.1 Fighting

The strength of your strikes affects the amount of stamina depending on the time how long you hold the weapon in "rest attack" position and release it.

The character goes into the fighting state after he has done his first attack. When he holds shift + moves forward, he will go back to the normal movement state.

Hold Shift while attack will play a movement attack -¿ so he won't go into the fighting state. When he does a "long attack" (hold the button for a longer time) he will interrupt the movement + play a jump attack.

Melee (short/long weapon)

Strikes: The fighting system is based on the idea, to make a combats more realistic to real fights. So you can swing your weapon in five ways:

Dachhau (from the top)

Zwerch (to the diaphragm)

Ochs (diagonal from the top)

Eber (diagonal from below)

Stitch

If you hold your mouse button your character will rest in the corresponding strike position. Its possible to interrupt this by switching into another strike or block.

Movement: Which strikes are available depends on which position/idle the player stands. To the battle idles also fits fight steps and jumps/dodges, that begins with the relevant foot. It is possible, to move in each possible direction during fights. The player have to control, which direction he wants to go.

Blocking: The player have to face the enemy to make a save block. Also he shall look in the direction, from where the strike comes. Depending on the mouse position on the Z-axis the player plays different block animations.

Spear

Same as above, just with one difference, that the character throws the spear when the player presses the attack button for a long time.

Bow

Strikes: The character will shoot one arrow when the player presses the left mouse button. If you hold (wheel up/down -¿ twice) your mouse button your character

will rest in the corresponding strike position. Its possible to interrupt this by pressing another mouse button + "blocking key".

Blocking: When you press the "blocking key" the character will move his upper body back.

B.3.2 Vital

The health can go over 100% if you act healthy. For example, after a good dinner and a sleep and the daily morning routine, you can have a boost of 20%. The vitality of the player is not only influenced by a physical damage. Your vitality is also affected by starvation/well-fed, thirst/not thirsty, sleeplessness/rested, poisoning/doped and bleeding. You have to care of your needs and wounds, otherwise you will lose vitality.

Adrenaline is a component that will appear if you are in danger like a fight. It turn off or reduce effects like starvation, thirst, sleeplessness and poisoning.

Heal & Regenerate

As said before, you can have more than 100% health if you treat your tamagotschi right.

Wounds will reduce your health, depending which kind of wound the player need to treat them. lets say the player gets cut, so he need to put bandages on, or otherwise he will bleed out.

The player can regenerate a certain amount of health with herbs and treating wounds, but for some sort of wounds/sickness, he needs time.

Visual effects for low health:

<50% from time to time blurry, slightly red borders, colours begin to grey out

<20% turn into grey (except blood)

<5% turn into black, just a little tube in the middle shows the world

In-game effects for health:

<75% character get slow down

<30% low heath affects the stamina, it will exhaust faster

<20% another animation set

Visual effects for low stamina:

<30% from time to time blurry, another animation set

In-game effects for stamina:

<0% attacks your health

Eat & Drink

The player need to take care of his character, eating and drinking is important to hold your players vitality on the highest possible level.

The player will get thirsty and hungry over time, so he loses energy, but if he eats too much, he will get slow and tired.

Eating The player need to find the balance between salad fingers/healthy food and pork-ass/unhealthy food, for example:

flesh gives the player energy, but will poison him over time, on the other hand, salad gives just a little bit energy but will give you healthy points.

Speaking of that, if the player eats a bit of everything, nothing will happen, he is healthy and has a lot of energy, but if he eats only one type of food he will have a loot energy but sick, or healthy and not much energy. (healthiness of food is regulated by the (programmer) poison, health and fill value of the item)

Some food/drinks may raise your thirst, like bread or beer, so the player needs to find the balance there too.

Drinking The player needs to drink, or otherwise he will get slow over time, and will die after some days without water. Physical activities like pressing a brownie out of you anus, stabing fat guards or run like forest, will dehydrate you.

Poop & Urinate

If your filled up, you have to poop and piss after time. Otherwise you will slow down and after a long time you will poop/piss in your pants. NPCs will react to you, if you have trousers full of poo. For pooping you have to search for a calm place.

Pooping in the game is regulated over a quick-time event. You have to press the [E] key continuously, if you see, that the player is pressing.

B.3.3 Items

Items in general:

Weight, Value

Empty/ Full Weight

Name, Description

Identification Colour (For category)

Used Hands (One, Two, No)

Can the item: Equip?, Drop?, Pick-up?, Use?, Store?

Current Owner, Original Owner

Icon (HUD)

Armour Item:

Cannot Pick-up

Can only equip to a slot (on a character)

Can only be stored in chests/horse/etc.

Quest Item (Player Specific Items):

Important for Quests only

Map, Compass, ...

Key-chain

Decision Maker

Basic Item:

Food Items (affects Health, Stamina, Starvation, Poo, Poison values)

Bread (poison 0, prevent starvation 10, ...)

Meat (poison 10, prevent starvation 20, ...)

Salad (poison -10, prevent starvation 5, ...)

Drink Items (affects Health, Stamina, Thirst, Urinal, Poison, Alcohol, Drug values)

Water (poison -5, Thirst 20, ...)

Beer (poison 15, Thirst -5, ...)

Refill Bottles?

Healing Items (affects Health, Stamina?):

Bandages

Ointment

for healing, strength dependent on the herbs you collected

Inventory Item:

Stores specific item types: Basic Items, Herb Items, Quest Items, or all general items (like a Chest)

Herb Item:

health, stamina, poison, drug effects on use

potentially can effects all: Health, Stamina, Starvation, Thirst, Poo, Urinal, Alcohol, Poison, Drug

Appendix C

RPG User Interface Study

Role playing-game user interface study

Gothic, Risen, The Witcher & The Elder Scrolls series

Hans Ferchland, January 2014

Piranha Bytes

Gothic and Risen series

Gothic 2: The Night of the Raven

2003, Piranha Bytes, Jowood

Inventory

Gothic 3

Enhanced Edition CP 1.7

2006, Piranha Bytes, Jowood

HUD

Gothic 3 2006, Piranha Bytes, Jowood

Vanilla Game Version

Inventory

Risen 2009, Piranha Bytes, Deep Silver

World Map & HUD

Risen 2: Dark Waters

2012, Piranha Bytes, Deep Silver

HUD

Risen 3: Titan Lords

2014, Piranha Bytes, Deep Silver

Fighting HUD

Risen 3: Titan Lords

2014, Piranha Bytes, Deep Silver

Inventory

CD Project RED

The Witcher series

The Witcher

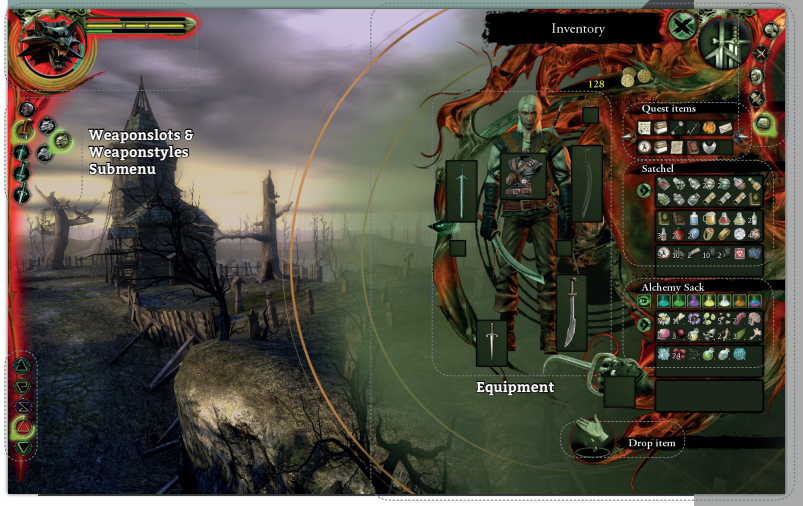
2007, CD Project RED, CD Project & Atari SA

HUD

The Witcher

2007, CD Project RED, CD Project & Atari SA

Inventory



The Witcher
2007, CD Project RED, CD Project & Atari SA

Character Improvement

The Witcher
2007, CD Project RED, CD Project & Atari SA

Alchemy/ Crafting

The Witcher 2: Assassins of Kings
2011, CD Project RED, CD Project, Atari, Namco Bandai & 1C

HUD

The Witcher 2: Assassins of Kings
2011, CD Project RED, CD Project, Atari, Namco Bandai & 1C

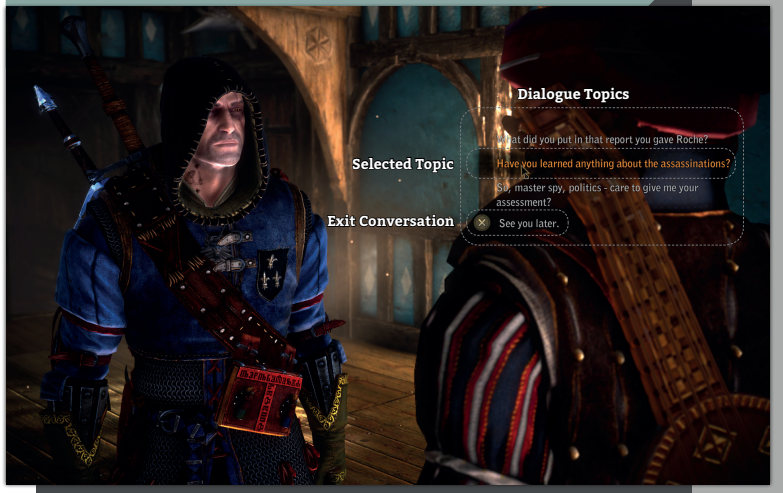
HUD + Quick Actions

The Witcher 2: Assassins of Kings
2011, CD Project RED, CD Project, Atari, Namco Bandai & 1C

Character Action & Improvement Hub

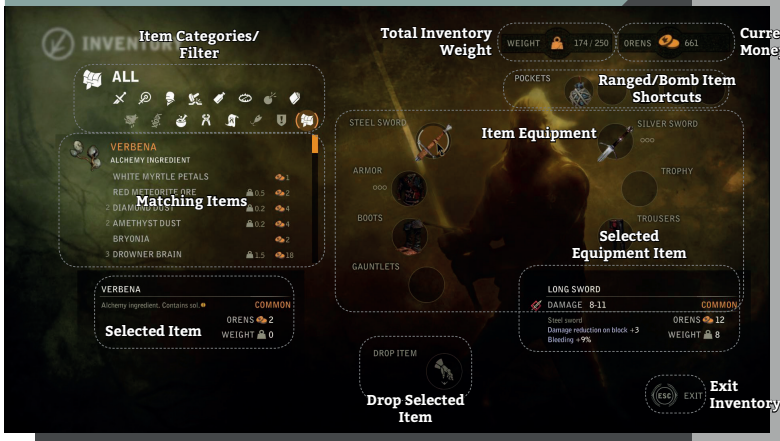
The Witcher 2: Assassins of Kings
2011, CD Project RED, CD Project, Atari, Namco Bandai & 1C

Dialogue HUD



The Witcher 2: Assassins of Kings
2011, CD Project RED, CD Project, Atari, Namco Bandai & 1C

Inventory



The Witcher 3: Wild Hunt
2015, CD Project RED, not published

HUD

Bethesda Game Studios

The Elder Scrolls series

The Elder Scrolls 3: Morrowind

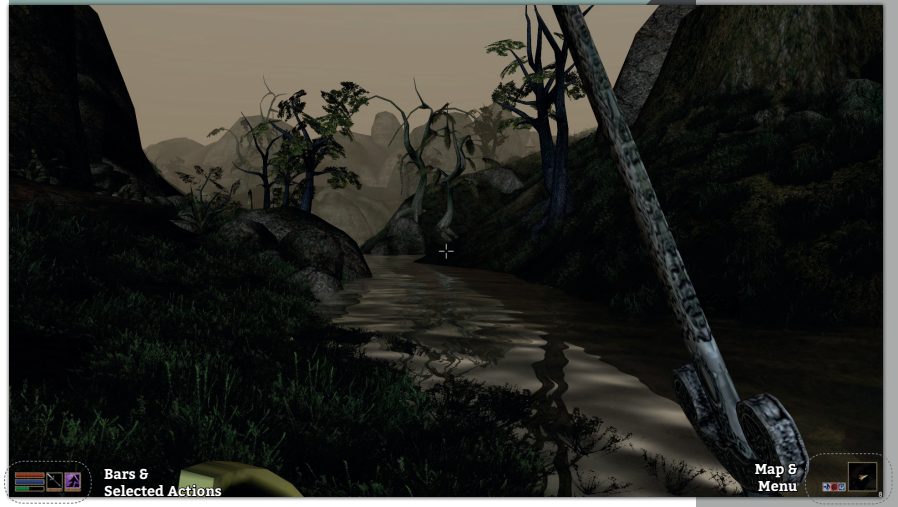
2002, Bethesda Softworks, Ubisoft

Character Overview & World Map

The Elder Scrolls 3: Morrowind

2002, Bethesda Softworks, Ubisoft

HUD



The Elder Scrolls 3: Morrowind

2002, Bethesda Softworks, Ubisoft

Item Interaction

The Elder Scrolls 4: Oblivion

2006, Bethesda Softworks,
Ubisoft, 2K Games & Bethesta Softworks

(Fight) HUD

The Elder Scrolls 4: Oblivion

2006, Bethesda Softworks,
Ubisoft, 2K Games & Bethesta Softworks

World Map

The Elder Scrolls 5: Skyrim

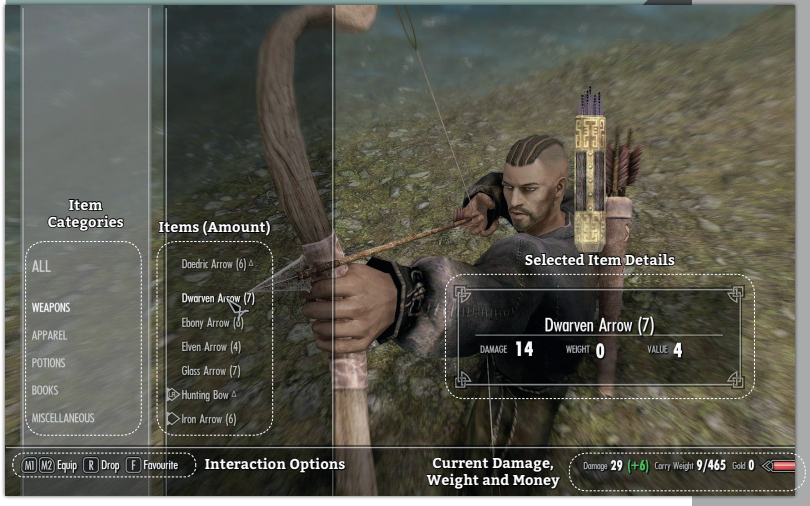
2011, Bethesda Game Studios, Bethesta Softworks

HUD

The Elder Scrolls 5: Skyrim

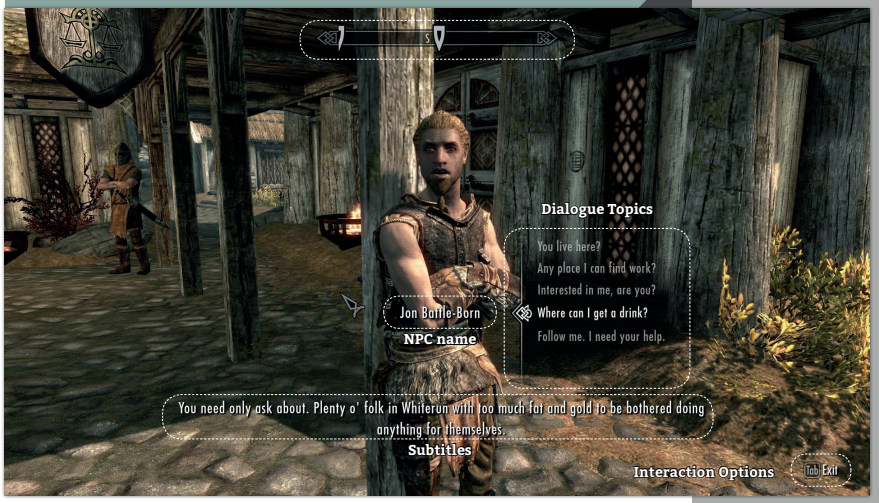
2011, Bethesda Game Studios, Bethesta Softworks

Inventory



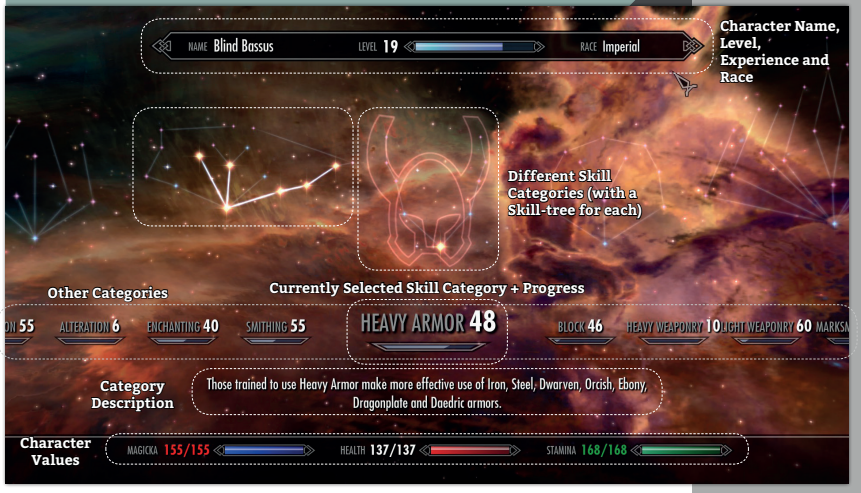
The Elder Scrolls 5: Skyrim
2011, Bethesda Game Studios, Bethesta Softworks

Dialogue HUD



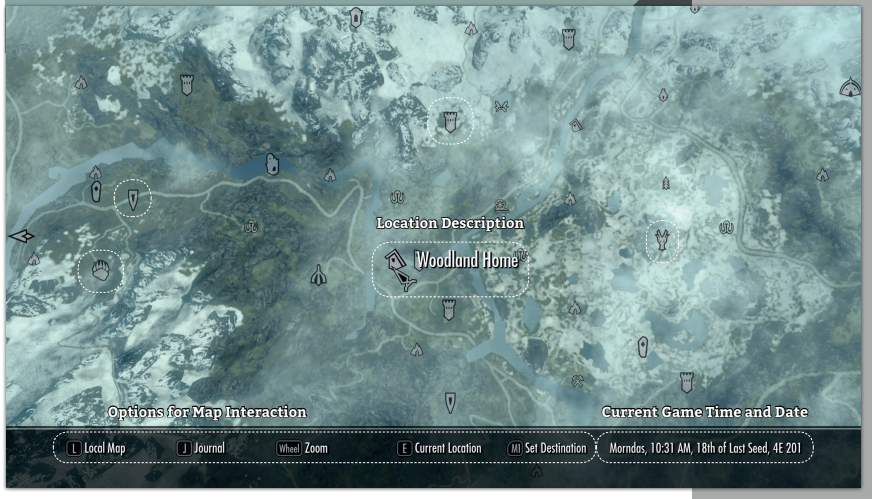
The Elder Scrolls 5: Skyrim
2011, Bethesda Game Studios, Bethesta Softworks

Skill Overview



The Elder Scrolls 5: Skyrim
2011, Bethesda Game Studios, Bethesta Softworks

World Map



Appendix D

Design Concepts, Paper Prototype and Session 1

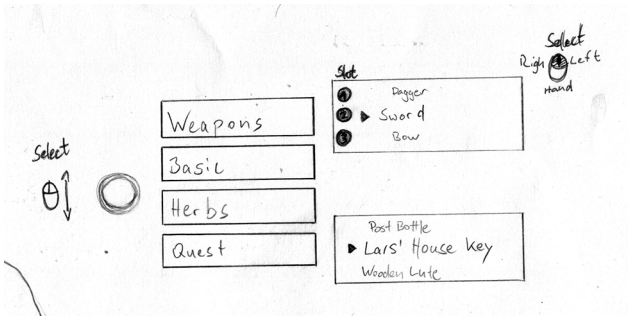


FIGURE D.1: Table Inventory

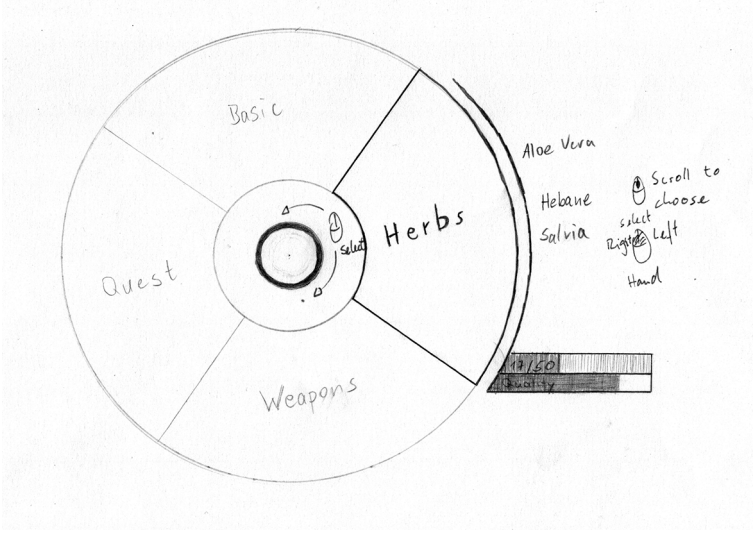


FIGURE D.2: Circle Inventory

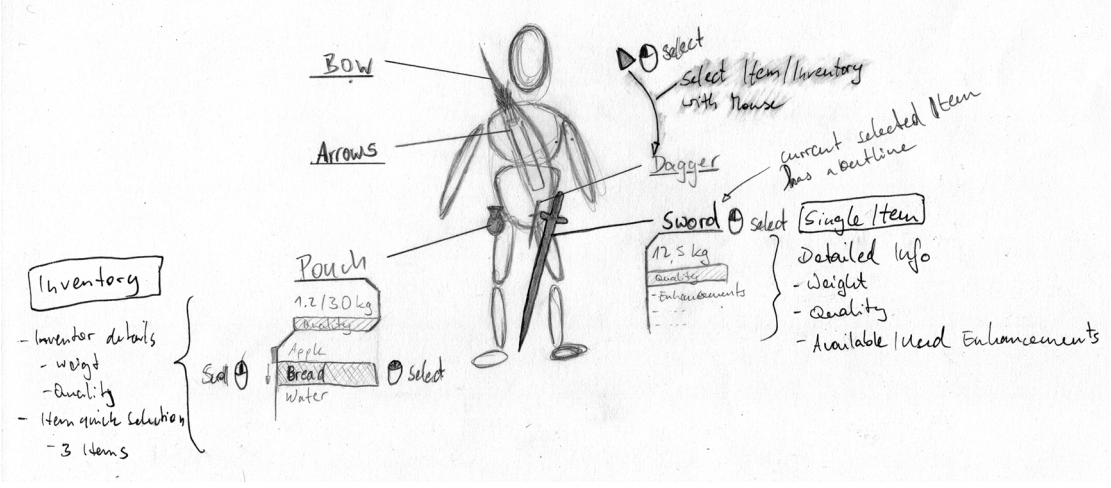


FIGURE D.3: Ingame Inventory

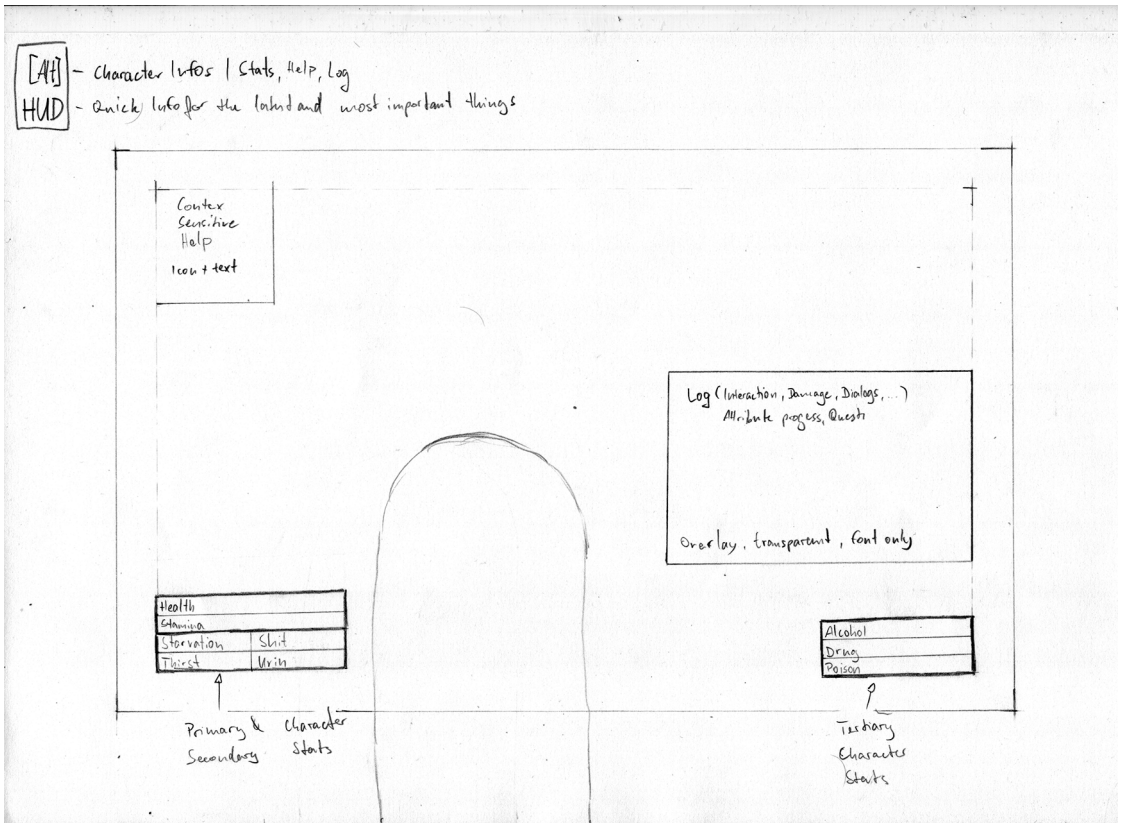


FIGURE D.4: Character Info HUD

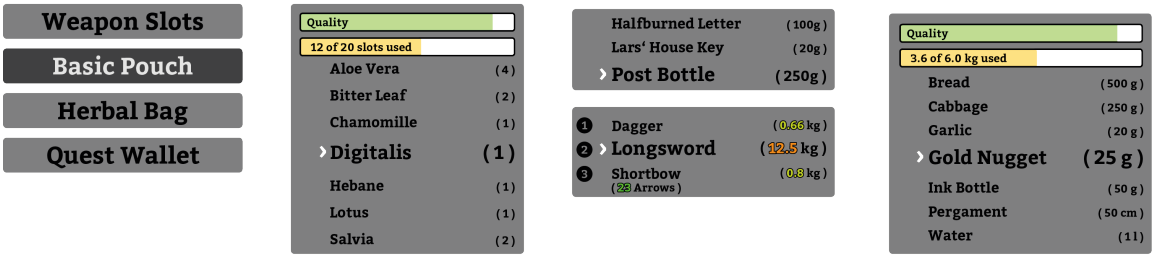


FIGURE D.5: Digital table UI for the inventory

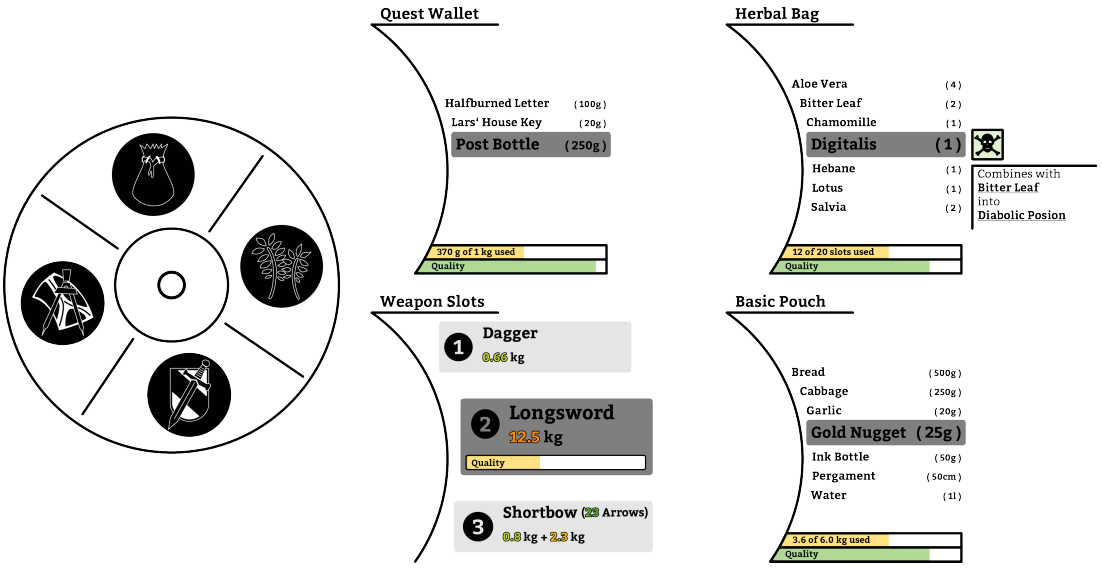


FIGURE D.6: Digital circle UI for the inventory

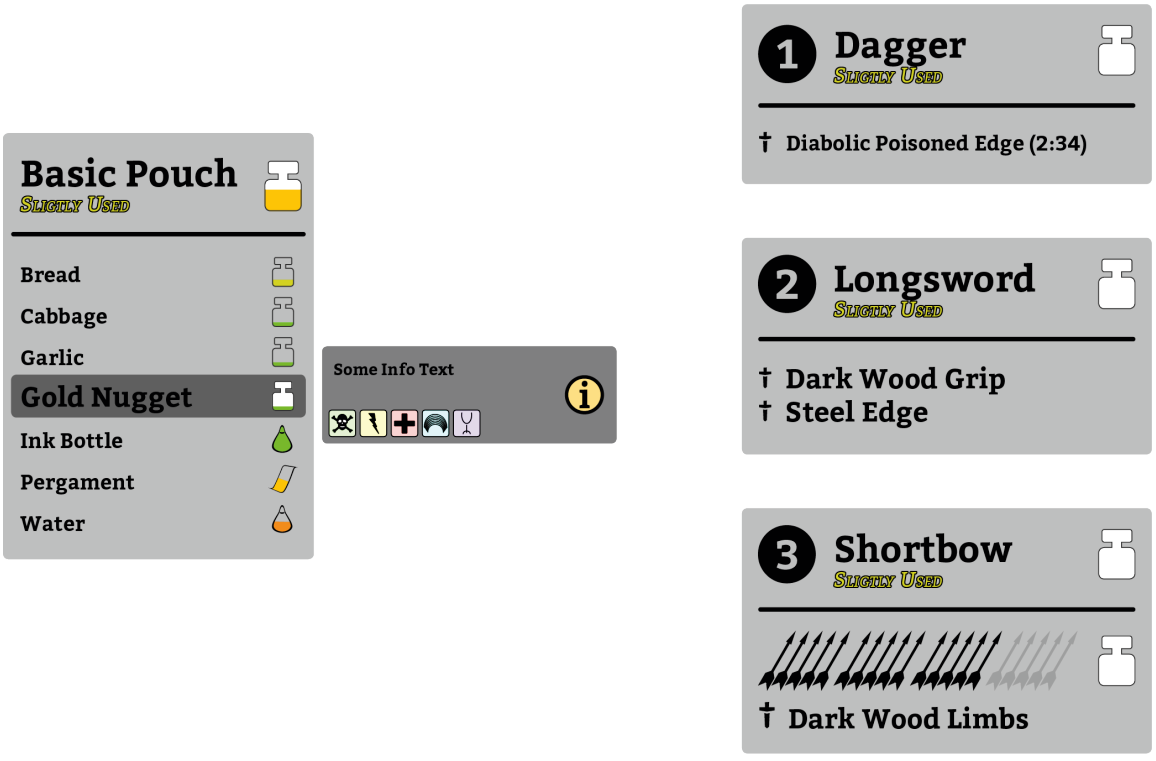


FIGURE D.7: Digital in-game UI for the inventory

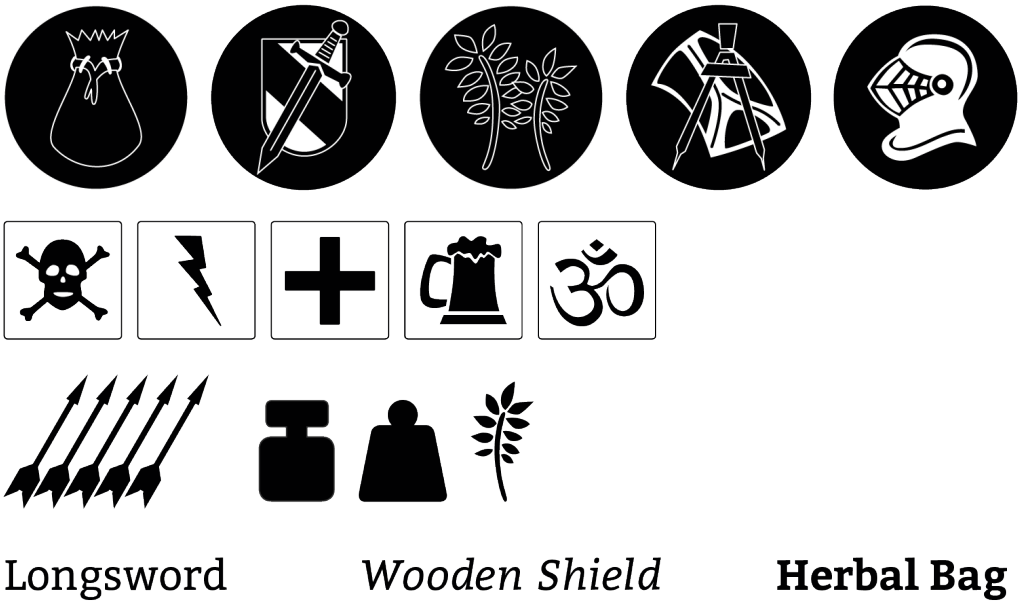


Figure D.8: Some selected icons and fonts

How much do you play during a day?

- ○ less than 1 hour
- ○ between 1 and 2 hours
- ○ between 2 and 3 hours
- ○ between 3 and 4 hours
- ○ between 4 and 5 hours
- ○ between 5 and 6 hours
- ○ between 6 and 7 hours
- ○ between 7 and 8 hours
- ○ more than 8 hours

On which platforms do you play?

iPhone	Playstation 3	Linux	Wii U
iPad	Playstation 4	Android Phone	Nintedo DS
XBOX 360	PC	Android Tablet	Nintendo 3DS
XBOX ONE	Mac	Wii	

Rank the following game genres according to your personal interest (1 - 10)

- ○ First Person Shooter (FPS, Singleplayer Only)
- ○ Role-playing games (RPG, Singleplayer Only)
- ○ Action Adventure (Jump'n'Run, Survival, Stealth)
- ○ Adventure (Real Time 3D, Graphic, Point & Click, Visual Novels)
- ○ Simulation (Life, Economics, Infrastructure, Construction)
- ○ Stategy & Tactics (RTS, RTT, Tower Defence, Turn-based)
- ○ Competitive (Sports, MOBA, Beat'em'up, Fighting, Multiplayer; FPS, RTS)
- ○ Massivle Multiplayer Online Games (MMOs in all kinds)
- ○ Browser games (Flash, HTML, Java, etc.)
- ○ Puzzle, Reflex, Card, Sort & Order, Casual

Which of the following Role-playing games have you played?

The Witcher	The Legend Of Zelda: Twilight Princess
The Elder Scrolls 5: Skyrim	Risen
Final Fantasy XII	Diablo 2
Hellgate: London	Mass Effect
The Witcher 2	The Legend Of Zelda: Phantom Hourgalss
Dragon Age: Origins	Risen 2
Final Fantasy XIII	Diablo 3
Torchlight	Mass Effect 2
Gothic 1	The Legend Of Zelda: Spirit Tracks
Dragon Age 2	Risen 3
Final Fantasy XIII-2	Fallout 3
Dungeon Siege	Mass Effect 3
Gothic 2	The Legend Of Zelda: Skyward Sword
World of Warcraft	The Elder Scrolls 3: Morrowind
Lightning Returns: Final Fantasy XIII	Fallout New Vegas
Dungeon Siege 2	Chrono Trigger
Gothic 3	The Legend Of Zelda: A Link Between
Diablo 1	Worlds
Final Fantasy XIV	The Elder Scrolls 4: Oblivion

Game UI Prototyping with Unreal Engine 4
Bachelor of Science (B.Sc.) Thesis

Survey & Expert Evaluation 12.11.14

		1				5
1	Less UI is better UI.	☐	☐	☐	☐	☐
2	A good UI explains itself.	☐	☐	☐	☐	☐
3	The game have to explain the UI.	☐	☐	☐	☐	☐
4	I want to customize my UI.	☐	☐	☐	☐	☐
5	I want to find help at any time.	☐	☐	☐	☐	☐
6	I want to make errors.	☐	☐	☐	☐	☐
7	I need to know what errors i can make.	☐	☐	☐	☐	☐
8	I want to know what the result will be.	☐	☐	☐	☐	☐
9	I do not need several ways to do the same thing.	☐	☐	☐	☐	☐
10	Consistent and thematic colors are important.	☐	☐	☐	☐	☐
11	Consistent and thematic fonts are important.	☐	☐	☐	☐	☐
12	Consistent and thematic sounds are important.	☐	☐	☐	☐	☐
13	Consistent and thematic icons are important.	☐	☐	☐	☐	☐

		1				5
14	Hiding information the game already has from the player makes a game more annoying than difficult.	☐	☐	☐	☐	☐
15	I want to see as much information as needed to make progress throughout the game.	☐	☐	☐	☐	☐
16	The UI in a Game is important.	☐	☐	☐	☐	☐
17	If the UI is not good, the game is less fun.	☐	☐	☐	☐	☐
18	I do not care about the UI as long as the game is fun.	☐	☐	☐	☐	☐
19	I am willed to learn a rather complex UI for myself.	☐	☐	☐	☐	☐
20	I am willed to learn a rather complex UI through the game.	☐	☐	☐	☐	☐
21	I have learned different new UIs in the past.	☐	☐	☐	☐	☐
22	I feel confident in exploring and using a new UI.	☐	☐	☐	☐	☐
23	I expect more from a UI as time goes on and new things are developed.	☐	☐	☐	☐	☐

1	Strongly Disagree
2	Disagree
3	Cannot Decide
4	Agree
5	Strongly Agree

Subject Number	1	2	3	4	5	6
Age	22	25	24	27	27	25
Gender	male	male	male	male	male	male

Subject Number	Play Duration	Max. h
P1	between 2 and 3 hours	3
P2	less than 1 hour	1
P3	between 2 and 3 hours	3
P4	between 1 and 2 hours	2
P5	between 2 and 3 hours	3
P6	between 3 and 4 hours	4
Avg.		2,7
Median		3

How much do you play during a day?

Options:

less than 1 hour
between 1 and 2 hours
between 2 and 3 hours
between 3 and 4 hours
between 4 and 5 hours
between 5 and 6 hours
between 6 and 7 hours
between 7 and 8 hours
more than 8 hours

Subject Number	P1	P2	P3	P4	P5	P6
iPhone	No	No	No	No	No	No
iPad	No	No	No	No	No	No
XBOX 360	Yes	No	No	No	No	Yes
XBOX ONE	No	No	No	No	No	No
Playstation 3	No	No	Yes	No	No	Yes
Playstation 4	Yes	No	No	No	No	Yes
PC	Yes	Yes	Yes	Yes	Yes	Yes
Mac	No	Yes	No	No	No	No
Linux	No	No	No	No	No	No
Android Phone	No	Yes	No	Yes	Yes	No
Android Tablet	No	No	No	No	Yes	No
Wii	No	No	Yes	No	Yes	Yes
Wii U	No	No	Yes	No	No	No
Nintedo DS	No	No	No	No	No	No
Nintendo 3DS	Yes	No	No	No	No	Yes

On which platforms do you play?

Options:

Yes
No

Genre	P1	P2	P3	P4	P5	P6
Action Adventure (Jump'n'Run, Survival, Stealth)	2	2	1	5	4	5
Adventure (Real Time 3D, Graphic, Point & Click, Visual Novels)	4	3	4	7	6	4
Browser games (Flash, HTML, Java, etc.)	8	6	10	9	10	8
Competitive (Sports, MOBA, Beat'em'up, Fighting, Multiplayer; FPS, RTS)	10	10	5	6	2	6
First Person Shooter (FPS, Singleplayer Only)	3	7	8	8	1	10
Massivle Multiplayer Online Games (MMOs in all kinds)	7	9	3	4	9	3
Puzzle, Reflex, Card, Sort & Order, Casual	9	8	9	10	8	9
Role-playing games (RPG, Singleplayer Only)	1	1	2	2	3	1
Simulation (Life, Economics, Infrastructure, Construction)	6	5	6	3	7	7
Strategy & Tactics (RTS, RTT, Tower Defence, Turn-based)	5	4	7	1	5	2

Rank the following game genres according to your personal interest (1 - 10)

Genre	Median	Average
Action Adventure (Jump'n'Run, Survival, Stealth)	3	3,2
Adventure (Real Time 3D, Graphic, Point & Click, Visual Novels)	4	4,7
Browser games (Flash, HTML, Java, etc.)	9	8,5
ompetitive (Sports, MOBA, Beat'em'up, Fighting, Multiplayer; FPS, RTS)	6	6,5
First Person Shooter (FPS, Singleplayer Only)	8	6,2
Massivle Multiplayer Online Games (MMOs in all kinds)	6	5,8
Puzzle, Reflex, Card, Sort & Order, Casual	9	8,8
Role-playing games (RPG, Singleplayer Only)	2	1,7
Simulation (Life, Economics, Infrastructure, Construction)	6	5,7
Strategy & Tactics (RTS, RTT, Tower Defence, Turn-based)	5	4,0

Median and Average Ranking of Genres

Game Title	P1	P2	P3	P4	P5	P6
The Witcher	X		X		X	
The Elder Scrolls 5:	X		X	X	X	
Final Fantasy XII					X	X
Hellgate: London					X	
The Witcher 2	X	X	X	X		
Dragon Age: Origins	X	X			X	X
Final Fantasy XIII					X	X
Torchlight					X	
Gothic 1					X	X
Dragon Age 2			X			X
Final Fantasy XIII-2						X
Dungeon Siege					X	
Gothic 2	X	X		X	X	X
World of Warcraft	X		X	X	X	
Lightning Returns: Final Fantasy XIII						
Dungeon Siege 2					X	
Gothic 3		X		X	X	
Diablo 1			X		X	
Final Fantasy XIV						
The Legend Of Zelda: Twilight Princess	X		X		X	X
Risen					X	X
Diablo 2			X		X	
Mass Effect	X				X	X
The Legend Of Zelda: Phantom Hourgalss						
Risen 2						
Diablo 3		X	X		X	
Mass Effect 2	X	X	X	X	X	X
The Legend Of Zelda:						
Risen 3						
Fallout 3	X				X	
Mass Effect 3	X			X		X
The Legend Of Zelda: Skyward Sword			X			X
The Elder Scrolls 3:	X	X	X			X
Fallout New Vegas	X					
Chrono Trigger		X			X	
The Legend Of Zelda: A Link Between			X			
Worlds						
The Elder Scrolls 4: Oblivion	X		X	X	X	
Deus Ex 1						
Deus Ex 2	X					
Deus Ex 3	X					
Lufia					X	
Syndicate					X	

Question No.	Subject Number	1	2	3	4	5	6	Avg	Min	Max	Median
1	*Less UI is better UI.*	4	4	4	3	4	5	4,0	3	5	4
2	*A good UI explains itself.*	5	5	5	5	5	5	5,0	5	5	5
3	*The game have to explain the UI.*	3	2	1	2	5	3	2,7	1	5	3
4	*I want to customize my UI.*	2	2	4	2	3	4	2,8	2	4	3
5	*I want to find help at any time.*	4	2	2	5	5	4	3,7	2	5	4
6	*I want to make errors.*	4	1	3	1	5	4	3,0	1	5	4
7	*I need to know what errors i can make.*	3	1	4	5	4	4	3,5	1	5	4
8	*I want to know what the result will be.*	5	3	5	5	2	5	4,2	2	5	5
9	*I do not need several ways to do the same thing.*	2	2	3	4	3	4	3,0	2	4	3
10	*Consistent and thematic colors are important.*	4	5	5	4	5	5	4,7	4	5	5
11	*Consistent and thematic fonts are important.*	4	4	5	4	5	5	4,5	4	5	5
12	*Consistent and thematic sounds are important.*	5	4	5	4	5	5	4,7	4	5	5
13	*Consistent and thematic icons are important.*	5	5	5	4	5	5	4,8	4	5	5
14	*Hiding information the game already has from the player makes a game more annoying than difficult.*	2	5	4	3	4	5	3,8	2	5	4
15	*I want to see as much information as needed to make progress throughout the game.*	3	5	4	5	3	5	4,2	3	5	5
16	*The UI in a game is important.*	5	5	5	4	5	5	4,8	4	5	5
17	*If the UI is not good, the game is less fun.*	3	4	5	4	4	5	4,2	3	5	4
18	*I do not care about the UI as long as the game is fun.*	1	3	2	4	2	4	2,7	1	4	3
19	*I am willed to learn a rather complex UI for myself.*	3	4	4	3	3	5	3,7	3	5	4
20	*I am willed to learn a rather complex UI through the game.*	5	5	2	2	5	5	4,0	2	5	5
21	*I have learned different new UIs in the past.*	4	5	4	5	5	5	4,7	4	5	5
22	*I feel confident in exploring and using a new UI.*	5	5	4	5	5	4	4,7	4	5	5
23	*I expect more from a UI as time goes on and new things are developed.*	3	4	5	4	4	5	4,2	3	5	4

Table D.1: Session 1 – Findings for the table UI prototype

Subject	**Findings**
2 3 4	better as a windows approach (movable, close, minimize, etc.)
1 3	better use tabs
1 2 4 5 6	menu structure too deep
6	use arrow keys to navigate

Table D.2: Session 1 – Findings for the circle UI prototype

Subject	**Findings**
1 3	interaction with mouse, point and click, drag and drop, hover and focus
2	No rotation, static circle. 2nd level info on left/right/top/bottom
2	more character info is accessible in the centre as an additional icon
5	if items can combine, highlight the components (e.g. herbs, crafting)

Table D.3: Session 1 – Findings for the in-game UI prototype

Subject	**Findings**
3	items and weapons are better comparable
2 4 5	use different camera perspectives
6	additional to perspectives, zoom onto the item/weapon/inventory
5	combine in-game and circle inventory
6	hover over in-game item
6	show only one item in detail (or toggle with click)

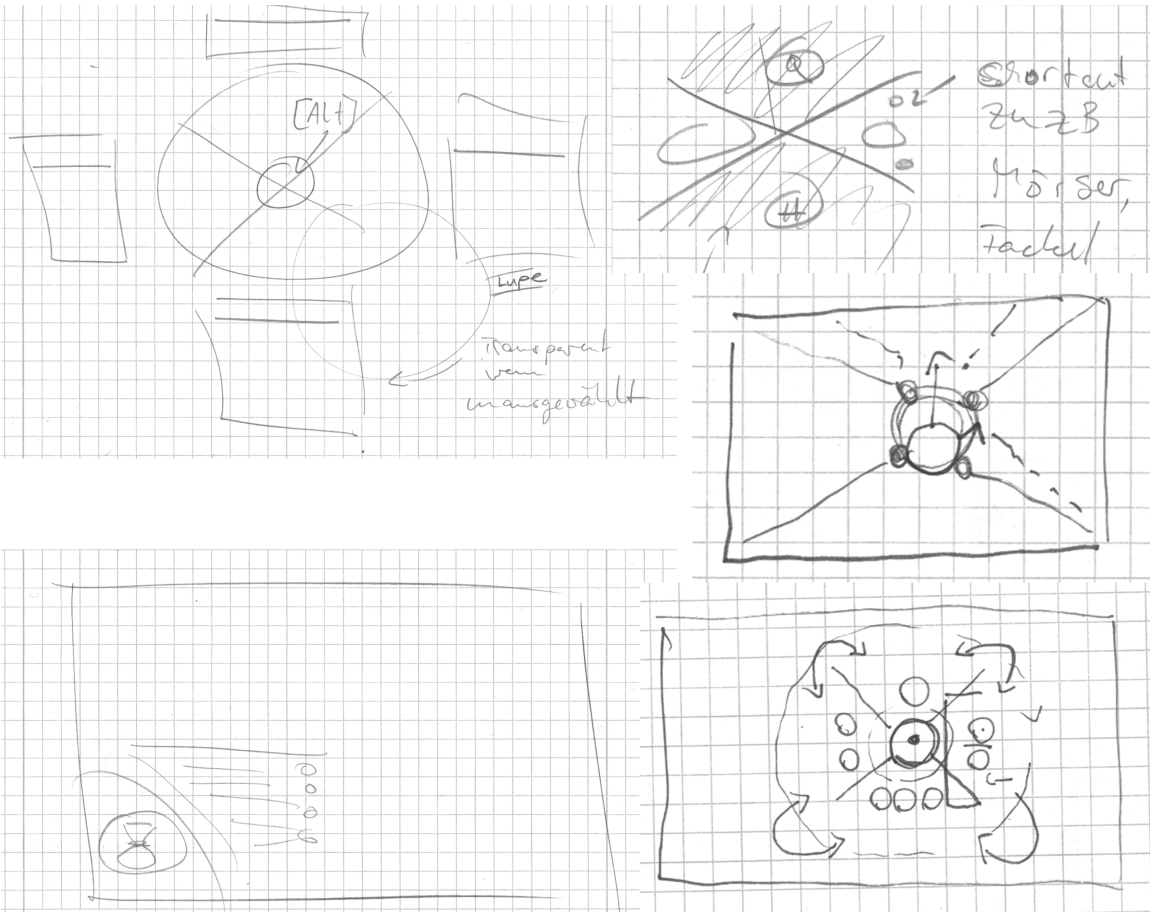


FIGURE D.9: Concepts created along the paper prototyping

Table D.4: Session 1 – General findings

Subject	**Category**	**Findings**
1 2 4	UI	numeric values better for amount
1	UI	sans serif font
1	UI	overlays for menus
1 2 6	Game Mechanics	pause when inventory open
2	Game Mechanics	slow motion when inventory open
3 4	Game Mechanics	real-time when inventory open
2	Game Mechanics & UI	mark/display current selected hand items
2	UI	make HUD bars (health, stamina, etc.) visible all the time
2	Game Mechanics & UI	game log in HUD as book (as in-game world model)
2	UI	less HUD bars, icons for critical states (like poisoned, drunken, starve, thirst, etc.)
2	UI	left-handed and right-handed have different expectations
1 2 4 5	UI	[I], [Mouse 3] or [TAB] to open inventory
6	UI	[I] to open inventory
3	UI	[Mouse 3] to open inventory
2	Game Mechanics & UI	make weapons change name (e.g. on murdering)
2	UI	make item name changeable
2	Game Mechanics & UI	make shields have customizable coat of arms
2	Game Mechanics & UI	key ring as a item, in-game object and UI element
2 5	Game Mechanics & UI	quality is visualized by the item name's font, it corrodes the lower quality/durability is
3	Game Mechanics & UI	hold [TAB] to keep inventory open
3	UI	items must be comparable, especially on pick-up
3	UI	show buffs (for character, weapon, etc.) in 1st level in UI
3	Game Mechanics & UI	can the food get mouldy (visual feedback like flies?)
5	UI	mouse wheel to change weapons
5	UI	use a specific key to draw weapons
5	UI	make sorting possible
5	UI	use more icons
5	UI	when close and reopen inventory, remember last state
6	Game Design	recipes for crafting or cooking
6	Game Design & UI	use specific wording to indicate item attributes/quality/durability
6	UI	display (item) interaction options
6	UI	display amount of item type (all items of a kind) in inventory fill level
6	Game Mechanics & UI	make items/weapons get rusty or used in-game
6	Game Design	crafting only on defined objects (e.g. on a herbs cabinet)

Appendix E

First Refinement and Session 2

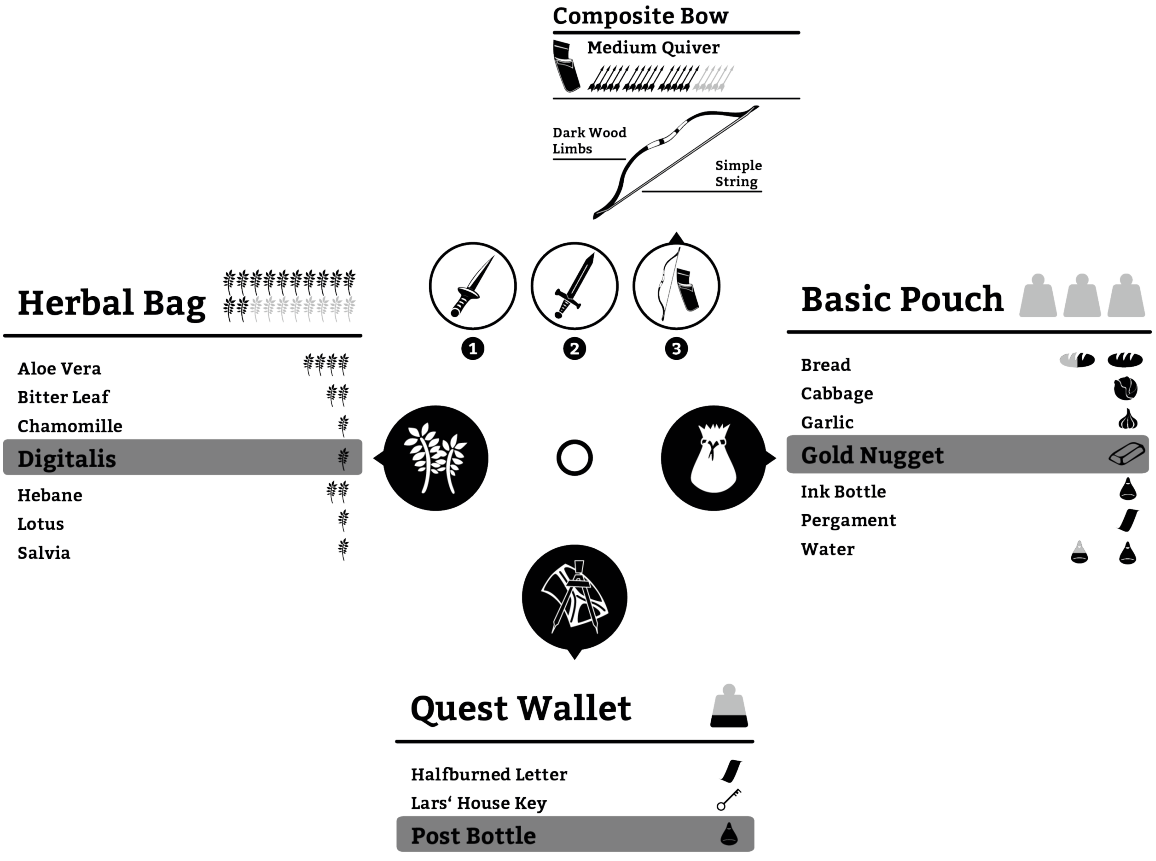


FIGURE E.1: First re-design of the circle UI

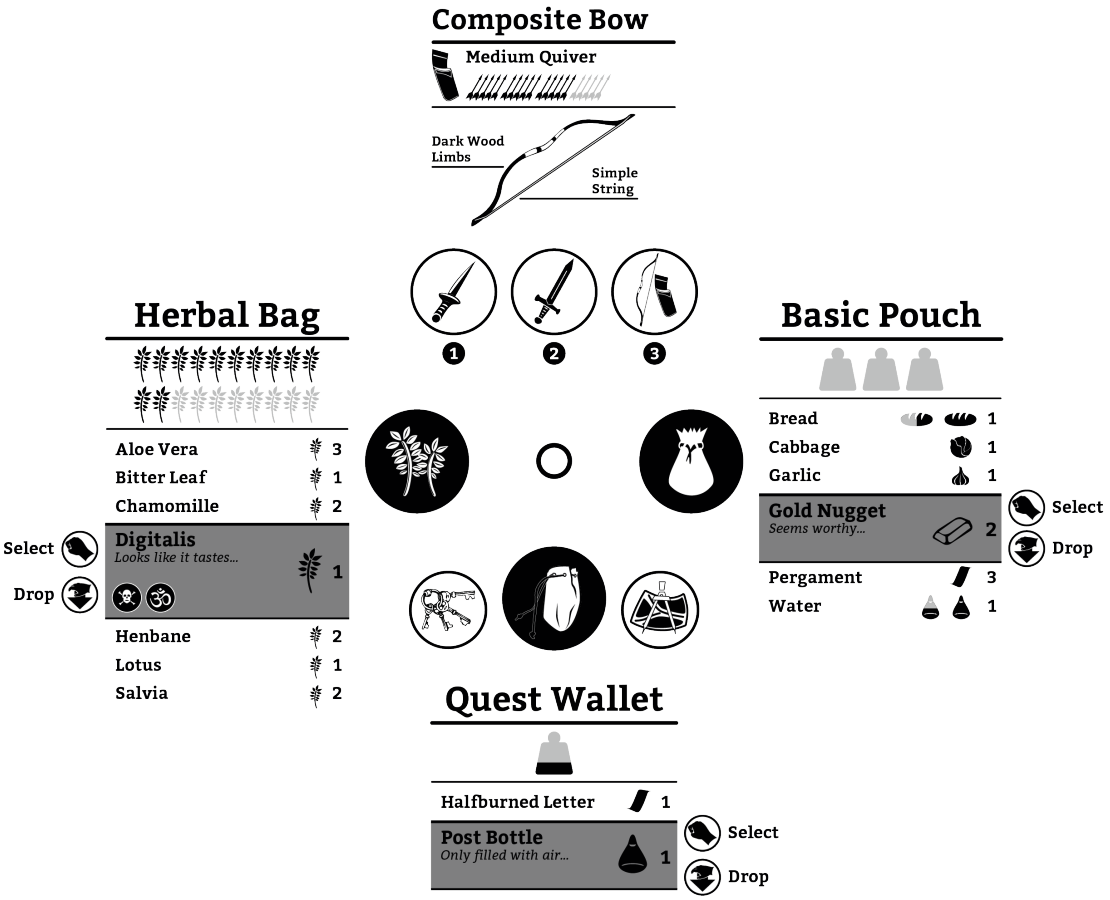


FIGURE E.2: Enhanced re-design of the circle UI

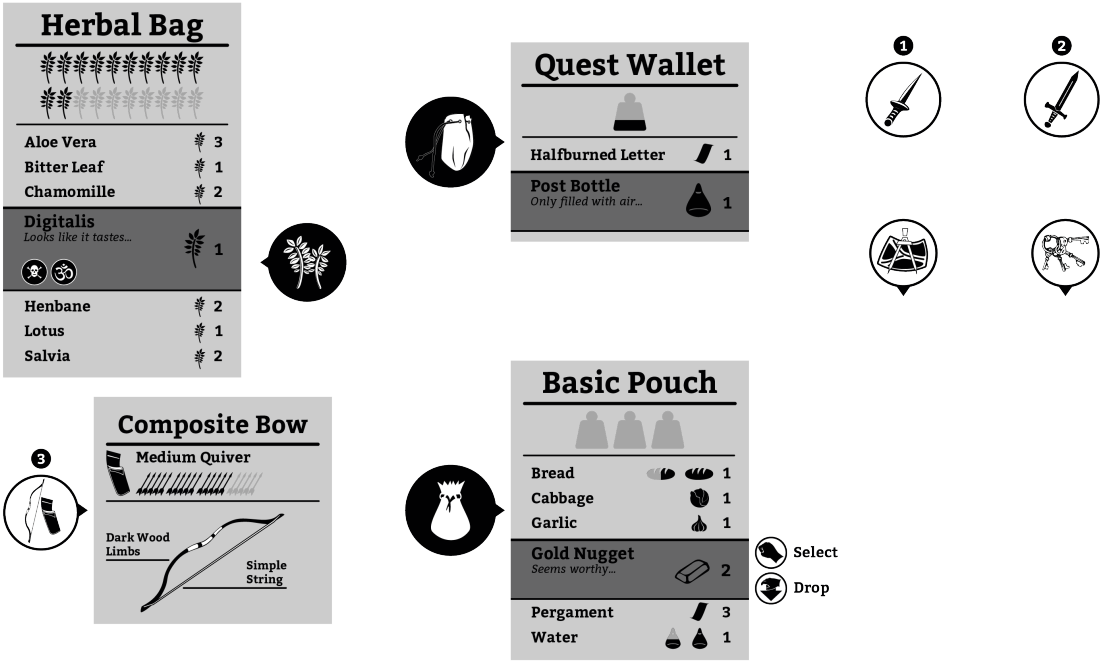


FIGURE E.3: Re-design of the in-game UI

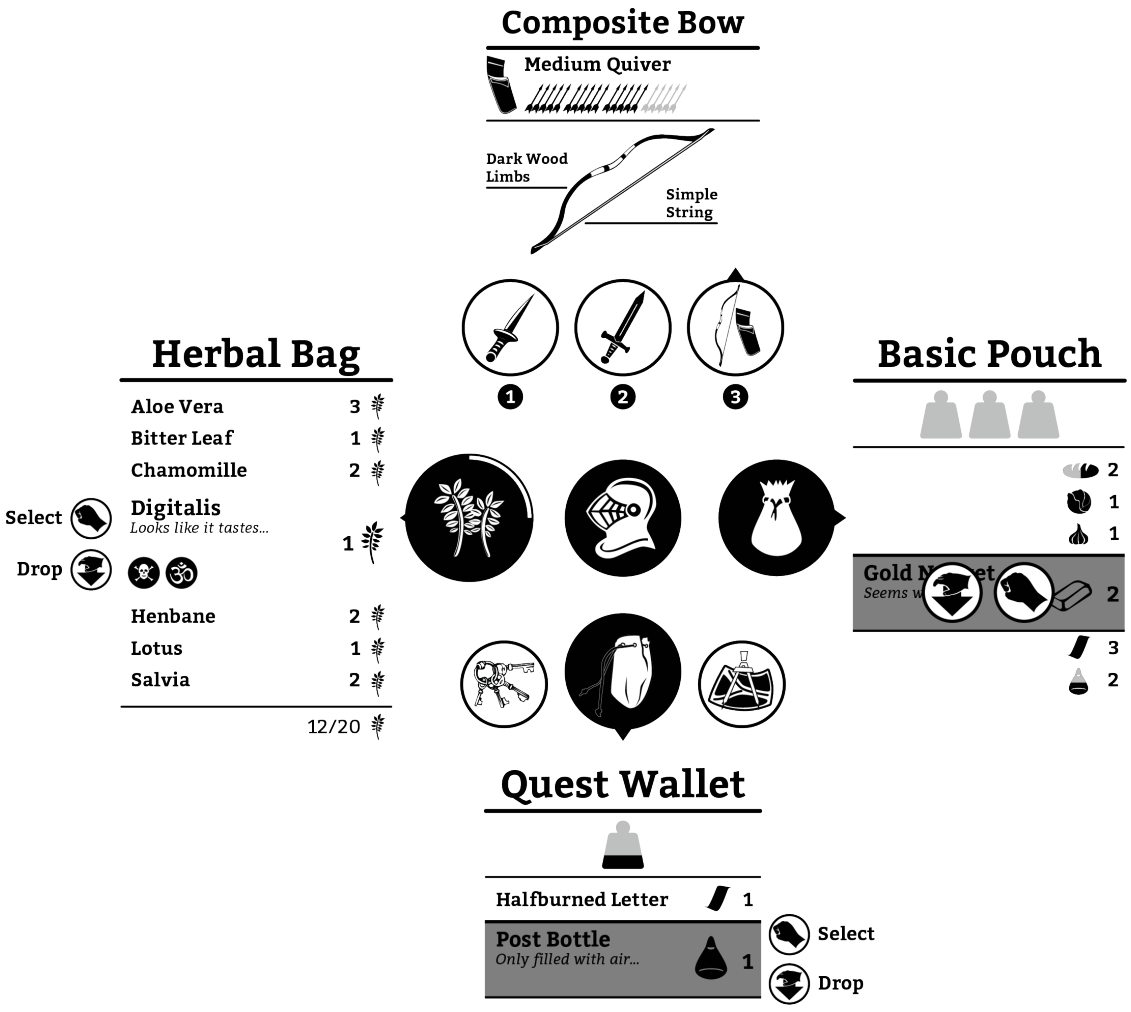


FIGURE E.4: Enhancements created in the team prototyping session

Table E.1: Session 2 – Findings for circle UI gathered during team prototyping

Category	**Findings**
UI	use centre for a reference icon for armour, other equipment or character info
UI	remove centre point from inventory
UI	use centre to display feedback
UI	find approaches for better scale with more inventories, weapons and items
UI	display inventory fill-level at the bottom right in the 2nd level menu
UI	good use of space, allows everything to be visible on one level
UI	use combination of icon and number
UI	display fill-level of inventory either with icon-fill or numerical
UI	make fill-level visualization smaller and more compact on right side
UI	display fill-level as circular progress bar
UI	inventory content must offer a scroll-bar if content is too large
UI	reduce mouse movement distances
UI	overlay interaction options for items, make use of transparency
UI	tool-tips on hovering, for interactions and item details (poison, etc.)
UI	hot-keys for weapons, certain items, map, log, etc.
UI	colours for icons, warning or error feedback
UI	contrast the UI from background (DoF, Blur, Overlay)
UI & Game Mechanics	use a separate menu for weapon enhancing
UI	add tool-tips for a weapons components
UI & Game Design	less numbers for attributes, use a description on what advantages enhancements/components give
UI	weapon or item preview 3D model or drawing
UI	centre weapon details at the top
UI & Game Mechanics	dynamic hot-keys, make customization easy, display connection between hot-key and item
UI	circular vs. rectangular layout
UI	layout like a character/person shape
UI	combine item consumption and amount of item type
UI & Game Mechanics	support learning processes
UI	experiment with simplicity vs. informations and aesthetics
UI	show item traits/attributes below description

Table E.2: Session 2 – Findings for in-game UI gathered during team prototyping

Category	**Findings**
UI & Game Mechanics	more immersive and realistic, but interaction concept to clumsy
Game Mechanics	use animation and sounds for feedback
UI	interaction is to slow for power users and can get annoying with frequent usage
UI & Game Design	use as a more detailed view on the character, instead of a fast inventory
UI	no improvement or advantage using this design
UI	to much actions needed to achieve goals

Appendix F

Session 3: Heuristics & Usability Expert Evaluation

Table F.1: Heuristics on controls, settings, sensitivity, responsiveness and customization

ID	Heuristics
A	Settings should be customizable; the controls are easy to manage, minimized, mapped in a natural or common way and intuitive, while having an appropriate level of sensitivity and responsiveness.
A1	Controls should be customizable and default to industry standard settings.
A2	Allow users to customize video and audio settings, difficulty and game speed.
A3	Controls should be intuitive and mapped in a natural way.
A4	Provide controls that are easy to manage, and that have an appropriate level of sensitivity and responsiveness.
A5	Minimize control options.
A6	Mechanics should feel natural and have correct weight and momentum.

Table F.2: Heuristics on learning, error prevention and recovery

ID	Heuristics
B	Provide instructions, training, warnings, help and means for errors and recovery – do not expect the user read the manual!
B1	Provide instructions, training, and help.
B2	Do not expect the user to read a manual.
B3	Provide means for error prevention and recovery through the use of warning messages.

Table F.3: Heuristics on aesthetics and immersion

ID	Heuristics
C	The interface is non-intrusive, diegetic and convenient where possible, provides unobstructed views on the players state and actions and supports in immersion.
C1	The interface should be as non-intrusive as possible.
C2	A player should always be able to identify their score/status in the game.
C3	Provide unobstructed views that are appropriate for the user's current actions.
C4	Follow the trends set by the gaming community to shorten the learning curve.
C5	Minimize control options.
C6	Art should speak to its function.
C7	Players should be able to save games in different states.
C8	Get the player involved quickly and easily.

Table F.4: Heuristics on feedback and cognitive load

ID	Heuristics
D	The feedback is given immediately and uses multiple convenient layers; visual representations are easy to interpret, provide consistent response and use minimized possible interface depth.
D1	Feedback should be given immediately to display user control.
D2	Provide visual representations that are easy to interpret and that minimize the need for micromanagement.
D3	Provide consistent responses to the user's actions.
D4	Interfaces should be consistent in control, colour, typography, and dialogue design.
D5	For PC games, consider hiding the main computer interface during game play.
D6	Minimize the menu layers of an interface.
D7	Use sound to provide meaningful feedback.
D8	Allow users to skip non-playable and frequently repeated content.

Game heuristics from the literature according to Federoff (2004)

Game Interface

- Controls should be customizable and default to industry standard settings (Bickford, 1997; Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
- Controls should be intuitive and mapped in a natural way (Pinelle et al., 2008; Ferderoff, 2004)
- Minimize control options (Shelley, 2001)
- The interface should be as non- intrusive as possible (Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
- A player should always be able to identify their score/status in the game (Malone, 1982; Shneiderman, 1992; Ferderoff, 2004; Pinelle et al., 2008)
- Follow the trends set by the gaming community to shorten the learning curve (Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
- Interfaces should be consistent in control, color, typography, and dialog design (Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
- For PC games, consider hiding the main computer interface during game play (Bickford, 1997; Ferderoff, 2004)
- Minimize the menu layers of an interface (Shelley, 2001; Ferderoff, 2004)
- Use sound to provide meaningful feedback (Norman, 1990; Ferderoff, 2004)
- Do not expect the user to read a manual (Norman, 1990; Ferderoff, 2004)
- Provide means for error prevention and recovery through the use of warning messages (Ferderoff, 2004)
- Feedback should be given immediately to display user control (Bickford, 1997;
- Malone, 1982; Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
- Provide consistent responses to the user's actions. (Pinelle et al., 2008)
- Allow users to customize video and audio settings, difficulty and game speed. (Pinelle et al., 2008)
- Provide unobstructed views that are appropriate for the user's current actions. (Pinelle et al., 2008)
- Allow users to skip non-playable and frequently repeated content. (Pinelle et al., 2008)
- Provide controls that are easy to manage, and that have an appropriate level of sensitivity and responsiveness. (Pinelle et al., 2008)
- Provide instructions, training, and help. (Pinelle et al., 2008)
- Provide visual representations that are easy to interpret and that minimize the need for micromanagement. (Pinelle et al., 2008)

Game Interface: Heuristics	References
Settings should be *customizable; the controls are easy to manage, minimized,* mapped in a *natural* or *common* way and *intuitive*, while having an appropriate level of *sensitivity* and *responsiveness.*	
Controls should be customizable and default to industry standard settings.	(Bickford, 1997; Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
Allow users to customize video and audio settings, difficulty and game speed.	(Pinelle et al. 2008)
Controls should be intuitive and mapped in a natural way.	(Pinelle et al., 2008; Ferderoff, 2004)
Provide controls that are easy to manage, and that have an appropriate level of sensitivity and responsiveness.	(Pinelle et al., 2008)
Minimize control options.	(Shelley, 2001)
Mechanics should feel natural and have correct weight and momentum	(Ferderoff, 2004)

Game Interface: Heuristics	References
Provide *instructions, training, warnings, help* and *means for errors and recovery* - do not expect the user read the manual!	
Provide instructions, training, and help.	(Pinelle et al., 2008)
Do not expect the user to read a manual.	(Norman, 1990; Ferderoff, 2004)
Provide means for error prevention and recovery through the use of warning messages.	(Ferderoff, 2004)

10.01.15

Game Interface: Heuristics	References
The interface is *non-intrusive, diegetic* and *convenient* where possible, provides *unobstructed views* on the players *state* and *actions* and *supports in immersion.*	
The interface should be as non-intrusive as possible.	(Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
A player should always be able to identify their score/ status in the game.	(Malone, 1982; Shneiderman, 1992; Ferderoff, 2004; Pinelle et al., 2008)
Provide unobstructed views that are appropriate for the user's current actions.	(Pinelle et al. 2008)
Follow the trends set by the gaming community to shorten the learning curve.	(Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
Minimize control options.	(Shelley, 2001)
Art should speak to its function	(Ferderoff, 2004)
Players should be able to save games in different states.	(Ferderoff, 2004)
Get the player involved quickly and easily	(Bickford, 1997; Clanton, 1998; Sanchez-Crespo Dalmau, 1999; Shelley, 2001, Ferderoff, 2004)

Game Interface: Heuristics	References
The *feedback is given immediately* and uses *multiple convenient layers*; visual representations are *easy to interpret*, provide *consistent response* and use *minimized possible interface depth.*	
Feedback should be given immediately to display user control.	(Bickford, 1997; Malone, 1982; Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
Provide visual representations that are easy to interpret and that minimize the need for micromanagement.	(Pinelle et al., 2008)
Provide consistent responses to the user's actions.	(Pinelle et al. 2008)
Interfaces should be consistent in control, color, typography, and dialog design.	(Sanchez-Crespo Dalmau, 1999; Ferderoff, 2004)
For PC games, consider hiding the main computer interface during game play.	(Bickford, 1997; Ferderoff, 2004)
Minimize the menu layers of an interface.	(Shelley, 2001; Ferderoff, 2004)
Use sound to provide meaningful feedback.	(Norman, 1990; Ferderoff, 2004)
Allow users to skip non-playable and frequently repeated content.	(Pinelle et al., 2008)

Game UI Prototyping with Unreal Engine 4

Bachelor of Science (B.Sc.) Thesis

Heuristic Evaluation

10.01.15

Number _____

Age _____

Gender _____

Notes

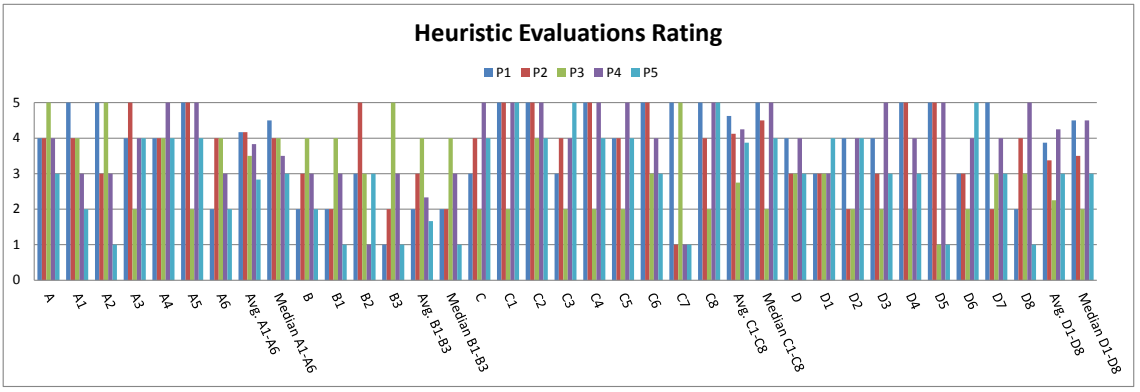
A	1 ☐☐☐☐☐ 5		C4	1 ☐☐☐☐☐ 5
A1	1 ☐☐☐☐☐ 5		C5	1 ☐☐☐☐☐ 5
A2	1 ☐☐☐☐☐ 5		C6	1 ☐☐☐☐☐ 5
A3	1 ☐☐☐☐☐ 5		C7	1 ☐☐☐☐☐ 5
A4	1 ☐☐☐☐☐ 5		C8	1 ☐☐☐☐☐ 5
A5	1 ☐☐☐☐☐ 5		D	1 ☐☐☐☐☐ 5
A6	1 ☐☐☐☐☐ 5		D1	1 ☐☐☐☐☐ 5
B	1 ☐☐☐☐☐ 5		D2	1 ☐☐☐☐☐ 5
B1	1 ☐☐☐☐☐ 5		D3	1 ☐☐☐☐☐ 5
B2	1 ☐☐☐☐☐ 5		D4	1 ☐☐☐☐☐ 5
B3	1 ☐☐☐☐☐ 5		D5	1 ☐☐☐☐☐ 5
C	1 ☐☐☐☐☐ 5		D6	1 ☐☐☐☐☐ 5
C1	1 ☐☐☐☐☐ 5		D7	1 ☐☐☐☐☐ 5
C2	1 ☐☐☐☐☐ 5		D8	1 ☐☐☐☐☐ 5
C3	1 ☐☐☐☐☐ 5			

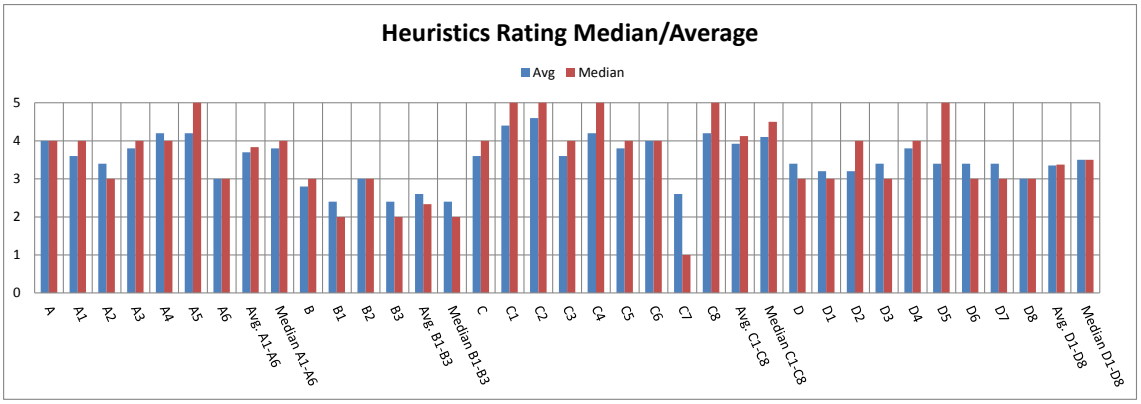
Table F.5: Session 3 – Findings recorded during the heuristic evaluation

Category	**Findings**
UI	use double [E] to pick-up, or better left mouse click
UI & Game Mechanics	use more animations
UI	reset mouse to the centre when entering inventory
UI	use more mouse over effects
UI	display slot names
UI	rearrange slots
UI	change open/close inventory mouse interactions
UI	add feedback if no interaction is possible with an item in world
UI & Game Design	if inventory is not available in fights, display according feedback
UI	show hand item at 1st level
UI	show inventory content all the time, no need to open/close
UI	improve drag and drop, add more hover effects
UI	adjust inventory size for the bottom inventory
UI	display the shortcuts (for weapons) all the time
UI	make item stacks move with [Shift] drag and drop
UI	display the weight of an item numerical
UI	add feedback if a drop is possible
UI	drag and drop items out of inventory to drop them from inventory
UI	make icon of item only visible on hover
UI	add inventory full feedback at 1st level
UI	visualize slots more significant
UI	add audio feedback for cannot pick-up, equip, select, drop, etc.
UI	provide better options for slot sorting
UI	offer fast access on the last selected item, universal quick-draw

Table F.6: Session 3 – Heuristics Evaluation Results

Subject/ Question	**1**	**2**	**3**	**4**	**6**	**Avg**	**Median**	**Std. dev**
A	*4*	*4*	*5*	*4*	*3*	*4*	*4*	*0,6*
A1	5	4	4	3	2	3,6	4	1,0
A2	5	3	5	3	1	3,4	3	1,5
A3	4	5	2	4	4	3,8	4	1,0
A4	4	4	4	5	4	4,2	4	0,4
A5	5	5	2	5	4	4,2	5	1,2
A6	2	4	4	3	2	3	3	0,9
Avg. A1-A6	*4,2*	*4,2*	*3,5*	*3,8*	*2,8*	*3,7*	*4*	*0,5*
Median A1-A6	*5*	*4*	*4*	*4*	*3*	*3,8*	*4*	*0,5*
B	*2*	*3*	*4*	*3*	*2*	*2,8*	*3*	*0,7*
B1	2	2	4	3	1	2,4	2	1,0
B2	3	5	3	1	3	3	3	1,3
B3	1	2	5	3	1	2,4	2	1,5
Avg. B1-B3	2,0	3,0	4,0	2,3	1,7	2,6	2	0,8
Median B1-B3	2	2	4	3	1	2,4	2	1,0
C	3	4	2	5	4	3,6	4	1,0
C1	5	5	2	5	5	4,4	5	1,2
C2	5	5	4	5	4	4,6	5	0,5
C3	3	4	2	4	5	3,6	4	1,0
C4	5	5	2	5	4	4,2	5	1,2
C5	4	4	2	5	4	3,8	4	1,0
C6	5	5	3	4	3	4	4	0,9
C7	5	1	5	1	1	2,6	1	2,0
C8	5	4	2	5	5	4,2	5	1,2
Avg. C1-C8	4,6	4,1	2,8	4,3	3,9	3,9	4	0,6
Median C1-C8	5	5	2	5	4	4,1	5	1,1
D	4	3	3	4	3	3,4	3	0,5
D1	3	3	3	3	4	3,2	3	0,4
D2	4	2	2	4	4	3,2	4	1,0
D3	4	3	2	5	3	3,4	3	1,0
D4	5	5	2	4	3	3,8	4	1,2
D5	5	5	1	5	1	3,4	5	2,0
D6	3	3	2	4	5	3,4	3	1,0
D7	5	2	3	4	3	3,4	3	1,0
D8	2	4	3	5	1	3	3	1,4
Avg. D1-D8	3,9	3,4	2,3	4,3	3,0	3,4	3	0,7
Median D1-D8	5	4	2	5	3	3,5	4	0,9


Heuristic Evaluations Rating
P1 P2 P3 P4 P5
5
4
3
2
1
0
A A1 A2 A3 A4 A5 A6 Avg. A1-A6 Median A1-A6 B B1 B2 B3 Avg. B1-B3 Median B1-B3 C C1 C2 C3 C4 C5 C6 C7 C8 Avg. C1-C8 Median C1-C8 D D1 D2 D3 D4 D5 D6 D7 D8 Avg. D1-D8 Median D1-D8


Heuristics Rating Median/Average
Avg Median
5
4
3
2
1
0
A A1 A2 A3 A4 A5 A6 Avg. A1-A6 Median A1-A6 B B1 B2 B3 Avg. B1-B3 Median B1-B3 C C1 C2 C3 C4 C5 C6 C7 C8 Avg. C1-C8 Median C1-C8 D D1 D2 D3 D4 D5 D6 D7 D8 Avg. D1-D8 Median D1-D8