# VIDEO GAME DESIGN PROCESS

CASE STUDY OF CHALLENGERS OF KHALEA

Eero Antero Lamminmäki

University of Tampere
Faculty of Communication Sciences
Human-Technology Interaction
Master's Thesis
March 2017

UNIVERSITY OF TAMPERE, Faculty of Communication Sciences
Human-Technology Interaction
LAMMINMÄKI, EERO: Video Game Development Process, Case Study of Challengers of Khalea
Master's Thesis, 95 pages
March 2017

---

The gaming industry has continued to grow at a large rate in the last ten years and the amount of games we get to digital stores like Steam daily is probably few times bigger than what it was a decade ago. The increasing amount of games that are on the market and the increasing demand for video games has created more requirements for understanding how video games are made and how we can make them better. Understanding the process of video game development is one way to better answer these questions and even though the process has been brought forward by game professionals and industry workers, the process still requires more research due to its erratic nature that comes from the changing industry and demands set by technology and games themselves.

In this thesis, I will examine the game development process and there are two major questions that I seek answers to. Firstly, what kind of process video game development is and secondly what problems can we find when comparing the case study process to the process explained in the literature.

The results of this study showed that video game development is a very extensive process varying from conceptualization to engine selection, resourcing, prototyping, testing and iterating. The literature also showed that by combining the information and professional talks, the process can be understood and defined better from a company's perspective. The results also show that video game development has a lot of different areas of problems that arise due to multiple professional fields colliding together. They also showed that despite the problems, lot of companies are able to overcome them and still achieve great results.

Keywords: Video games, games, development, process, game industry, design, art, iteration, prototypes

# Foreword

The process of completing this thesis was faster than what I expected that I could achieve, but over the years I have come to understand better how much I can achieve if I truly put my mind into it. The last two years have been extraordinary in a way that just a few years ago, I was taking my first steps into game development and I have not regretted a moment. Continuing in the game industry has taught me many things, what I did wrong and how to further avoid those mistakes and to better understand them. This thesis was first and foremost a learning experience for me in order to better understand the game development process, its difficulties and how to be a better game designer and to understand production and the people who work in the gaming industry. It was secondly providing something for others, which they can hopefully use to better understand this process and hopefully also create a spark to examine the literature even further, and finally if I dream hard enough, to make someone better at their work. This thesis has been a great journey and another step to working in the gaming industry.

I want to thank Dreamloop Games for first of all giving me an opportunity to work for them as a game designer and secondly allowing me to write this thesis. Everyone whom I interviewed took their time to sit down with me and talk about their work and even when the time was at the essence they did not fall back on their promises. I would have wanted to have even more interviews, but unfortunately master's thesis does have its own restrictions.

I also want to thank my supervisor Markku Turunen who took his time to read my early texts and provided me with feedback that has helped me greatly to enhance my work, and Jaakko Hakulinen for also providing me valuable feedback. Finally, I want to thank Nancy Smith who worked with me when I was studying in Canada. She was the first person ever to tell me that I could be anything I wanted to be, making me believe in myself and my abilities for the first time.

*"I am no longer afraid of becoming lost, because the journey back always reveals something new, and that is ultimately good for the artist."* – Billy Joel

Tampereella 11.3.2017

Eero Lamminmäki

# Contents

# 1. Introduction

The gaming industry has continued to grow at a large rate in the last ten years and the amount of games we get to digital stores like Steam daily is probably few times bigger than what it was a decade ago. The increasing amount of games that are on the market and the increasing demand for video games has created more requirements for understanding how video games are made and how we can make them better. Understanding the process of video game development is one way to better answer these questions and even though the process has been brought forward by game professionals and industry workers, the process still requires more research due to its erratic nature that comes from the changing industry and demands set by technology and games themselves.

Terms such as pre-production, production and post-production are not video game development specific terms as they have been borrowed from the film industry. Due to film industry developing around an environment where long live action shootings would be very expensive, the industry had to learn to squeeze the process into more specific parts. Splitting the whole process into three parts was the ideal choice so the process would have a clear start, middle and the end [Sylvester 2013]. The problem with games however is that they have so many more elements into them than what the industry has, so creating a bullet proof process for the video game industry is extremely difficult. Sylvester [2013] argued that one of the reasons why game studios have different processes and there is not a standardized process is the fact that we have not yet formed a proper process that would fit most of the industry.

This brought us to the two essential questions that should be asked.

1. *What kind of process is video game development from the perspective of a company?*

2. *What problems can be found from the company's development process in comparison to the literature?*

The first question is extensive and I will mainly approach it by focusing on pre-production and production phases, but post-production is covered as well. Professional literature will be examined in order to answer this question and industry talks when proper ones are available in order to answer *what* happens in the development process of a video game.

Second question requires answers to the first question in order to do comparison between the company's process and the literature. It is important to answer *how* these processes differ and if possible to also answer *why* they are different.

By providing answers to these questions, it is possible to take a further step into better understanding game development process and hopefully with the increased knowledge we will be able to adapt better to the challenges of the process and be able to create better games in the future.

This study is structured as follows. This chapter's function is to serve as an introduction to the subject and to explain the research questions and methodologies used in the study. The second chapter will present theoretical background of the game development process provided by professional literature, Gamasutra sources and academic articles. Third chapter presents the case study itself that has been assembled from conducted interviews and company documentations. Fourth chapter discusses the differences between the case study and the theory and provides information on key differences in the process and also proposes refinements to the game development process. Finally, the fifth chapter sums up the key findings from the study, discusses potential shortcomings of the thesis and brings up thoughts and ideas for future research needs.

The methodological approach chose for this thesis is of a qualitative nature, which arose as the best method to inspect the issue due to qualitative methods being used to interpret or describe certain activities, phenomenon's or incidents. The primary sources of information were professional literature and designer talks and lastly academic articles. Due to academic articles being fairly scarce resource on game development process, they were not used as much. The professional literature also showed that despite some of it being published over ten years ago, lot of the information was still valid when looking more recent publishing's. By collecting data from multiple sources it was predicted that bigger picture of the process would most likely be found that would more clearly describe and show the process of game development.

This type of interaction is also part of hermeneutic circle, which describes that during the researching process the researcher learns more about the greater picture by learning about the details. Learning of the larger picture helps us to understand more about the specific details. The biggest effort of data collection was the data from the professional books coming as a form of text and collecting information from designer talks.

The case study itself was conducted as half-structured interviews. The reason for selecting half-structured method was because some of the material that was about to be collected had happened approximately one and a half a year ago. Due to this it was a concern that strict questionnaires and strict interviewing environment would hurt the data gathering by possibly leaving out important information. This is why the interviews were started with general topics to talk about based on the literature structuring and themes and room was given for the interviewees to speak freely about the topics and what they remembered from the process. There were a total of six interviewees: Lead designer/CEO, two technical artists, 3D artist/concept artist, programmer, and a marketer. Interviewees were selected based on their knowledge and profession of the specific issue and topic. In order to understand the early process and prototyping of the game, the two founders were interviewed together and all the other interviews were individual interviews. All interviews were recorded and then the important and relevant data was collected from these interviews.

The analysis of the interview data was qualitative and its primary focus was on finding issues and topics that could be compared with the professional literature. Additionally, information that felt surprising, the interviewees considered specific part to be important or something was mentioned in different interviews were considered important data. Great amount of the data collected was presented and written in the thesis and only a small handful was left out. This was due to some of the data being slightly repetitive or simply left out due to the restrictions of the thesis itself, such as talks about animations.

The case study itself was qualitative instrumental case study and this type of approach was selected due to its definition that is to study the case of an organization, department, occupation, person or a specific group in order to provide insight into a particular issue or in order to build theory, Durepos et al. [2010]. This kind of approach seemed to best fit with the using of the qualitative method and partially hermeneutic circle.

By collecting the data from the interviews and the professional literature and designer talks, it was the best way of creating more understanding of the whole process of game development and to better understand the big picture of the process.

The author of this thesis is a game designer intern working at Dreamloop Games since late May 2016. The author has not worked with the parts that have been mentioned in this thesis, except for providing usability feedback for different tools that are used to create some of the games content.

# 2. Game Development Process

The first section of the thesis will start by describing the video game development process as it is presented in numerous literature works and articles. The section will be divided into three main parts of the process; Pre-production, production and post-production. Due to time constraints and the fact that there is a lot of ground to cover in the development process, some areas will not be covered and other areas might only see a smaller glimpse on the matter.

Adams [2009] stages that as a result of there being so many different types of video games out there, it is practically impossible to define a simplistic single step-by-step process for development.

## 2.1. Pre-Production

First section of the development process is pre-production and it will include key areas of the early development such as conceptualizing, how ideas merge, resourcing and the role of the Game Design document (GDD).

### 2.1.1. Conceptualization

#### 2.1.1.1. Selecting a genre

The process of conceptualization according to Adams [2009] is the early part of the process where you make the decisions that you have to live with for the rest of the project. He continues that the general idea of a game concept is about asking how are you going to entertain someone through gameplay and also why you believe the player would find it a compelling experience at a deeper level.

The first major step of the conceptualization is to define the genre for the game. Vuorela [2007] explains that the relation of genre to all game types is not always clear. Genre gives the player that imaginative framework for possibly absurd actions, concluding that genre should first and foremost support empathizing. Each game has a genre even if it is sometimes hard to define what exactly it is. Adams [2009] tells that game's gameplay defines its genre and games can have very identical settings in the end and yet they can belong to different genres.

Fullerton [2014] however argues that many designers try to take a shortcut by thinking that the best way to create a game concept is to start with an existing set of mechanics or as she describes it as a "genre of play". She continues that all genres produce proven gameplay, which is what publishers want and what players say they want. Fullerton expresses that instead of picking existing set of mechanics or genre, she encourages developers to experiment with the game mechanics in order to explore new directions of play. She explains her reasoning by saying that there are many genres of play that she considers as “solved problems” and if the developer is not able to ask new questions about the genre, it might be wise to seek for a new territory for the gameplay.

As the figure 1. below shows, there are multiple different genres to choose from. However, from reading the literature I cannot but to wonder how essential the process of specifically picking a genre is in conceptualization. From what has been learned through this literature, it would be more logical to think of the gameplay first and the genre is slowly formed through the conceptualization of the gameplay and mechanics. It is not to say that genre is not an important factor, it is for publishers and it is for most of the players, but more importantly it is due to marketing reasons and not because it would be a conceptual necessity.

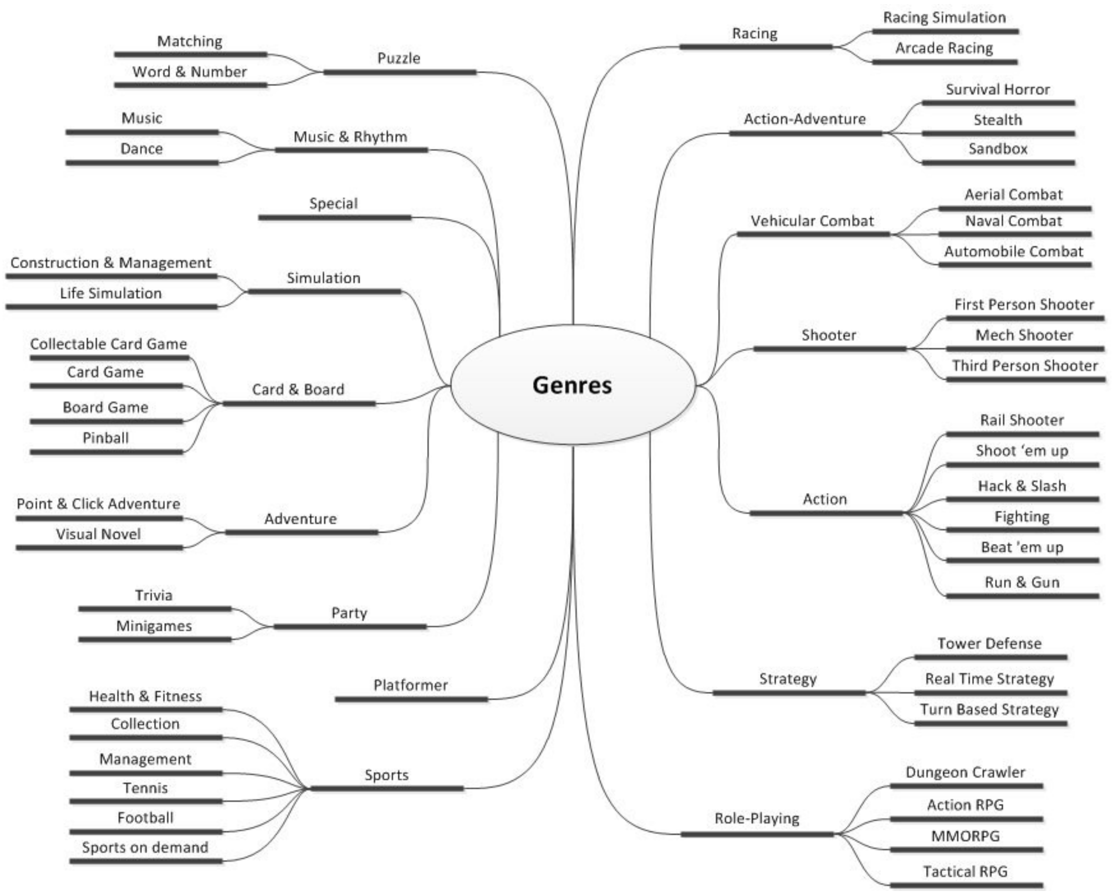


Figure 1. Genre tree [TrueAchievements]

### 2.1.1.1. Target Group

Adams [2009] starts off by saying that designers often make the mistake of thinking that all players enjoy everything the designer enjoys, which he considers dangerous hubris and tells that we make video games to entertain the audience. He continues that one of the first questions that a publisher could ask from the developer is “who will buy this game?”. Adams continues to list questions in order to define the characteristics of your players they have in common, how do they differ from other gamers, what challenges do they enjoy and what they do not. He explains that a game concept is not complete with a statement describing its intended audience.

Target groups are often first divided into three groups: casual gamers, core gamers and hard-core gamers. Adams starts by describing core gamers being ones that enjoy playing a lot of games, which is more than light entertainment for them. Core gamers spend a lot of time reading about games, being part of a community and they love to be able to complete games. He then compares core gamers to casual gamers by saying that casual gamers play for the sheer experience and if the game does not feel enjoyable for them or it becomes frustrating, they stop playing. In essence, casual gamers are not willing to spend multiple hours in order to learn complex control schemes or killed by enemies that require specific strategies to be beaten.

Wikipedia describes hard-core gamers as gamers who emphasize action, competition, complexity and also being interested in hardware and software. Lawrence [2011] argued that there was a misconception that a platform would determine a hard-core gamer, but instead said that the attitude, depth of interest in games and everything surrounding games and hours spent playing games were the core factors that would determine it.

While the three groups presented are a great start, the questions Adams wanted to ask in the first paragraph leads us to Bartle’s player types.

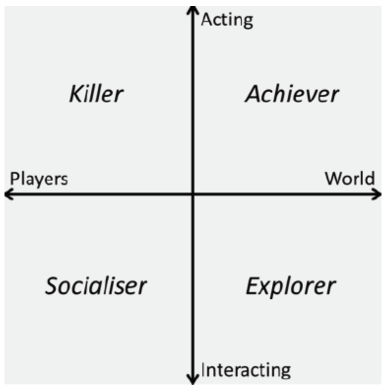


Figure 2. Bartle’s player type axes [Bartle taxonomy of player types, Wikipedia]

Hamari and Tuunanen [2014] describe that even though Bartle's player types have received some criticism for being too simplifying, they consider it to possibly be a good tool for design purposes. There are two dimensions to playing according to the figure 2. presented on the previous page. Namely action versus interaction, and player-orientation versus world-orientation. In order to define what type of player one is, he/she needs to define the position he/she is in the axes. Socialisers like to interact with players, explorers interact with the world etc.

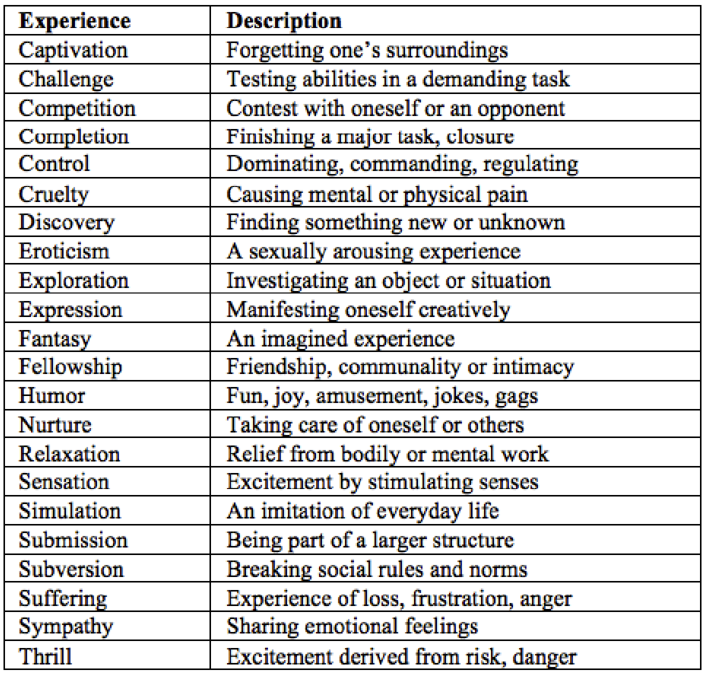
Target groups like hard-core gamers do at times have different definitions and there has been discussion on how important defining these different player types actually is. Adams brings forward the idea that defining the target audience is important mainly because then designers can ask themselves the appropriate questions regarding their target audience. If the game is meant for casual gamers, then the types and level of challenges needed for the game will change, because of how casual gamers tend to play games. Similarly, for hard-core gamers the control schemes could be more challenging and the general level of challenge.

#### 2.1.1.2. Description and the captivation

Vuorela [2007] starts first by saying that often concepts are oversized, which is because developers think that the more of captivating elements the game has, better the game will be. He continues that a game with one single great captivating element is better than ten bad ones. He urges to think one to three important elements and to design them well. He argues that often during playtesting more elements are found through player feedback. Vuorela also advices not to confuse captivating elements to features of the game as it is more important to be interested in what the player wants to get out from the game and not about how he does it.

It becomes apparent that the term Vuorela uses is a bit outdated and could be better defined with the word player experience. Fullerton [2014] describes player experience goals as goals that the designer sets for the type of experience that players should have during the game. Like Vuorela, Fullerton also advices not to confuse these to the features of the game. She continues that they are descriptions of interesting situations where the designer hopes the player will find themselves.

The theoretical framework of Playful User Experience (PLEX) according to Holopainen, Lucero et al. [2013] is a categorization of playful experiences that comes from previous theoretical work on pleasurable experiences, game experiences, emotions, elements of play and why people play. These categories are said to capture the most prominent playful features of different kinds of products that can be seen on the figure 3.



| Experience | Description |
|---|---|
| Captivation | Forgetting one's surroundings |
| Challenge | Testing abilities in a demanding task |
| Competition | Contest with oneself or an opponent |
| Completion | Finishing a major task, closure |
| Control | Dominating, commanding, regulating |
| Cruelty | Causing mental or physical pain |
| Discovery | Finding something new or unknown |
| Eroticism | A sexually arousing experience |
| Exploration | Investigating an object or situation |
| Expression | Manifesting oneself creatively |
| Fantasy | An imagined experience |
| Fellowship | Friendship, communality or intimacy |
| Humor | Fun, joy, amusement, jokes, gags |
| Nurture | Taking care of oneself or others |
| Relaxation | Relief from bodily or mental work |
| Sensation | Excitement by stimulating senses |
| Simulation | An imitation of everyday life |
| Submission | Being part of a larger structure |
| Subversion | Breaking social rules and norms |
| Suffering | Experience of loss, frustration, anger |
| Sympathy | Sharing emotional feelings |
| Thrill | Excitement derived from risk, danger |

Figure 3. PLEX framework [Korhonen et al. 2009]

Fullerton also suggests that production should not be started without a deep understanding of the player experience goals. She explains that once the production actually starts, it becomes increasingly more difficult to alter the software design.

The way the literature handles player experience has changed over the years for certain. Ernest Adams for example does not talk about player experience more than on few occasions and even then they are mentioned in user interface design chapter and in storytelling section. Fullerton's advice to understand player experience goals before production starts feels like a big part of where development can go wrong and considering how much the PLEX framework is referred in more recent articles, it becomes apparent how important part of development and conceptualization player experience is.

### 2.1.1.3. Ideation

Vuorela [2007] starts of by saying that it is much easier to talk about the creation of babies then where ideas come from. He continues that ideas can come in an airplane, diving to a deep pool, reading a book or the emotional thrill left by a great movie. He adds that the best stories can come from grief and anxiety but adds that game ideas come mostly from positive experiences, not always but he recommends to use positive experiences as a launching pad. Adams [2009] tells that you can find ideas from practically anywhere, but only if you actually look for them and he adds that creativity is an active and not a passive process. He then adds that one of the most unexpected things can become an idea for a game, like a game called Viva Piñata released for Xbox 360 by Rare in 2006 and it was essentially about breeding piñatas.

One of the most recent examples of where ideas come from, comes from a game called ”That Dragon, Cancer” developed by Ryan Green. The story of developing the game was published by Wired in 2016 in article called “A Father, a Dying son and the Quest to Make the Most Profound Videogame Ever”. In the article, Ryan talked about the process going through his son’s brain cancer that was diagnosed in 2012. Due to the constant ups and downs Green suffered because of his son’s condition, he got an idea for a game in a church. Wikipedia site of the game described that the game was about Ryan’s and his wife’s experiences with their son Joel and after their son died in 2014, Ryan and his wife did a lot of reworking on the game in order to memorialize and personalize their experience and interactions with Joel for the player. Reception of the game was excellent by most reviewers and the game was nominated for the 2016 Independent Games Festival excellence in audio and narrative awards.

Rouse [2005] says that there is no shortage of ideas in the gaming world. He continues that the challenge of game development is not about coming up with good ideas, but it is about the ability to follow through and be able to create a compelling game around that idea. Adams [2009] adds that it is a common misconception that a great game idea can make you a fortune and says that it can occur, but extremely rarely. Adams continues that computers can create almost any kind of visual experience one can imagine and even experiences that are not physically possible in the real world. Adams says that the designing of computer games begins with a question “What dream am I going to fulfill?”. He encourages to dream the dream, understand it, feel it and knowing who else can dream it and why.

### 2.1.2. Resourcing

Vuorela [2007] tells that resources are any kind of physical or mental resource that can be invested to a game making process. After the conceptualization has been done, developers should already know what resources they have available so the concept and the resources should now be made to correspond each other. Vuorela says that in an ideal case the limitations of resources have already been taken into account, but there is a chance that during conceptualization the concept will get oversized. Vuorela explains that the general rule should be that there is never right amount of resources available, so the only two options left are to reduce the concept or growing the resources.

Cohen & Bustamante [2010] describe that the producers job here is to provide sufficient information on how many team members will be needed, equipment, tools and other resources required. They continue that these are considered as below-the-line costs as in the resources that are actually needed to build the game. Any other specialists such as voiceover talent or composers are above the line costs that are typically handled by the publisher. Both say that in the end it does not matter if resources are large or small, it is about how you leverage the resources you have at hand and utilizing the team to its fullest potential. However, Cohen & Bustamante remind that too short of a production cycle can cause unavoidable damage to the project, and is something even the most efficient use of resources cannot save the game from it.

Cohen & Bustamante continue explaining that having large teams allows every discipline to be covered with full attention, since team members do not have to concern themselves with multiple duties from different disciplines. However, they also explain that when it comes to creativity, gameplay, communication and teamwork, bigger is not always better. Oversized staff can cause communication breakdowns and general lack of unity inside the team. According to Cohen & Bustamante the biggest challenge a small team will face is bandwidth. Since team members have to take additional duties then they would have in a large team, they will have to work harder with longer hours and it can cause lack of focus. Smaller teams do however have advantages as they tell. Familiarity with the team members is better and the team tends to be more united. Also, communication will be easier because fellow team members often work more closely and are not separated by a different floor or building.

Bethke [2003] adds that most game projects tend to fail to meet their financial expectations due to inability to clearly articulate what their expectations are. He tells that if one does not have enough resources it might be better to think of what is the best game you can make with the resources you have in hand. He ends the paragraph by saying that too many developers try to take the gaming world by storm by taking one ambitious step.

### 2.1.3. Selecting your engine

The topic of game engines is not always the most covered topic when it comes to game development and design literature. However, selecting the engine for the project is important phase during the pre-production and comes directly from the conceptualization.

Gamedesigning [2016] lists three of the most popular free game engines which currently are: Unreal Engine, Unity and GameMaker. Unreal Engine is developed by Epic Games and the engine has been used to developed notable games such as Mass Effect series, Bioshock, Gears of War and Batman: Arkham series. Similarly, Unity is also one of the most popular free game engines to this date if not the most popular. Unity has been used to develop many successful games over the years such as Pillar of Eternity, Blizzard's Hearthstone and many others. Also, GameMaker has proved to be a very capable engine after hits such as Hyper Light Drifter, Hotline Miami and Undertale. These three are almost always mentioned when talking about game development tools for beginners but also for more experienced developers.

Moran [2016] in his article about top five game engines talks a bit more about these engines. He talks about developers using Unity for almost any type of game and Unity is essentially a multi-platform game engine that allows the creation of both 2D and 3D games. Unreal Engine has very similar qualities and can be used for making both 2D and 3D games, but GameMaker is almost exclusively used to create 2D games rather than 3D.

Then how about creating your own engine for the project? Kissner [2015] says it surely is a possibility to create your own engine, but he advices not to if you can possibly avoid it. He continues that Unity and Unreal Engine are excellent choices and flexible enough to make pretty much any game you wanted. He then lists few reasons of why you should write your own engine: Essentially if you need certain functionalities that are not available or the solutions presented in those engines are unstable, one believes he can create a better engine and wants to stay in control of the development.

Not having certain functionalities or having unstable solutions is likely one of the bigger reasons to create own engine for the project. An Indie game studio called Redhook Studios that are known for their 2D gothic horror style game called Darkest Dungeon had an Ask Me Anything (AMA) in reddit website in 2014. Tyler Sigman who is the producer and game designer of the game was asked what the company used to build Darkest Dungeon and if they used GameMaker or Unity for prototyping. He answered that instead of using these, their technological director Kelvin McDowell was confident about creating own engine for their project. Because Darkest Dungeon does not require many of the features of Unity and because the emphasis was on the 2D art style, which at the time was problematic in Unity, the engine was created because of these major reasons. Unity essentially introduced a new 2D toolset in patch 4.3 in 2013 when Darkest Dungeon was already under development.

When it comes to comparing Unity 3D and Unreal Engine, both have their own advantages. Deetman [2014] in his articles about differences between these engines brings out one of the biggest factors, the asset store. Of course because Unity asset store has been around much longer then the Unreal store has it has become very populated with 2D and 3D assets and plugins. However, even after few years Unity store is still extremely packed and is considered more useful then Unreal store that is still lacking content. Although Deetman points out that

Unreal is less dependent on third party software fixes to their engine, while Unity has taken broader approach, allowing much more tools for developers to enable broader game making ability.

Gregory [2014] talks more about game engine architecture and talks about the game engine reusability gamut.

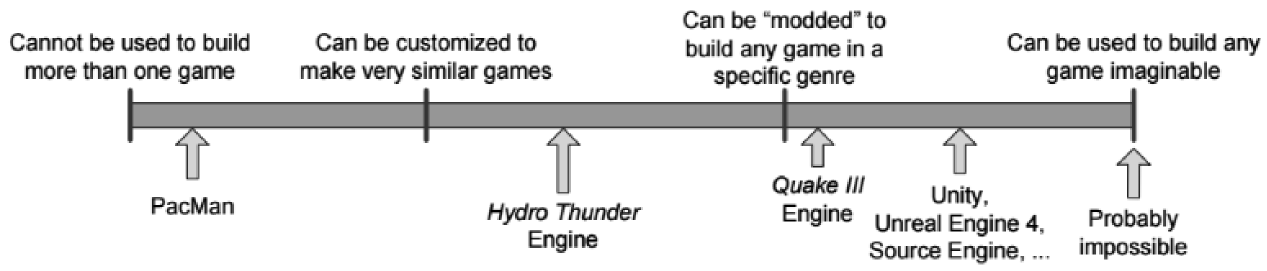

Cannot be used to build more than one game
Can be customized to make very similar games
Can be "modded" to build any game in a specific genre
Can be used to build any game imaginable
PacMan
Hydro Thunder Engine
Quake III Engine
Unity, Unreal Engine 4, Source Engine, ...
Probably impossible

Figure 4. Game Engine reusability [Jason Gregroy, 2016]

Gregory argues that the line between a game and its engine is sometimes blurry and some engines make fairly clear distinction and others make almost no attempt to separate the two from each other. He says that no studio essentially creates a perfect separation between the game and the engine, which he says is understandable considering that the definitions of the two components often shift as the game design solidifies. He then says that when a game contains hard-coded logic or game rules or has special-case code to use or render specific types of game objects, it becomes hard to reuse the software to make a different style of game. He tells that the term game engine should probably be reserved for a software that can be extended and used as a foundation for different types of games without requiring major modifications.

Referencing to the picture above, Gregory talks that one might think that game engine would be a software that could play virtually any game content that can be thought. He continues that this is something that most likely will never be achieved and that most game engines are crafted and tuned to run a specific game on a specific hardware platform. Gregory says that even the most general purpose multiplatform engines are really suitable for building games in one particular genre, like racing games or first-person shooters (FPS). He sums that the more general purpose a game engine is made less optimal it will be for running a specific game on a specific platform.

Gregory also goes through different genres and explains some of their technological requirements and sums the chapter with saying that each game genre has their own technological requirements that explains why game engines tend to differ from genre to genre. However, he also points out that as a result of there being a great deal of technological overlapping between genres, it becomes increasingly more possible to reuse the same technology across various genres.

I think the literature shows that while multiplatform game engines are being developed more and they are advancing in a way that they can be used to build and develop games from more than games from few genres, all of them still have their own falling points. Conceptualization is the phase where gameplay and the feel of the game is thought, but often the significance of the engine is not brought into the picture. Developing different types of games brings their own specifications and requirements to the table, which more or less affect the engine that would best fit that concept. Unity and Unreal Engine fit the part in the figure 4. where modifications to the engine allow to build almost any kind of game, but that is through either the asset store or developing those plugins yourself. When creating game concepts, game engines should have a small room in the process and it would be beneficial to question what engine would best fit the type of game and project that is about to be created. Unity allows a great store for purchasing or getting some plugins for free that can potentially help different projects a lot, which makes the engine quite beneficial for many types of games. However, it can also potentially create plugin dependencies that can too be questioned how good they actually are for the project.

### 2.1.4. Game Design Document

#### 2.1.4.1 What GDD is and types of documentation

Adams [2009] starts of by saying that developers and especially game designers do need documents in order to tell others about their design. He continues that the type of documentations produced varies from project to project but they do in the end follow a common thread. Adams continues that programmers often make a mistake of thinking up a game and then they immediately dive into and start programming, which he says that in commercial game development this kind of approach can be disastrous and project tend to require different degrees of formalities.

Adams tells that the key part of the documentation is to communicate the design to other members of the team. In practice, communication takes place not through the documentation but during team meetings and general conversations. He says that documentation is important, but it is more about leaving a paper trail of what has been agreed orally within the team. He says that the process of writing documents tends to turn a vague idea into an explicit plan. If a game feature is not described in writing, then there is a good chance that someone has to make quickly make it up. In a worse situation, each team member could have a different idea of what they intended to do. Rouse [2005] says that design documentation should be started as early as possible in order to create focus on the project and he adds that the concepts that are most central to the game should be the focus.

Rouse then adds that in essence focus should be something that heavily thrills you and fuels your creativity, because without these feelings it will be hard to live through the hard times during the development.

Adams [2009] describes few different types of documents a designer might want to create: the high concept, game treatment, character design, world design, and story or level progression document and game script. He reminds that it is not a specific list nor does every project have them, but feels that they are most common ones to be made. Adams describes that high concept and game treatment documents are generally for sales purposes in order to communicate the concept to possible publishers. Similarly, the game treatment document is for sales purposes as well, but the treatment document is more suitable for someone that has already interest in it and essentially wants to hear more about the game.

Character design document according to Adams is specifically about documenting the design of one character appearing in the game and its primary meaning is to communicate the character's appearance with a high amount of concept art presenting the character in different poses and a with different facial expressions. More importantly the document should include the characters moveset, as in a list of animations to indicate how the character moves. Additionally, the document should also include information about the character like values, likes and dislikes, history and other important elements.

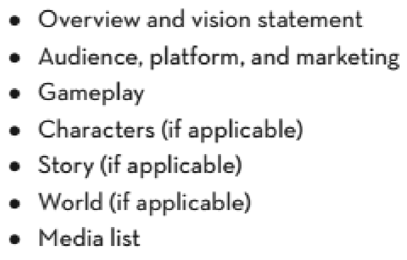
Adams continues explaining the world design document, which is the tool of portraying the game world. It is not supposed to be very specific list but rather a proper background information on what the world will contain. He continues that if you have different sceneries like landscapes or cityscapes, then the document should include a map. The document should also be about the feeling of the world, aesthetics and setting the emotional tone for the world.

Adams says that story and level progression documentation should record the large-scale story of the game, if it has one. The purpose of the document is not to outline everything that can happen inside the game, but to outline the players experience from beginning to the end. In this document, you are also supposed to indicate how the player experiences the story. As in how the story is essentially told, through cut-scenes, briefings, dialogs or through other narrative elements.

Lastly Adams talks about the game script, which in the early times of the industry was basically a big tome that incorporated every document that has been discussed in earlier paragraphs. In its essence, the game script should include the rules and the core mechanics of the game and Adams continues that the document should make it possible for you to play the game, at least in theory by understanding all the rules. Game script is not supposed to include technical design but it may include details such as target platform and minimum technical specifications. Adams mentions that if there is a technical design document, it is often based on the game script and is written by a lead programmer or technical director.

### 2.1.4.2. Inside the modern Game design document

Fullerton [2014] starts of by listing the general contents of a design document as can be seen in figure 5.



- Overview and vision statement
- Audience, platform, and marketing
- Gameplay
- Characters (if applicable)
- Story (if applicable)
- World (if applicable)
- Media list

Figure 5. List of common GDD contents [Fullerton, 2014]

Fullerton continues that because design documents tend to change continuously, everyone in the team do not have the time to read the document again every time changes are being made, so version history should be made so it becomes much easier to read for everyone. It is necessary to note that what is listed in the figure 5. might not apply to every project and game, but it is a very common list of GDD contents.

The vision statement of the game is about capturing the essence of the game to the reader in order to make it as compelling as possible. It should have a very brief logline to describe the game, synopsis of the gameplay in order to create a mental image of the user experience and how the game will play out. Additionally, Fullerton recommends briefly explaining in more detail what makes the game unique, what are the core mechanics, setting and a summary of the look and the feel of the game.

Next the audience, platform and marketing should be described, which is essentially what has already been discussed in chapter 2.1.1.1. Choosing a platform is essential and especially asking why specific platform/s were chosen in the first place. System requirements are about what kind of limits does the game have in terms of requirements, especially when it comes to target audience. Fullerton suggests to describe what is required to play the game and why those choices for the requirements were made.

In order to understand better what the project is up against, Fullerton suggests that a list of top performers and feature comparison should be in the document. She suggests that for comparing top performers sales figures, release dates and other important information on the title should be included. Additionally, she suggests comparing your game to the listed competition and thinking of why the consumer would purchase your game over the others. Additionally, she suggests providing sales expectations within key markets like Europe and United States or more country specifically.

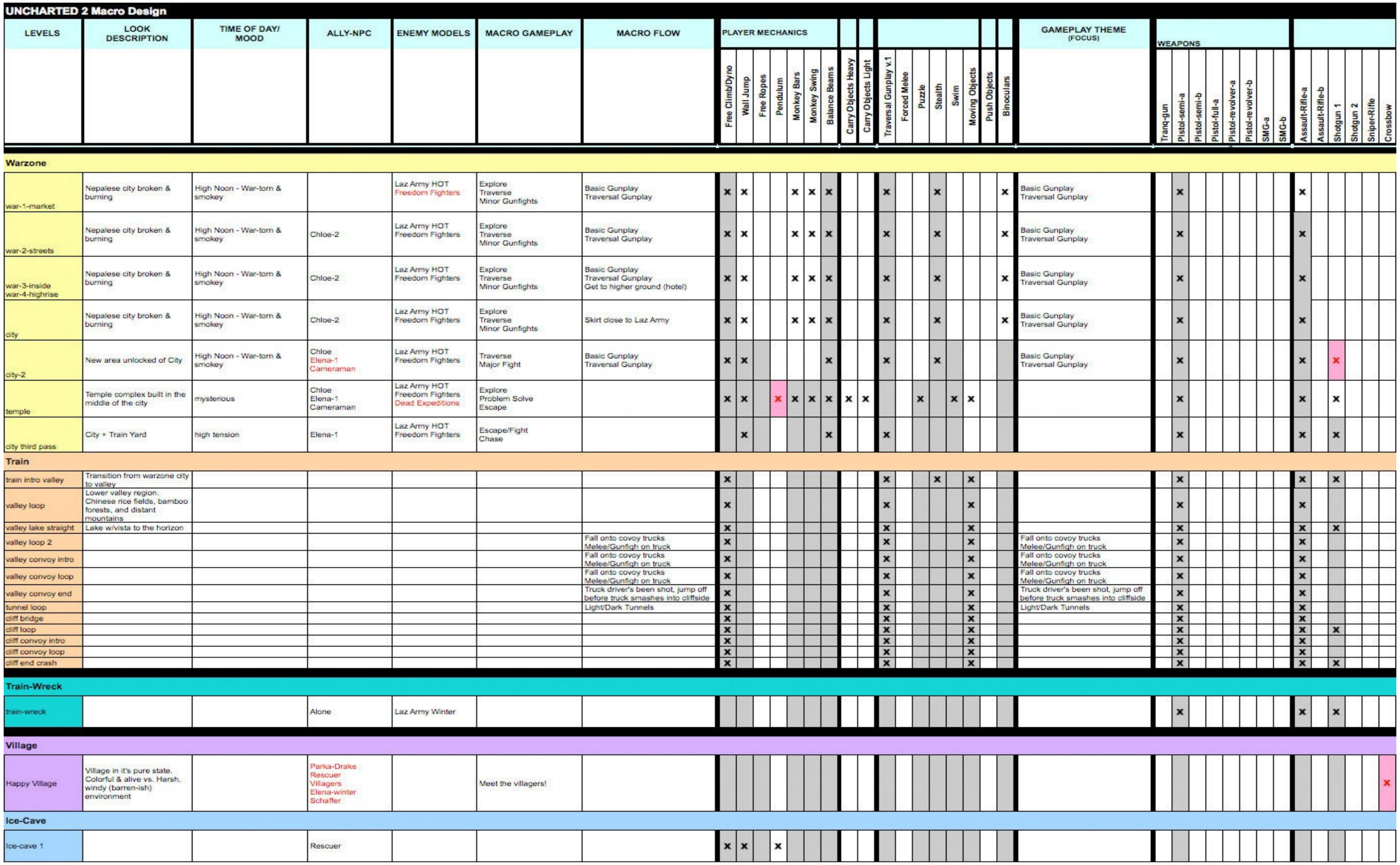
The gameplay part is where core gameplay is described and should be tied to a prototype that gives an overview on how the prototype functions according to Fullerton. Controls should be described and visualized by using control tables or flowcharts. Wireframes of interfaces should be included for artists and a description on how each interface functions and transitions between various interfaces. Fullerton continues that if a prototype has already been made, then describing the rules of the game should be an easy task that includes defining game objects, concepts and their behaviours and what their relations are. Scoring and winning conditions should also be included in the rules section and more importantly if the game includes both a single and multiplayer style gameplay.

Lastly Fullerton goes into the technical specifications or the technical document. According to Fullerton it is often a separate document like Adams also stated. Technical documentation should include information about new technology that might be used in the project and it would best to describe it in great detail. She also recommends to describe major software development tasks that could come from using a pre-existing engine that has been created for the project. In addition, risks and alternative solutions to counter these risks should be included that might come with a specific strategy being implemented. She continues to list estimated resources needed to develop the game, software tools and any kind of hardware as well. The technical documentation is as Fullerton shows it is fairly long and it is not possible to cover everything that there is to it. She also adds in the end that the list she has is a suggested list and does not necessarily need all of the components that are in it.

Fullerton [2014] agrees that the information that has been listed previously is a lot to digest for most teams, but at different points of development specific type of information will become necessary. Fullerton suggests that one should create documentation that is just enough and it is important to listen and understand team members to find out what is needed from you at any given time. She introduces a chart as a way to organize a high-level view of the game that was developed by a company called Naughty Dog for their game Uncharted 2, Among Thieves. This spreadsheet as presented in figure 6. in the next page includes all the story beats, character goals and gameplay moments the team planned to create. The information was worked out in the meetings and then minimalized into a macro design spreadsheet. Due to its conciseness, it is easy to be moved around in the spreadsheet when the design changes and progresses over time. Of course, there might be more details to them, but with this type of spreadsheet the entire team is able to think about the vision as a whole more easily.

It becomes clear that game design documentation is an important tool in pre-production, but it can also become a pain for the team if it is not constructed the right way. As Fullerton also mentioned, long documentation can become a tedious task for everyone to go through so it becomes essential to include only what is needed and essentially what is enough for different members of the team to see and understand. Sayenko [2015] says that one of the common mistakes that developers make with the GDD is that they try to dive too deep too early and into specifics. He says that writing the GDD should be done in stages and eventually the this more conceptualized document should be shifted into a suitable guideline document with the help of everyone.

Game design document in its essence is the culmination of the conceptualization done previously in pre-production. Everything from the general idea to target group, genre and gameplay, player experience and resources should be included there are they should be turned into a concise document that can provide necessary information and focus not only for the team but for a potential publisher as well. Fullerton described that over the time Game design documents have moved from these physical documents to more digitalized and concise packages. She described that especially when teams have their own website or their own wikis, GDD's are mainly implemented as separate Wikipedia pages. The design macro spreadsheet was an interesting way to solve the problem of the huge documentation load by introducing a single page that captures the essence of pre-production, but of course it cannot remove the necessity of having a specific documentation. Documentation is always needed, it creates focus and helps team members to keep track on what is being built and what the game is supposed to be and game design documentation remains to be an important part of pre-production.



**UNCHARTED 2 Macro Design**

| LEVELS | LOOK DESCRIPTION | TIME OF DAY/ MOOD | ALLY-NPC | ENEMY MODELS | MACRO GAMEPLAY | MACRO FLOW | PLAYER MECHANICS | | | | | | | | | | | | | | | | | GAMEPLAY THEME (FOCUS) | WEAPONS | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Free Climb/Dyno | Wall Jump | Free Ropes | Pendulum | Monkey Bars | Monkey Swing | Balance Beams | Carry Objects Heavy | Carry Objects Light | Traversal Gunplay v.1 | Forced Melee | Puzzle | Stealth | Swim | Moving Objects | Push Objects | Binoculars | | Tranq-gun | Pistol-semi-a | Pistol-semi-b | Pistol-full-a | Pistol-revolver-a | Pistol-revolver-b | SMG-a | SMG-b | Assault-Rifle-a | Assault-Rifle-b | Shotgun 1 | Shotgun 2 | Sniper-Rifle | Crossbow |
| **Warzone** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| war-1-market | Nepalese city broken & burning | High Noon - War-torn & smokey | | Laz Army HOT<br>Freedom Fighters | Explore<br>Traverse<br>Minor Gunfights | Basic Gunplay<br>Traversal Gunplay | x | x | | | x | x | x | | | x | | | x | | | | x | Basic Gunplay<br>Traversal Gunplay | | x | | | | | | | x | | | | | |
| war-2-streets | Nepalese city broken & burning | High Noon - War-torn & smokey | Chloe-2 | Laz Army HOT<br>Freedom Fighters | Explore<br>Traverse<br>Minor Gunfights | Basic Gunplay<br>Traversal Gunplay | x | x | | | x | x | x | | | x | | | x | | | | x | Basic Gunplay<br>Traversal Gunplay | | x | | | | | | | x | | | | | |
| war-3-inside<br>war-4-highrise | Nepalese city broken & burning | High Noon - War-torn & smokey | Chloe-2 | Laz Army HOT<br>Freedom Fighters | Explore<br>Traverse<br>Minor Gunfights | Basic Gunplay<br>Traversal Gunplay<br>Get to higher ground (hotel) | x | x | | | x | x | x | | | x | | | x | | | | x | Basic Gunplay<br>Traversal Gunplay | | x | | | | | | | x | | | | | |
| city | Nepalese city broken & burning | High Noon - War-torn & smokey | Chloe-2 | Laz Army HOT<br>Freedom Fighters | Explore<br>Traverse<br>Minor Gunfights | Skirt close to Laz Army | x | x | | | x | x | x | | | x | | | x | | | | x | Basic Gunplay<br>Traversal Gunplay | | x | | | | | | | x | | | | | |
| city-2 | New area unlocked of City | High Noon - War-torn & smokey | Chloe<br>Elena-1<br>Cameraman | Laz Army HOT<br>Freedom Fighters | Traverse<br>Major Fight | Basic Gunplay<br>Traversal Gunplay | x | x | | | | | x | | | x | | | x | | | | | Basic Gunplay<br>Traversal Gunplay | | x | | | | | | | x | | x | | | |
| temple | Temple complex built in the middle of the city | mysterious | Chloe<br>Elena-1<br>Cameraman | Laz Army HOT<br>Freedom Fighters<br>Dead Expeditions | Explore<br>Problem Solve<br>Escape | | x | x | | x | x | x | x | x | x | | | x | | x | x | | | | | x | | | | | | | x | | x | | | |
| city third pass | City + Train Yard | high tension | Elena-1 | Laz Army HOT<br>Freedom Fighters | Escape/Fight<br>Chase | | | x | | | | | x | | | x | | | | | | | | | | x | | | | | | | x | | x | | | |
| **Train** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| train intro valley | Transition from warzone city to valley | | | | | | x | | | | | | | | | x | | | x | | x | | | | | x | | | | | | | x | | x | | | |
| valley loop | Lower valley region. Chinese rice fields, bamboo forests, and distant mountains | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | | | | |
| valley lake straight | Lake w/vista to the horizon | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | x | | | |
| valley loop 2 | | | | | | Fall onto covoy trucks<br>Melee/Gunfigh on truck | x | | | | | | | | | x | | | | | x | | | Fall onto covoy trucks<br>Melee/Gunfigh on truck | | x | | | | | | | x | | | | | |
| valley convoy intro | | | | | | Fall onto covoy trucks<br>Melee/Gunfigh on truck | x | | | | | | | | | x | | | | | x | | | Fall onto covoy trucks<br>Melee/Gunfigh on truck | | x | | | | | | | x | | | | | |
| valley convoy loop | | | | | | Fall onto covoy trucks<br>Melee/Gunfigh on truck | x | | | | | | | | | x | | | | | x | | | Fall onto covoy trucks<br>Melee/Gunfigh on truck | | x | | | | | | | x | | | | | |
| valley convoy end | | | | | | Truck driver's been shot, jump off before truck smashes into cliffside | x | | | | | | | | | x | | | | | x | | | Truck driver's been shot, jump off before truck smashes into cliffside | | x | | | | | | | x | | | | | |
| tunnel loop | | | | | | Light/Dark Tunnels | x | | | | | | | | | x | | | | | x | | | Light/Dark Tunnels | | x | | | | | | | x | | | | | |
| cliff bridge | | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | | | | |
| cliff loop | | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | x | | | |
| cliff convoy intro | | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | | | | |
| cliff convoy loop | | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | | | | |
| cliff end crash | | | | | | | x | | | | | | | | | x | | | | | x | | | | | x | | | | | | | x | | x | | | |
| **Train-Wreck** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| train-wreck | | | Alone | Laz Army Winter | | | | | | | | | | | | | | | | | | | | | | x | | | | | | | x | | x | | | |
| **Village** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Happy Village | Village in it's pure state. Colorful & alive vs. Harsh, windy (barren-ish) environment | | Parka-Drake<br>Rescuer<br>Villagers<br>Elena-winter<br>Schaffer | | Meet the villagers! | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | x |
| **Ice-Cave** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Ice-cave 1 | | | Rescuer | | | | x | x | | x | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Figure 6. Macro Design chart from Uncharted 2 [Among Friends: How Naughty Dog Built Uncharted 2, Eurogamer]

### 2.1.5. What is your role?

Vuorela [2007] says that defining roles should be the last part of the pre-production. He mentions that most of the time games are not made alone and even for hobby games can include one or two more people who might for example help with the graphical style. For bigger team's however everything that is mentioned in the concept should be tasked evenly for everyone or based on their current skills. Vuorela reminds that if the resourcing part that was talked in chapter 2.1.2. has failed and members of the team have tasks that they possibly cannot complete or they are much higher of their skill level, this would be the time to make necessary changes as making those changes later during production will be much harder and should be avoided.

Fullerton [2014] says that often smaller companies are essentially a group of friends who enjoy working together and at the start of these companies, roles of these people might not be very clear. Especially for small indie companies a common phrase to hear is that everybody does everything, but as the companies or projects grow, responsibilities become clearer. Often when you look at smaller companies and different developer roles at sites like Kickstarter, it is very common to notice that each member might have from three to five different responsibilities. But as Fullerton says the set of skills person has is what ultimately decides what responsibilities the person has in the project.

The gaming industry has many types of jobs that become more and more specified the bigger the companies and projects are, which makes it not necessary to go through all of the roles but the roles that are most essential in terms of the case study section coming later in this thesis.

Game designer is the one responsible for the play experience of the game through conception to completion and it is essentially his job to ensure that the gameplay works on all levels. Since gameplay is heavily linked to how that game is programmed, visualized and supported by other elements like music, it is important to be able to work closely with other members. Fullerton reminds that companies do not always have dedicated game designers, but the title might be undertaken by other roles like programmers, artists and producers. But as Bethke [2005] says, game design can also be split to different expertise. Especially bigger companies might have different design parts split and commonly they can be split to a lead designer, mechanic/gameplay designers, level/mission designers and then story and dialogue writers.

Fullerton [2014] tells that producer is simply the project leader and is responsible for delivering the game. Scheduling, budgeting and resource allocation are few of their primary tasks. Producers are also ones that work between the development team and the publisher's producer if the project has one. Producers are often in bigger companies split into different types of producers like assistant producers, senior producers etc. where each has different tasks they need to take care of.

Fullerton describes programmers as someone who are involved in technically implementing the game. Programmers however are often also seen as engineers and programmers have multiple different titles they can essentially hold within a company and have different tasks. Depending on the project there can be separated programmers for coding tools and the engine, database programmers, net code programmers etc.

Visual artists do multiple different tasks like programmers, Fullerton tells. They can be divided into 3D artists, concept artists, animators, illustrators, interface designers and multiple other positions based on the type of project. Bethke [2005] does give a good basic understanding of what different artists do. Concept artists are responsible for using simply pen and paper or other software tools to sketch ideas about characters, game worlds, objects and different kind of real life subjects. Additionally, they also try to capture the mood and feel by doing high quality concept art that can then be turned into different types of assets by 3D artists. 3D artists generally are just that, they make assets based on reference materials, but it is not uncommon for a 3D modeller to be a concept artist as well. However, 3D modellers also need to be aware of technical facts so they can understand certain limitations created by the games technology. Technical artists are generally as someone who communicate between artists and the programmers. Essentially their job is to be able to efficiently integrate art assets into the game and to ensure that the quality is not sacrificed during the process Cohen & Bustamante [2010] describe.

Quality assurance or QA is about finding the bugs according to Fullerton. She reminds us that QA testers can be good friends for the designer in terms of finding out the problems before the game ships out. Rogers (2010) points out that testers need to be patient, persistent and have great communication skills to report back any problems that the game might have. He also says that even though QA is not the most glamorous job in the industry, it is very essential part of the development and without it games would have broken balancing, horrible combat systems and would constantly crash. Also, he tells that publishers do have certain standards of quality so that when the game is bought by a consumer, it should be mostly bug free.

### 2.1.6. Prototyping

Fullerton [2014] describes that prototyping is a very essential part of good game design and it allows to create a working model of the conceptualized game and allows you test it feasibility easily. Game prototypes are playable, but the early prototypes only include very basic features, artwork, sound and effects. There is no need to create quality assets, because the purpose is just to focus on the game mechanics and features. When making a prototype, developers should not concern themselves with perfecting it but to worry about the fundamentals of mechanics, which will eventually tell you how solid the design actually is, Fullerton explains.

Fullerton explains that there are multiple ways of doing prototyping from physical to visual prototypes, software prototypes etc. She explains that it is important to remember that using different methods is about formalizing those ideas in order to isolate issues and learn about possible problems. Paper prototypes according to her are one of the best ways to prototype. Paper prototyping is beneficial because it allows designers to focus on the gameplay and not on the technology and Fullerton explains that programmers often will jump into coding very quickly and then making changes to the gameplay becomes more time consuming and problematic. On paper however, any kind of iteration can be quickly made.

Fullerton says that people might argue that paper prototyping does not represent the player experience like it is on computer. She says that physical prototyping allows to build the structure of the game allowing to think the interaction and formulate ideas on how the game will function and the feeling of moving in a 3D space is only one part of the game experience. At least physical prototyping will force you to think the design elements and define them, Fullerton says. Schell [2014] also reminds us that developers should not get too attached to the prototypes. The truth is that most likely many different prototypes will be made and it is problematic if you get attached to them. Schell suggests that developer should have the mindset that everything is temporary and all that matter is answering the important questions.

*“You must learn how to cut up your babies.” – Nicole Epps*

Cohen & Bustamante [2010] explain that there are three types of prototypes; first playable, white box and vertical slice. First playable is practically that, a first version of the game that has basic interaction and nothing else. White box prototype is what allows the team to iterate on the game design and on sometimes on the target platform. Here designers and programmers can test their ideas and to see if the results pay up. Cohen & Bustamante explain that white boxed levels are not just for character movement and actions but for level design as well, allowing the designers to move objects and change the world the way it better suits their goals.

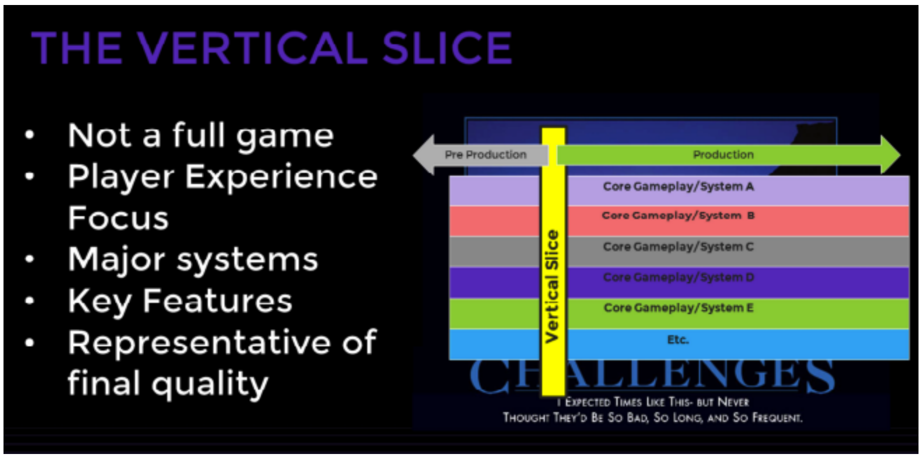
Macklin and Sharp [2016] say that the important part of prototyping is to consider what kind of questions in terms of game design need to be answered. Such questions can be if the primary activity is enjoyable, do players understand the theme of the game, is the colour palette working stylistically etc. Both suggest that prototypes should be created fast in order to avoid attaching to them because the quicker prototypes can be made, sooner it can be tested and changed.

Macklin and Sharp also say that prototyping is not only about testing out the design but it expands to other areas as well. While the playable prototype might only have simple placeholders for art, art can be developed in tandem to explore ideas around the visuals. Art prototypes focus more on general colour palette, typography, illustration and modelling style. Art prototypes wants to answer the question of how the game will visually feel.

Vertical slice is a prototype that is much more polished and is supposed to represent the final game. In 2014 Volition had a lecture at GDC where their senior producer Greg Donovan talked about some of the challenges they had to face in making Saints Row series. The core of the lecture was about their learning process in designing and creating the games and one of the major parts he talked about was Vertical Slicing (VS). This term tends to go with other names as well, sometimes being First playable, sometimes proof of concept or sometimes Minimum viable product.

Figure 7. provided in the lecture explains what the vertical slice is about. Vertical slice is a solid prototype representative of the final product, which is not a full game, it focuses on the player experience, has all the major systems built in it and features. Emphasis there lies on the fact that the art assets need to be quality assets and representative of the final quality as well. One of the key questions that Volition wanted answers to in terms of

layer experience is that is it delivering what it is supposed to? Is it fun, is it interesting, does it work and what issues might be down the road later? These were defined by the team and not only by the players. If the team was comfortable with what they were creating, then it was a clear signal for them that they could transition from


THE VERTICAL SLICE

- Not a full game
- Player Experience Focus
- Major systems
- Key Features
- Representative of final quality

Figure 7. Vertical slice as presented by Volition [Donovan Greg, 2015]

pre-production to production. It was also to show players what kind of experience they were getting and feedback was gathered from them to create a good understanding of the product they were creating.

Another prototyping type was used by Valve that was presented by Everman [2009] in the process of creating levels for Mass Effect 2. Everman presented that their prototyping had five different phases, which one was "Orange box" phase that they learned about from a Game Developer Magazine article from 2005. Jacobson and Speyer [2005] presented that the initial gameplay prototype took place on orange maps which can be seen in figure 8.

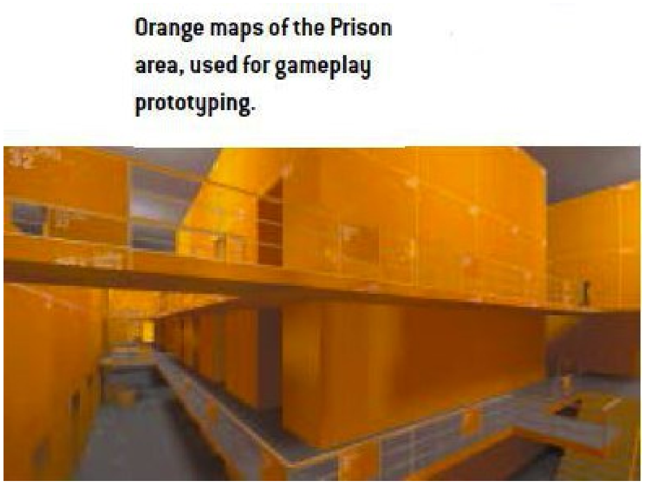

Orange maps of the Prison area, used for gameplay prototyping.

Figure 8. Orange maps for prototyping [Jacobson & Speyer, 2015]

The orange maps used an orange grid texture on walls and grey on floors and ceilings, which solved few issues early in the development. This method prevented level designers from investing to art before the core mechanics had been validated through playtesting. This heavily reduced the cost of iteration and avoided any kind of early commitment and allowed everyone in the team to focus more on critiquing the gameplay and not the art. For Everman [2009] the "Orange box" phase was about finding the fun and in order to move to the next phase, the fun had to be found. The levels also were supposed to have actual collisions after the orange box phase but geometry of the level was not textured. Additionally, the level would also support basic gameplay like seeking cover behind objects in order to fight enemies in the level.

Prototyping is a very crucial phase of the early development and it is often neglected by the developers, who have more desire to jump straight into production than fully prototyping and testing out different ideas. Prototyping is a process itself going from the first playable through white box prototype to vertical slice version and it was recommended in the literature to give time to the prototyping phase, finding out the fun and what the experience is going to be like not only for the player but also for the team. Prototyping is not only restricted to digital prototyping, but paper prototyping can be extremely helpful as well in order to find out structure and core design problems much easier then with digital prototyping.

### 2.1.6. Ending the pre-production

It becomes very clear that pre-production is a very essential process of the whole game development process. As Fullerton explained, she does not prefer to transfer to production before understanding core player experiences of the game you are about to build. Conceptualization can be a phase where everything goes overboard very quickly and then the game can become impossible to produce with the resources that are at hand. Pre-production is essentially about building the foundation of the game, like you would build a house and being realistic about the plan. You need a solid understanding of what kind of product you are about to create, who and what type of consumers do you want to impress and what sort of experience is the game supposed to create for them.

The experience and concept also need to be documented efficiently in order to communicate the ideas to different personnel in the team. The literature showed that the amount of documentation can be very dull to get into, meaning that the documentation should include only what needs to be known. The literature makes it seem like creating the documentation is a process of its own that carries on from the pre-production until the early parts of the production. Early documentation is required but it needs to be slowly worked out to communicate the direction, when designs change, they need to be communicated properly to other personnel. From early documentation, the process slowly carries on forward to a more specific high quality documentation that is then supposed to present the games ideas and architecture to allow the whole product to be efficiently built.

Pre-production does not always necessary include working prototypes, but it is a common procedure. Fullerton advices that a working prototype should be made during pre-production and it should be properly play tested. She says that it is especially important if any kind of new technology is involved in the project and it is essential to test those ideas out in order to find possible flaws in the designs. At the end of pre-production, it is also possible to show early prototypes to possible investors like publishers in order to create interest in the product.

## 2.2. Production

After the pre-production, we will move to second development phase, production. In this chapter, we will discuss common phases of production; Moving to physical product, creating the core, iteration process and transition to post-production.

### 2.2.1. Moving from theory to a physical product

After the pre-production, it is time to take all that documentation and start moving from the concept and theory to actual physical product. Cohen & Bustamante [2010] describe that during production the team will actually start building the game based on the concept and the work during pre-production. Fullerton [2014] describes production to be the longest and also most expensive stage of development where the vision of the game is supposed to be executed. While there might already be a prototype of the game, prototypes as described by Vuorela [2007] are often just a bare bone representation of the game with basic functionalities, graphics working under the game engine so during production this prototype can be used as a foundation to start actual building.

As Vuorela states, production will always need a solid plan to be executed by a producer. Production plan is basically a list of tasks meant for different personnel and then a realistic schedule. Cohen & Bustamante [2010] say that scheduling can be one of the most daunting and also complicated tasks during development. They describe that if tasks are incorrectly scheduled it can create problems in the entire production pipeline, so when scheduling some padding and wiggling room should be allowed for different tasks to be moved around if needed. Heavy scheduling is essentially a tight schedule with no wiggling room, which according to Cohen & Bustamante can be very overwhelming and difficult to accomplish. They remind that scheduling is important and without a detailed one projects can break down into chaos.

### 2.2.2 The Core of Production

The main cycle of production that according to Fullerton [2014] includes the creation of the features, assets and alpha code and other parts and then the iteration process when designing. Alpha code refers to code that has all the features needed, but not everything that is planned for the final product. She describes that production is the fleshing of the project where programmers make the game function, artists create different asset files based on the concept and requirements and animations are done. Sound designers or composers create the sound effects and the music for the game and writers write the dialogue and other in-game texts that are required. Additionally, Quality Assurance engineers keep a close look at the project, doing light testing and testing out early builds. Fullerton states that keeping all different personnel up to schedule and in sync can be a challenge and it is producers job to make sure that the communication flow remains ongoing and that the vision of the game remains consistent throughout the whole production.

As the production goes further and further, more levels and different environments, sounds and other assets are being poured into the project creating a more distinguishable form for the game. Fullerton describes that the goal of the production is to get to alpha code, which means that all features are successfully completed and no new features will be added. Sometimes this can mean that some features have to be cut in order to better meet the deadline or due to the feedback received through playtesting. Once the team achieves alpha code, it is then time to move to the post-production. However, production still has few topics that need to be covered and that are essential in the production phase; iteration cycle and design methodologies.

#### 2.2.2.1. Iteration

Iteration in game development essentially means playtesting and iteration in itself is a cycle that includes testing, analysing and then refining as can be seen in figure 9. Fullerton [2014] interviewed Eric Zimmerman about the iterative process and Zimmerman talks that your game should be played throughout the development by the development team and by other play testers. You should ask them lot of questions, observe them and then adjust the design accordingly and then playtest it again. Zimmerman reminds that it is very common for a designer to perfectly think and create a document where the designer thinks every little aspect of the game in detail so that there would be no questions needed in production. Without exception, he says that the final game and design does not resemble the original master plan that was created. He says that more iterative design process will result in a more successful product.

Rouse [2005] reminds us that during the production multiple questions will arise on different issues and this is because design document will not and cannot cover everything to make the game playable. He says that more questions will always arise on how to make the game essentially better and how to implement specific features. Rouse states that focus is very essential when these questions arise and he wants you to think if what is being implemented will be beneficial for the game in terms of focus or will it be a distraction for the project? Rouse says that players will essentially notice games that lack focus and will cast them aside quickly and play those that do not lack that focus. Rouse continues that goals of your game will most likely need to change during production and there are variety of reasons for this. He says that during the development you might find more compelling experiences that the game could provide that is however outside the scope of the original idea. While during pre-production these changes are fairly small and easy to make, during production however making changes can become painful and then the development team should think if making these changes are possible in terms of schedule and budget.

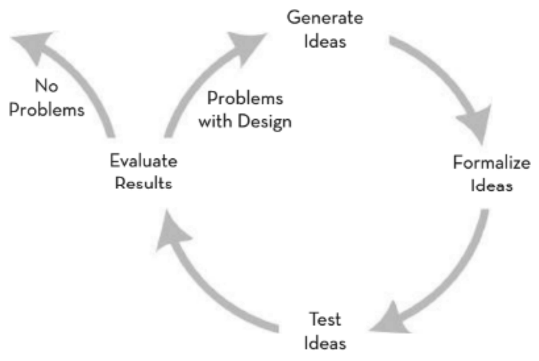


Figure 9. Iterative cycle [Fullerton, 2014]

The general attitudes towards iteration among developers according to Kultima [2015] is that developers consider iteration as both essential and natural and important part of game development. In her findings Kultima had combined attitudes towards different parts of iteration by developers that showed that iteration for some is a very natural process and for some it is very enjoyable. One of the interviewees expressed that nothing tends to go right the first time so iteration is always needed and other expressed that constant iteration helps in to make the game better. One interviewee said that when trying something new everything tends to go to hell the first few times and often it is the fifth time that essentially captures what is being created. The findings in Kultima's paper also pointed out that interviewees thought that some extra should always be added to the schedule for iteration purposes for the later parts in the production. However, some interviewees expressed that iteration should have more room in the pre-production while some thought that it would be for production, so the opinions here were split.

The process of iteration is interesting in game development and considered by everyone a very essential part of the process. However, iteration is not restricted to only specific features but it is a part of multiple areas in game development. Iteration in art is an important part and as explained by Augusto Quijano in his 2014 Game Developers Conference presentation. He tells that sometimes the art process can be very simplified going directly through few steps and then it is essentially ready, but in their game project called Guacamelee it was not that simple. Quijano explains first the general influences on how the art started to form and the colour style of the art and the whole direction of the art. The figure 10. below was one of the clicking moments for Quijano he said and through that picture they went back to the main playable character.

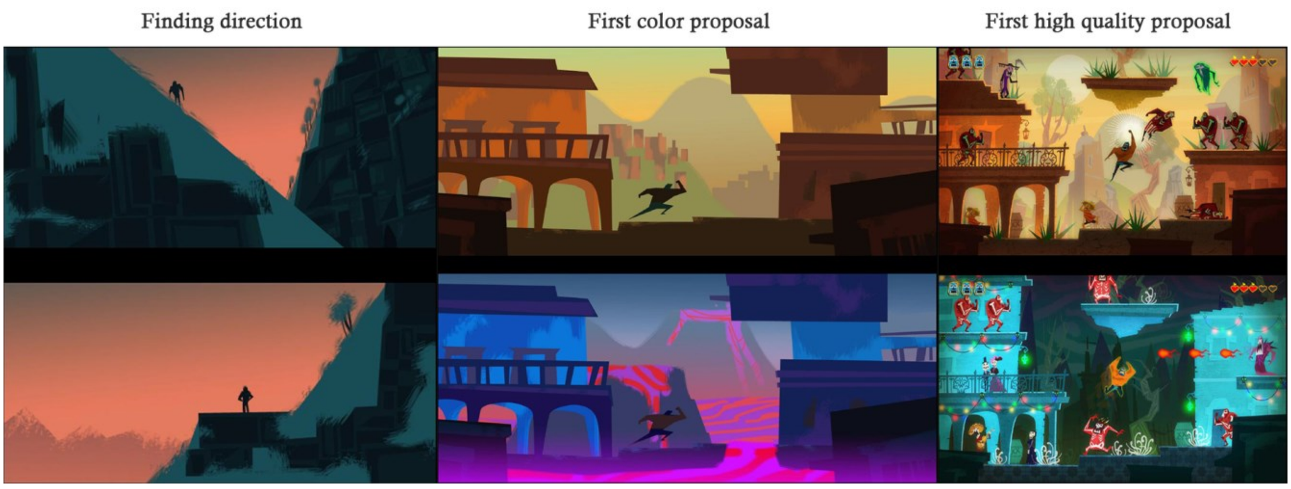


Figure 10. Finding the focus in Guacamelee [Quijano Augusto, 2014]

What made this picture special was the fact that it was the first one that presented edginess instead of following rounded up emphasis on everything that had been drawn before, which created the main focus for the art style of the game. The art style started to get more shape by trying out different colours and doing concepts.

The first picture in the figure above was the first colour proposal that also reflected the gameplay that was then turned into more detailed pictures of what the game would actually look like. Then the last picture showed one of the last sketches that essentially captured the feeling of what the game would actually look like. During early production however, the concepts that had been made had to be turned into more refined versions and the concepts of the main character were according to Quijano a hard process. He essentially received feedback from his team that the character they used in the early concepts and testing was not really what they wanted and it did not present the character best way possible like other concept they had did, which forced Quijano to think the character and iterate on the concept that can be seen in figure 11.

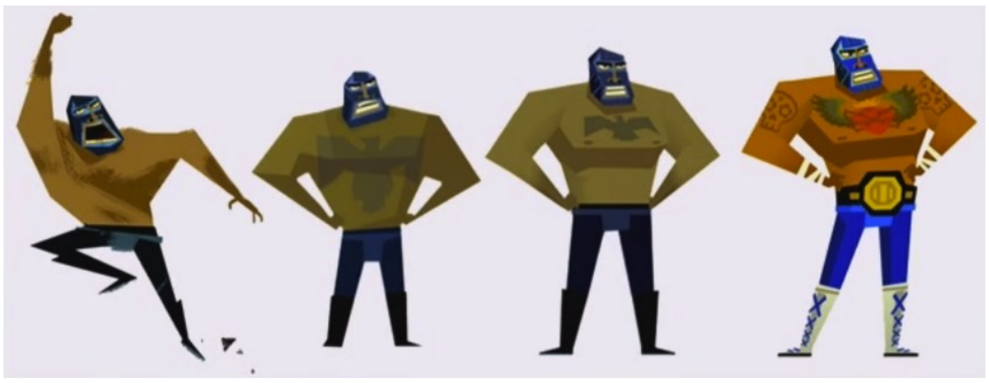


Figure 11. Iteration process of the main character [Quijano Augusto, 2014]

What the presentation of Quijano essentially captured was the fact that art had a very distinct iteration cycle of its own, slowly forming the look and the feeling of the game by testing out different things during the whole development cycle, but most of the work was done during pre-production before the team essentially agreed to make the game a reality. In the end of his lecture Quijano suggested that mistakes should be made as early as possible in the project and frequently. He recommended showing everything early and gathering feedback in order to find problems early and then fixing them.

One of the problems of iteration was mentioned by Everman [2009] in his GDC talk about Iterative Level Design process in Mass Effect 2. In his talk, he talks about the level design process and describes problems they had when developing Mass Effect 1 (ME1) that they eventually applied when doing Mass Effect 2. Everman described that “Silo Mentality” was their biggest problem when doing ME1, which meant that they would focus on different disciplines instead of the levels. He continues to explain that due to their working structure in level design, it was often assumed that pieces would automatically fit together and that deliverable assets were not always judged in the game. Due to lack of communication, iteration that was done in one department would cause a ripple effect or “iteration in other silos” as Everman presented it. Ripple effect meant a constant requirement changes that flows from one department to another and it would cause a lot of extra work for different personnel. He explains that a writer would make a change in the narrative like adding a new character, which would then ripple down to asset creation creating requirements for new geometry such as new room, doors or other assets. This would go down to scripting because cinematic creation process was affected by these changes.

Everman explains that due to the ripple effect, lot of costly and unplanned iterations were created. Cross department communication was not really encouraged because when people worked, they wanted to get everything done and they did not think that what they were doing, would also affect work in other departments. This would also create performance issues in the game due to creating something that was out of limits and made QA testing difficult and heavily delayed the content review process.

Iterative process is extremely important and considered by almost all developers a necessary process in game development. While iterative process is often seen as a more features specific process it does extend to many other areas of development like art. Iterative process is naturally infused into pre-production and production because of how even great documentation cannot provide all the necessary knowledge. Documentation will give developers focus and those details will help building the game, but there will always be questions arising throughout the production because of unseen problems and better experiences arising through the development. Iteration can also be a frustrating process because it can potentially disrupt original focus, which creates pressure to the leading elements of the team to keep everything aligned. While iterative process is needed, the Everman's talk also showed that iteration needs good communication. Iteration can create different requirements in other disciplines, so when iterating it becomes necessary to understand how something that is done can affect different areas in other fields of development.

#### 2.2.2.2. Playtesting

Even though playtesting is an internal part of iteration, here we want to talk about playtesting as a separate part because the iterative process presented in the last chapter was more development team focused iteration.

Fullerton [2014] expresses that in order to keep the focus during the long process of game development, continual iterative process is considered necessary in order to keep the focus. Fullerton describes a scenario where developer might say that testing is an expensive process and would not it be better to wait until beta version is available and conducting testing then. Fullerton heavily disagrees with this type of scenario and says that by the time that happens it can already be too late to make those necessary changes to the game. She advices, that playtesting should be started the moment the development process begins and by playtesting throughout the development it can make the game to live up to its potential. Schell [2008] admits that he hates playtesting as part of the process but recognizes that it is a necessary process in development. In essence he is afraid that people do not like his game and the rejection from people is what terrifies him. He encourages developers to get hold of that fear and conducting the playtesting despite the fears, because it can immensely help the process.

Fullerton says that playtesting is one of the most important parts of game development designer engages in and she also says that it is often of what designers understand least about. She says that it is very common misconception for designers to think that playtesting is just to play the game and gathering feedback. Fullerton reminds that playing the game is only one part of the process and it involves selection, recruiting, preparation, controls and analysis. She says that it is also common misconception that playtesting is about designers testing out the game themselves and talking about the games features or quality assurance team trying to find major flaws from the software, but this is not the case. Playtesting according to Fullerton, is about gaining insight on how players experience the game and this kind of playtesting can be conducted in few ways, which some are more informal, some are more qualitative and some are very quantitative.

While Fullerton advices to playtest the early prototypes through friends and family, she says that after the game becomes more playable and core rules start to set in, it might be time to let go of the people who have a personal attachment to you. Friends and family usually have emotional connections, which can and will obscure their objectivity. Most of them will be either too forgiving or too harsh and even if developers might think that they are given balanced feedback, they are still considered a small group of individuals that will not give you the broad criticism that you really need, Fullerton explains.

Fullerton feels that outsiders are a great resource in terms of playtesting. While she understands that bringing strangers to the testing can be scary and essentially means criticism, those strangers can still provide insightful feedback. This is because those strangers have nothing to gain or lose when they explain how they honestly feel about the game. Finding good testers is a process itself and it can be a good idea to recruit playetesters through local universities, colleges or other schools or facilities. Sometimes even libraries or gaming stores can provide playtesters if the invitation left is attractive enough Fullerton tells. She also advices to conduct a proper selection process after applications starts pouring in to find out which of the testers can actually provide to be useful and who might not be, which can be done by providing them questions about their motives of testing the game. Finding ideal playtesters can be hard but they can provide excellent feedback because they understand the types of games you are creating and most likely have experience playing some of the competition you are against says Fullerton. She says that diversity is great if you are able to create it within your target audience. In its essence, the testers should represent the total spectrum of the game you are creating.

One of the very recent examples of possible lack of playtesting is the case of Slain that was released in March 2016 for PC. Slain was expected by lot of fans due to its beautiful 2D pixel art style and the gothic dark themed world with metal soundtrack. However, the game turned out to be quite a disappointment for many after the launch and the general thoughts of the game were mixed saying that while the visual style and music was excellent, the controls did not work well, the game had lot of cheap deaths and was not just fun to play. In the original Steam comments (that are now unfortunately removed due to Steam's new review policy) there were few comments questioning if the developers ever did any kind of playtesting based on how the game turned out to be during the release. Developers did never answer this and the question remained, but from the reviews and player comments it could be seen that great amounts of players were not happy with the product and wished that it should have been better tested. Eventually the developers fixed some of the issues and re-released the game with name Slain: Back From Hell, which according to famous critic Jim Sterling was a redemption from the developers and heavily improved the game. Even though not everyone was still happy with how the game turned out after the re-release, the general feedback received got much more positive.

Reading both Fullerton's and Schell's attitudes towards playtesting, it becomes very clear of how necessary the process is, even though I can fully understand why playtesting can be a fearful process for developers. Large amount of developers put huge amount of effort into making their game and it can be a crushing experience to hear from playtesters that they did not like the game that you have worked so hard on. However, what the iteration showed in the last chapter was that despite working hard, mistakes happen during development, essentially what you thought as a developer could be completely wrong even though you might be fine with it as a developer. Playtesting should reveal major flaws in the experience and raise questions and solutions on how the game could become better through the testing. The discussion also showed that relying on friends and family can be fruitful in the beginning, but the longer the development goes those emotionally tied people should be cut out and new room should be preferably given to people who represent the audience you want to attract. The case of Slain game is interesting because it had many glaring problems and had lot of mixed reviews after the release and it partially proved the playtesting was more or less neglected by the developers. It proved that playtesting could have provided essential information to make the game better for the release, which then took the developers half a year or so to fix thus affecting the total sales of the game.

### 2.2.3. Development Methods

Development methods are crucial part of project success and also in game development and not only in regular software development where these development methods were first adopted and implemented to game development. Schell [2008] starts of by saying that when the software development was still fairly new in the 1960s programmers generally made their best guesses on how long everything would take and then immediately jumped to creating the code. This turned out be a gruesome approach that often led to projects going over budget. During the 1970s a new model called waterfall model was introduced that would better order the software development and create focus.

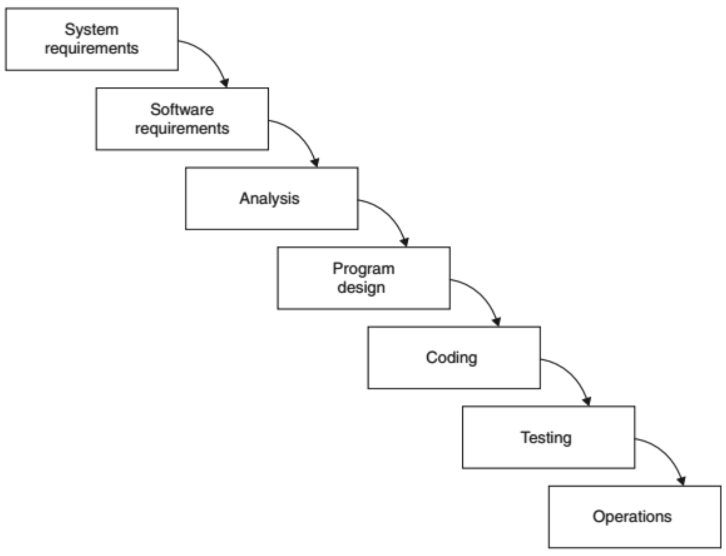


Figure 12. Waterfall model [Schell, 2008]

While the waterfall model presented in figure 12. was certainly appealing with its seven order steps, the problem with it was the it presented no iteration whatsoever as the waterfall name implies. Schell pointed out that it had one good quality however, and it was the fact that it made developers spend more time planning and designing instead of jumping straight into work without extensive documentation. Other than that, programmers found the waterfall model to be impossible because software systems are too complex for linear processes. In the 1986 a new model was presented that described more closely on how a real software development actually happened.

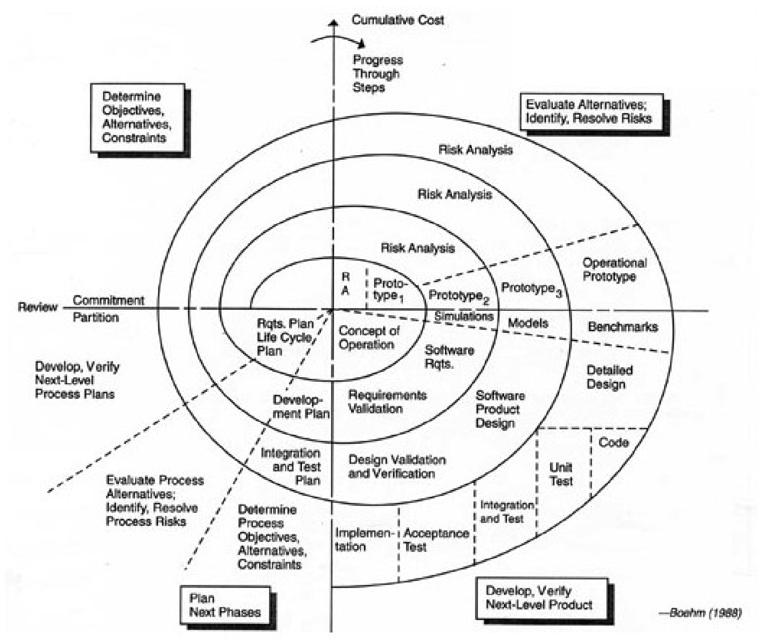


Figure 13. Spiral model of software development [Schell, 2008]

The spiral model presented in figure 13. offered three major ideas that were; risk assessment, prototypes and looping. In its essence, the spiral model suggests that you come up with a basic design and then figure out the greatest risks at your design and build a prototype to mitigate those risks. Once the prototype is ready it should be tested and the prototype should be adjusted based on what has been learned and then you loop back to figuring out new risks in the design, Schell explains.

The spiral model has been adopted in many ways and modified for different purposes in software development and slowly the methods have developed newer ones in the software development. Schell [2014] describes that the result of software engineers putting forth the values and principles of creating excellent software essentially gave birth to Agile development. Schell continues to list 12 principles that are followed in agile and there are few worth mentioning. Firstly, customer satisfaction is the key and it should be achieved by regularly delivering valuable software. Changing requirements should be taken with open arms as they are welcoming change in terms of competitive advantage. Thirdly paying high attention to technical excellence and great design enhances agility. Schell continues that while these practices have been taken and translated to different forms and names, mostly they are referred to as scrum. Schell says that Agile and scrum have had an immense impact on the software development world and especially in game development and developers have proven to be very passionate about embracing these principles. Schell adds that his observations have shown that approximately 80% of video game developers use some form of agile development.

Clinton Keith who has talked and written about game development methods and using agile says that one of the reasons why the industry has slowly started to drive towards agile development is the fact that markets force us to produce higher quality products with lower costs. He argues that the one of the crisis facing game development is the cost of creating games that grows faster than the market for games. Clinton [2010].

Clinton presents an imaginary scenario of creating a game for PlayStation 3 in two years and presents us few questions to think; he asks us if we would do anything differently if we could go back to the start of the project? Of course, you would and most likely you would not make all the same mistakes you did the first time, you would work efficiently and you would create the levels you know are fun, Clinton describes. He says that with the increased knowledge you have gained, you would be able to ship a far better game and much earlier. Clinton argues that one of the most problematic parts of waterfall approach to games is that the Big Design Up Front (BDUP) mentality is not a magic crystal ball that allows to see all the problems immediately. Eventually as we develop the game, we learn more. We learn with the controller in our hands, what looks great on the platform, how to make the game performance better and how to create better artificial intelligence (AI) enemies in order to make the game challenging. In essence, we create more knowledge every day, Clinton says.

The purpose of Agile development is to build knowledge about value, cost, and schedule and adjusting the plan to match reality. Clinton explains that finding the fun is the mantra of any iterative game development project and it can only be found when you have the controller in your hand. Waterfall projects tend to show minimal progress in finding the fun during two-thirds of the project, so often much work is spent executing a plan instead of demonstrating the value. Often it is not until the end of the project when everything starts to come together and the game is being tuned and debugged. Clinton argues that the end of the project is the worst time for this to occur. Agile practices focus the project on eliminating waste. By basically finding the fun in the game, the team finds the value of the game as soon as possible. The same principle applies to the development of assets and technology within a game. Changing assets at the end of the production cycle is also a lot more expensive than discovering the change before most of the assets are created. So, simple iteration enables game developers to explore more ideas. By delivering working software iteratively a project can prove whether an idea is viable.

The term kill-gate is often used in prototyping says Clinton [2010] and especially in mobile game development. The term refers to starting development of multiple prototypes, but only being able to fund one. Prototypes that do not have the required value are killed. PC and console port projects tend to have bigger prototypes then what mobile game prototypes are. This is why the kill-gate method is used in mobile game development, because teams are able to create working and proper prototypes in much quicker timeline in contrast to PC and console prototypes. Supercell was one of the companies saying that if they are not proud of something, the world will not see it. They also said to have abandoned 14 different projects over the years and they have a tendency to kill projects that don't meet certain requirements. The CEO of Supercell Ilkka Paanainen said that “Killing your darlings” is never easy but failures are celebrated at Supercell. Tynkkynen [2013]

Using different development methods can be a challenge because of how different teams and the projects can be and using one simple method where for example all the agile principles are adapted may often not work. The point of creating the knowledge early and adopting a mentality of finding the fun that Clinton brought up is very interesting. I would argue that often developers might be too eager to jump straight into development in order the get everything that is documented done, but the actual process of playtesting and finding the value from the game might be left adrift. I can imagine why agile is used much in game development, because it has a very customer-oriented approach that also shares marketing mindset. Hopefully developers do not be afraid of adopting more of agile development into their projects and making it their own that better suits their projects.

### 2.2.4. Ending production

Defining the end of production could sometimes be a taunting task, which is understandable as iterative process happens a lot, and focus may not always be very clear for all projects. I argue that developers can sometimes feel that their game is not ready and it needs to be iterated more and more, but the problem can become that they enter an infinite loop where ending is not really seen and the project goes heavily over budget. Reading and going through the literature has however showed that iteration is a necessary part of the development and it continues throughout the whole development. Even when we have lot of detailed documentation, we are still going to face problems during production because we are not able to foresee all those problems beforehand. Iteration becomes necessary and also playtesting so we can find those problems and fix them as early as possible, which is why bringing players in to test the game during production is also immensely important.

Production is a crucial phase in the development and it is necessary to understand two factors. Earlier in the production changes to the game are not very damaging and adopting more iteration cycle is recommended. However, once the production has reached its middle point the iteration and changing and adding new features become more difficult due to how games are structured and developed. At this point iteration becomes more loose because it cannot be adopted as much due to requirement for locking features down. Production should be planned well, but what has been presented shows that is should not be planned too tightly and room needs to be persevered for dealing with upcoming problems. Once the production is at its end, the game should have the right focus, all the necessary features locked and should be ready for final polish.

## 2.3. Post-production

In this chapter, post-production will be explained by going through important parts of the phase; testing, marketing, release and post-release details and lastly presenting project aftermaths and discussion on where projects go wrong.

### 2.3.1. What is post-production?

Post-production or sometimes also referred to as Quality Assurance (QA) and Polish phase by Fullerton [2014] is the last phase of the development process where the game is being polished and made ready for eventual release. Fullerton continues that during this phase the focus will shift from production of new code and features to making sure that what has been built during the production actually functions as intended. The alpha code is taken and slowly transferred to the final product where the user experience grows tighter, levels are polished and designers, programmers and QA engineers work on different bugs, user interface problems, control problems and game balance issues Fullerton explains. She refers to developer Steve Ackrich saying that approximately 70% of the quality of the game comes during the last 10% of development and he recommends developers to leave enough time in the schedule to accurately refine the game. Fullerton continues that during this phase the difference between rushed out game and a polished game can be enormous and the final tweaks to the game creates those breakout hits.

### 2.3.2. Testing (QA)

Bethke [2005] says that Quality assurance is a critical part of the game development and then of course the best way to test it, is to play it until it is as solid and fun as you know how to make it. He continues that the problem with this sort of method is that playing through a game by a single person would take a huge amount of time and that person would also over time developed bias towards the game and make errors during the testing. Industry general does not have one single method on how to do QA, which means that game projects and developers usually have their own QA process Bethke says. Bethke describes that smaller companies do not have a full-time QA staff as they would be only be useful under half of the project, but bigger companies tend to have full QA staffs. Fullerton [2014] explains that at the beginning QA team creates a testing plan in a form of a document that describes different areas and features of the game and different conditions under that they will be tested at. Developers produce builds that are different versions made of the game that are then tested, feedback is given and problems are fixed and then new build is produced. Companies use different software to maintain a database of the bugs in order to properly manage the huge amount of problems that arise in bigger projects. Fullerton explains that typically a console title can have several thousand bugs in the database.

Clinton [2010] argues that this kind of approach in QA and especially leaving it to post-production does not necessarily produce high quality games. He explains that the reason why he thinks so is the fact that companies tend to hire large amounts of testers for the time between the alpha and shipping date. Due to the amount of time left for the testing, companies train and introduce these personnel to the game very quickly so they get the very minimal training possible, Clinton explains. He continues that due to major defects often being rooted too deep in the design, it would be too late to address these kinds of issues, referring to for example level pacing not being fun. He also says that other problem with traditional testing is the fact that QA is often remoted responsibility and not the developers. Studios often encourage to implement features and asset creation and bugs that do not stop any kind of play progress are just considered part of the game and are generally tolerated. Implementing QA would slow this progress in production so it is often left for post-production, Clinton describes.

Clinton makes a brief introduction to agile testing and describes that is not a phase. He says that testing in agile takes place throughout the whole project and bugs are not generally tolerated. White-box testing is used in agile where testing focuses solely on single components of the game rather than the game as a whole, which requires someone who can specialize in specific areas to make this kind of process possible, Clinton says. He also argues that QA is part of everyone in the project and not only part of the specific QA team.

Developer Rami Ismail [2016] reached out in his blog post “Patch the Process” in later 2016 talking about the QA process of console games. Due to Ismail being under Non-Disclosure Agreement (NDA) he was not able to fully explain the issue, but provided interesting points about the process of QA for console games. He first states few facts saying that due to consoles being platforms and devices, they always come with certain expectations of quality and they also have systems to ensure that high quality. The console creators and the teams try to ensure that developers have a smooth process that players get a functional game, which comes in a form of “certification”. Ismail continue to explain that stores like Steam, Humble, GOG and other bigger stores essentially decided that if developers release a broken product, they will have to deal with the fallout themselves, allowing you to push a product out pretty much whenever you wish.

Consoles however still have this Seal of quality of a mindset that comes in a form of quality assurance process or certification. These certifications however do not come without their own problems Ismail explains by saying that most of the requirements that come in checklist sort of a form are reasonable, but going from a European build to for example US one does heavily alter the requirements. Some of the requirements just cause headaches Ismail explains, requiring very specific requirements for the user interface and some requirements make you desperate and angry due to poor documentation and instructions. This also makes development messy and unpredictable Ismail says.

### 2.3.3. Marketing

While Vuorela [2007] talks about marketing during the post-production phase, I think it is crucial to know that marketing should be started as soon as possible during pre-production and then continuing throughout the project. Fullerton [2014] agrees with this saying that bringing marketing team in early is a smart move, and they can be an asset for open-minded game designers.

Vuorela [2007] however does sum up the goals of marketing well by stating that developers should ask themselves few questions such as where do we find these customers and how do we raise their attention?

The era of internet has changed the approach to marketing a lot in the last 10 years and marketing is often done right after some sort of presentable content has been done, which can mean concept art pictures or prototype pictures. Finding customers has split to many different sources due to social media and probably the most important platforms of marketing are Reddit, Facebook, Twitter, YouTube and Twitch as also mentioned by Sergiu (2016), but these are mostly presented as ideas for indie games. Twitch.tv has grown as a platform immensely due to the rise of MOBA (Massive Online Battle Arena) games and the rise of Esports. It is very common to see current hits being presented in Twitch.tv with many thousand viewers coming from different streamers, which also makes it a great way to market games. Developers often send early copies for more popular Twitch.tv streamers who often also have functioning YouTube channel where to then put the material.

Presenting games in conferences and events such as PAX, Gamescom, Comic Con or for example Tokyo Game Show is also common. These types of events not only attract huge amounts of players but also lot of press personnel, so the odds of having the game covered by bigger magazines can increase. Still one of the most important parts of marketing is creating a great trailer for the game. DellaFave (2014) says that while trailers usually come later in the project, they are very important part of marketing in order to make people excited about the game.

### 2.3.4. Post-release

The work never really ends even though the game has been released. Fullerton [2014] tells that teams monitor the user feedback after the games release and fixes any glaring bugs that heavily affect the gameplay. Sometimes even more features are added to the game either as a separate patch or as a form of new content in a Downloadable content (DLC).

### 2.3.5. Post-mortems

Mayer [2016] explains that games succeed and fail and there are most likely more reasons then there are titles. He suggests to read different post-mortems through internet and says that you will learn that they will have more in common than just good or bad luck. Mistakes were made at different stages and even when things worked out they did not really work out the way they were expected. Mayer says that game development is a combination of modern technology, creative vision done in a team that is intended for public consumption. He continues that the modern game audience is very educated and more aware in the current days. They constantly look for acts of coercion and manipulation and they set a new challenge for the game development and It can only take a small mistake to transfer those devoted fans into an angry mob.

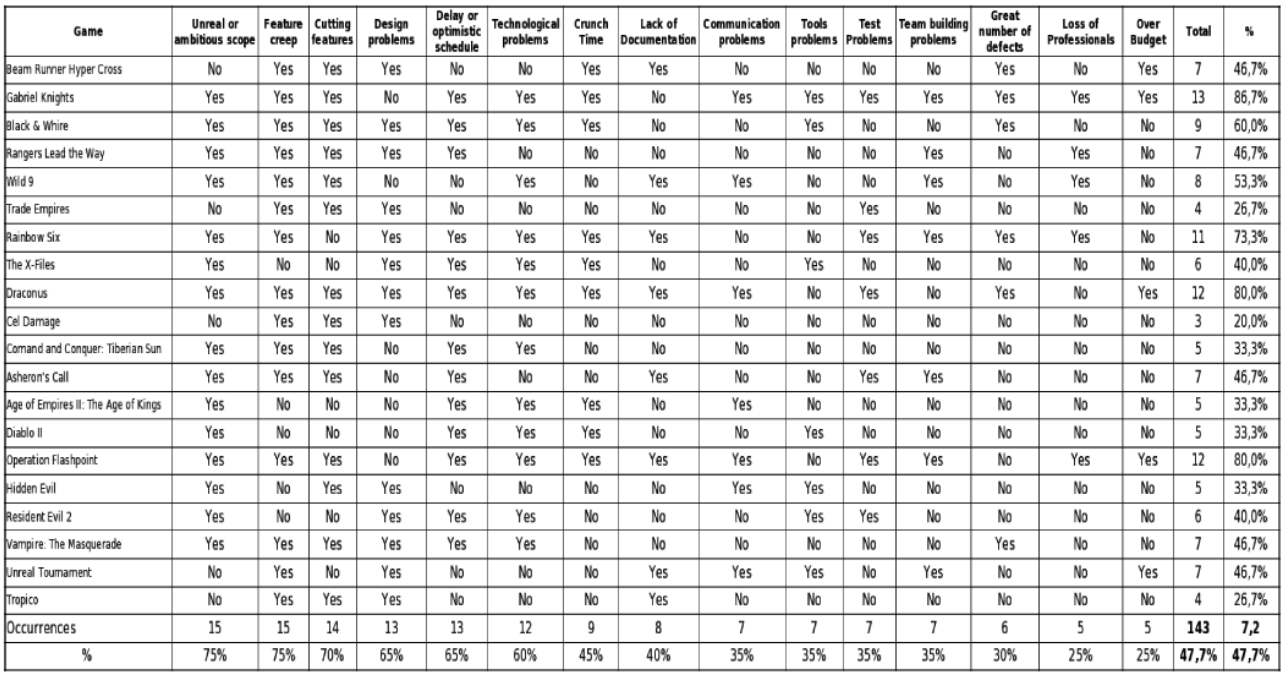
Clinton [2010] says that post-mortems tell a very familiar story saying that why do the projects start out with developers full of hope and then the projects end up in a numbing experience and overtime. Clinton says that there are three major problems in development; feature creep, overoptimistic schedules and the challenges of production.

According to Clinton feature creep is a situation where more features are added to the project after the original scope of the project has already been defined. He explains that there are two main reasons for feature creep, first is the emergent requirements that refers to stakeholders seeing the game and they start requesting developers to add more features. Second is the one where original features do not prove to be successful so more are added. Feature creep is not always necessarily bad, but if the budget remains the same and/or schedule remains unchanged, then it usually becomes a problem and Clinton tells that management tends to accept these kinds of situations easily and usually it is because in order to avoid cancellations, projects tend to agree to customer requests.

Secondly come overoptimistic schedule. Clinton says that even in our daily lives, there are dozens of reasons cropping up that can control our schedules. He says that there are multiple reasons estimated times to complete a task do not hold up. Firstly, studies have shown that productivity varies between people by a factor of ten and not everyone is capable of multitasking. Thirdly the tools and builds we use are not always stable and usable and lastly it is never really certain how many iterations it takes to turn something into fun.

Clinton starts of by saying that pre-production and production are very different, pre-production being about exploring what the game is going to be and during production everything is fleshed out by building the content. He says that that the challenge of production lies in creating predictability, maximizing efficiency and minimizing waste. Exploring what is fun and mass-producing mechanics are very difficult to schedule and when pre-production demands more time than given, projects often enter production because of the schedule demand. When production is started too soon, what ends up happening is that assets are built without proper knowledge. This leads to assets being created and based on false assumptions and requirements and when the true requirements are found, already a lot of wasted effort and time has been spent, Clinton argues.

Dietrich, Petrillo et al. [2009] created a study researching on what are common pitfalls in game development by analysing 20 game post-mortems that generally follow specific structure, that consists of things that went right and things that went wrong, and then a final message from the author.



| Game | Unreal or ambitious scope | Feature creep | Cutting features | Design problems | Delay or optimistic schedule | Technological problems | Crunch Time | Lack of Documentation | Communication problems | Tools problems | Test Problems | Team building problems | Great number of defects | Loss of Professionals | Over Budget | Total | % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Beam Runner Hyper Cross | No | Yes | Yes | Yes | No | No | Yes | Yes | No | No | No | No | Yes | No | Yes | 7 | 46,7% |
| Gabriel Knights | Yes | Yes | Yes | No | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | 13 | 86,7% |
| Black & Whire | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | Yes | No | No | Yes | No | No | 9 | 60,0% |
| Rangers Lead the Way | Yes | Yes | Yes | Yes | Yes | No | No | No | No | No | No | Yes | No | Yes | No | 7 | 46,7% |
| Wild 9 | Yes | Yes | Yes | No | No | Yes | No | Yes | Yes | No | No | Yes | No | Yes | No | 8 | 53,3% |
| Trade Empires | No | Yes | Yes | Yes | No | No | No | No | No | No | Yes | No | No | No | No | 4 | 26,7% |
| Rainbow Six | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | No | No | Yes | Yes | Yes | Yes | No | 11 | 73,3% |
| The X-Files | Yes | No | No | Yes | Yes | Yes | Yes | No | No | Yes | No | No | No | No | No | 6 | 40,0% |
| Draconus | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes | No | Yes | No | Yes | 12 | 80,0% |
| Cel Damage | No | Yes | Yes | Yes | No | No | No | No | No | No | No | No | No | No | No | 3 | 20,0% |
| Comand and Conquer: Tiberian Sun | Yes | Yes | Yes | No | Yes | Yes | No | No | No | No | No | No | No | No | No | 5 | 33,3% |
| Asheron's Call | Yes | Yes | Yes | No | Yes | No | No | Yes | No | No | Yes | Yes | No | No | No | 7 | 46,7% |
| Age of Empires II: The Age of Kings | Yes | No | No | No | Yes | Yes | Yes | No | Yes | No | No | No | No | No | No | 5 | 33,3% |
| Diablo II | Yes | No | No | No | Yes | Yes | Yes | No | No | Yes | No | No | No | No | No | 5 | 33,3% |
| Operation Flashpoint | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | No | Yes | Yes | 12 | 80,0% |
| Hidden Evil | Yes | No | Yes | Yes | No | No | No | No | Yes | Yes | No | No | No | No | No | 5 | 33,3% |
| Resident Evil 2 | Yes | No | No | Yes | Yes | Yes | No | No | No | Yes | Yes | No | No | No | No | 6 | 40,0% |
| Vampire: The Masquerade | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No | No | No | No | Yes | No | No | 7 | 46,7% |
| Unreal Tournament | No | Yes | No | Yes | No | No | No | Yes | Yes | Yes | No | Yes | No | No | Yes | 7 | 46,7% |
| Tropico | No | Yes | Yes | Yes | No | No | No | Yes | No | No | No | No | No | No | No | 4 | 26,7% |
| Occurrences | 15 | 15 | 14 | 13 | 13 | 12 | 9 | 8 | 7 | 7 | 7 | 7 | 6 | 5 | 5 | **143** | **7,2** |
| % | 75% | 75% | 70% | 65% | 65% | 60% | 45% | 40% | 35% | 35% | 35% | 35% | 30% | 25% | 25% | **47,7%** | **47,7%** |

Figure 14. Game post-mortem occurrences [Dietrich, Petrillo et al, 2009]

The figure 14. shows occurrences in different game projects and lists the most common problems as percentages below. Firstly, it should be noted that many of these games are well known and high percentage of these titles have proven to be successful. Dietrich and Petrillo show that having unrealistic scope is one of the bigger problems with feature creep included also at 75%. Following up closely came cutting features, problems in the design, delays and technological problems. Dietrich and Petrillo found it surprising result that Crunch time was not considered as a big problem in the projects. Crunch time is a standard sight in the gaming industry, which refers to people putting lot of overtime in order to meet deadlines. Interestingly also was the fact that going over budget was only considered a problem in five cases.

Game post-mortems are interesting, because they do have more in common than luck. What was shown in the study proved that having unrealistic scopes and feature creeps can cause problems in the projects and these problems occur very often in game projects. The positive thing about these studies is to notice that successful games do have their own problems and these problems were not enough to eventually destroy the whole project. The negative side of these studies is that only financially successful projects were used in the study, which does not fully make us understand what the difference between a successful project and a failed project is. It could be that they share very similar occurrences but the deciding factor might be that successful projects have the skilled individuals who are able to overcome these problems. Post-mortems do remain very interesting material to read and they do provide better understanding of what kind of issues are mostly faced in development.

# 3. Case Study, Development Process of Challengers of Khalea

This section of the thesis includes the case study of Challengers of Khalea. The case study undergoes the history of creating Khalea, and is structured similarly in comparison to the theory presented in Chapter 2. The purpose of this case study is to understand the development phases of video games through the framework of both theory, and the completion of an actual project instead of only relying on what has been written in the professional literature and what has been presented in different studies. After presenting the development process of Challengers of Khalea, we take the presented theory and create an analysis in order to more critically compare the project with the theory. A large amount of the information in this section has been gathered from company documentation such as the game design document, and from the interviews and the rest are gathered from different internet sources. Hopefully through this analysis the reader will get a much more accurate idea of what kind of process video game development is. It is intended that this analysis will also provide more viewpoints for the creator of the game (Dreamloop Games) regarding how to develop games more effectively in the future.

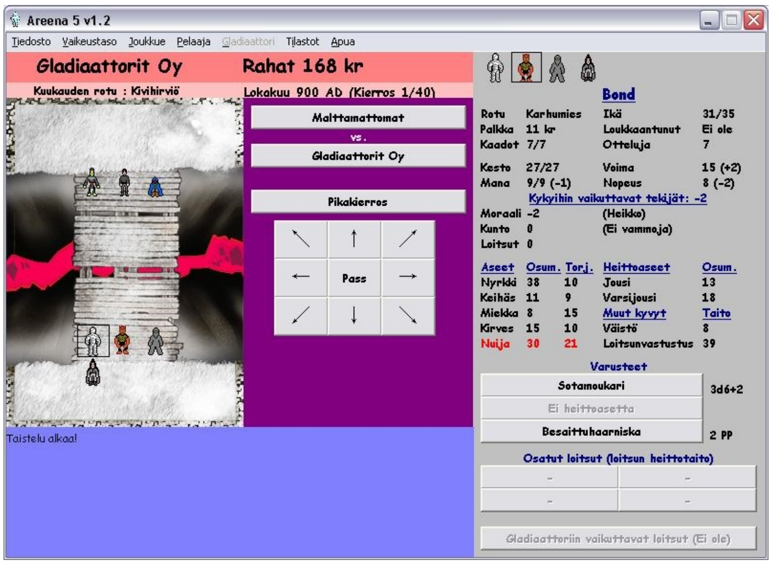
## 3.1. Project Glory

Before we can start with the current state and development of Challengers of Khalea we have to go back to 2014 when the concept and the overall idea of the game was very different of what the game currently is. Project Glory is the first version of Challengers of Khalea that started its pre-production two years ago in 2014 and we are going to start by looking at its development process first.

### 3.1.1. A Project called Glory

Project Glory, which is currently called Challengers of Khalea was the starting point of the current game. The original project started in a pitching competition in the previous company these two founders were working at in 2014, one founder being a technical artist and other one being a programmer. The pitch called Glory was highly liked and it won the competition within the company. The company however did not consider the project to be valuable enough to start its development, which essentially led to these two founders having a conversation during a gaming session and both felt that It was possible for them to start bringing the pitch to life.

One of the reasoning's that they had in starting was the fact that some of the developers at the company had seen potential in the game idea and seeing other developers at least partial approval, encouraged them to think the idea further as the idea felt too valuable to be left out. In the next six months, the ideas around the game grew heavily and new details for the game started to come out during brainstorming sessions.


Areena 5 v1.2
Tiedosto Vaikeustaso Joukkue Pelaaja Gladiaattori Tilastot Apua
Malttamattomat
vs.
Gladiaattorit Oy
Pikakierros
Pass
Sotamoukari
Ei heittoasetta
Besaittuhaarniska
Gladiaattoriin vaikuttavat loitsut (Ei ole)

Gladiaattorit Oy Rahat 168 kr

Kuukauden rotu : Kivihirviö Lokakuu 900 AD (Kierros 1/40)

Taistelu alkaa!

Bond

| | | | |
|---|---|---|---|
| Rotu | Karhumies | Ikä | 31/35 |
| Palkka | 11 kr | Loukkaantunut | Ei ole |
| Kaadot | 7/7 | Otteluja | 7 |
| Kesto | 27/27 | Voima | 15 (+2) |
| Mana | 9/9 (-1) | Nopeus | 8 (-2) |

Kykyihin vaikuttavat tekijät: -2

| | | |
|---|---|---|
| Moraali | -2 | (Heikko) |
| Kunto | 0 | (Ei vammoja) |
| Loitsut | 0 | |

| Aseet | Osum. | Torj. | Heittoaseet | Osum. |
|---|---|---|---|---|
| Nyrkki | 38 | 10 | Jousi | 13 |
| Keihäs | 11 | 9 | Varsijousi | 18 |
| Miekka | 8 | 15 | Muut kyvyt | Taito |
| Kirves | 15 | 10 | Väistö | 8 |
| Nuija | 30 | 21 | Loitsunvastustus | 39 |

Varusteet

Sotamoukari 3d6+2

Besaittuhaarniska 2 PP

Osatut loitsut (loitsun heittotaito)

Figure 15. Areena 5 gameplay [Pelihiiri.com, Areena 5]

There were two major inspirational sources for both: Areena 5 and Gladius. Areena 5 is considered a well liked classic of Finnish game development and was developed by Seppo Suorsa in the late 90's according to Wikipedia sources. Gladius came out much later in 2003 and was first released for PlayStation 2 and Xbox, and later for Nintendo GameCube.

Areena 5 presented in figure 15. is considered a classic and even though it was not graphically memorable, it had very interesting and unique management viewpoint. In Wikipedia, Areena 5 is described as a game where the player is a coach of a gladiator team. The point of the coaching is to hire gladiators and create teams that consist of different types of gladiators. Some are good with clubs, some with spears and some use swords. Each weapon type works a bit differently and the player is supposed to understand and learn how to defeat different kinds of teams he goes against. In addition, gladiators are also affected by morale, condition, different spells and fighters will also eventually retire from fighting once their age reaches a certain threshold. While Areena 5 was very popular game in Finland, it did not really capture any attention outside of the country, due to it only being released in Finnish. Current version of Areena is Areena 7, which is a web browser game and still has simplistic graphical style despite of few improvements.

Gladius was a game developed by LucasArts and was considered a successful game by different review magazines in 2003.  Gladius took the term gladiators and management into a new level.  It had a very similar idea of managing different gladiators and taking them to arenas to fight different teams, but it simplified the game in terms of only having humans, while Areena series included different races to choose from. Gladius also contained more story and background unlike Areena 5, even though in Gladius it was not well presented and it certainly did not want to focus on storytelling. Gladius was all about fighting and it also took the combat to a more interactive level. Style and interface of Gladius can be seen in figure 16.

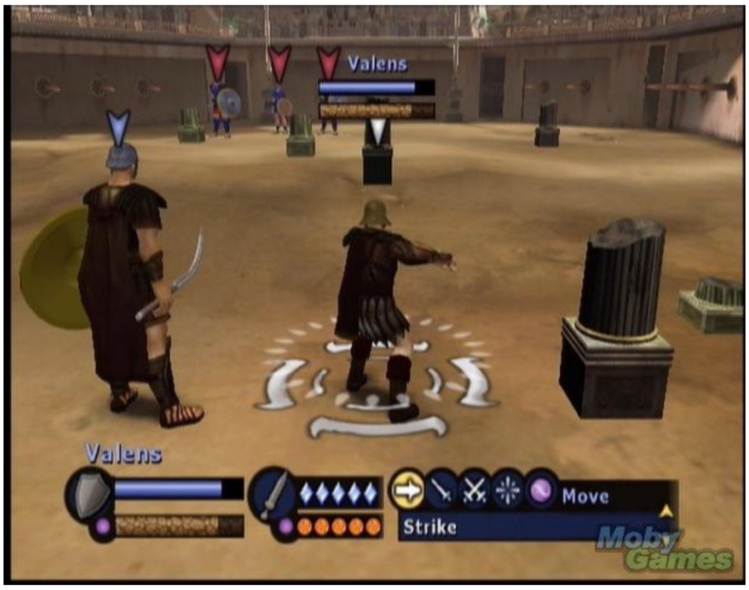


Figure 16. Gladius gameplay [Gladius Review, Destructoid]

While both games are considered to have influenced Khalea a lot, other games such as Fire Emblem and Final Fantasy Tactics series were also a high source of inspiration and they were very popular series during the 90s and in the early 2000.

The early vision of Glory for two of the founders was a game where the player has the ability to manage different types of gladiators who fight against other player’s teams. The player should be able to hire gladiators and create different contracts for them as well as manage and take care of their morale and condition between matches. Following the early conceptualization and ideas, the pre-production and prototyping started in late 2014. Dreamloop used paper prototypes to reflect the core ideas of the game mechanics and gathered results from these tests. They used two people from different sources such as friends and other acquaintances to test the prototype. The prototyping turned out to be successful in terms of results, that continued to encourage both to continue with the project. The digital prototyping continued for about six months and in 2014 Dreamloop’s Lead

Developer was also learning how to properly run a company. According to the two founders, most of the basic mechanics are more or less intact in comparison to what was tested in the paper prototypes.

At this point in the development, Dreamloop Games did not have a marketing plan. Instead they contacted a marketing and PR person who at the time was living in Ireland and he agreed to be part of the project even though there was no financial guarantee. The reason for acquiring a marketer was because Dreamloop did not want to split their focus with the development. They thought that due to the amount of work marketing includes it would have slowed down prototyping and design work.

Pre-production and prototyping continued in the late 2014. First programming parts were done by the programmer and the leader of the project, while also at the same time creating the GDD. Dreamloop explained that the early GDD of the project was pretty much what they had pitched for their previous company in 2014 and the GDD was expanded over time.  First bits of the prototype that was produced had the grid based system and movement, and it was done by using Unity engine. Grid based movement refers to a style of movement where no free movement exists but instead character's movement is based on tiles and are allowed to move certain amounts of tiles per turn as seen in figure 17.

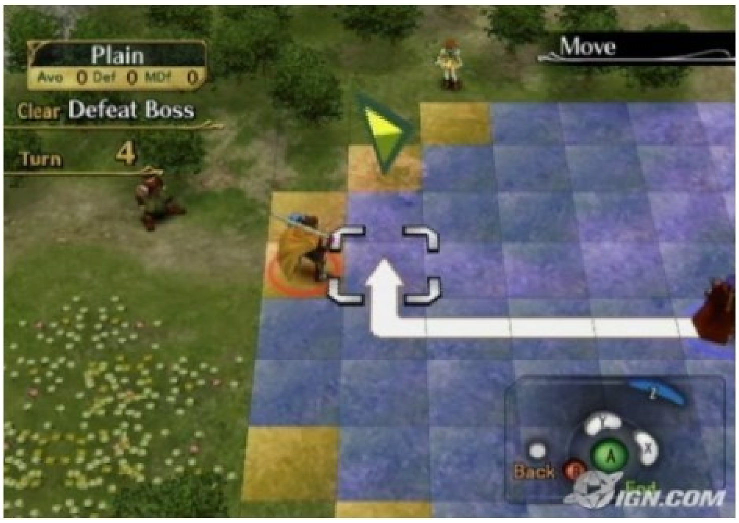


Figure 17. Fire Emblem Path of Radiance shows grid based movement [IGN Fire Emblem Path of Radiance Review]

### 3.1.1.1 - Choosing the Engine for Glory

The selection process of the Engine for Glory project proved out to be fairly simple. Because the team had extensive experience working with Unity engine through their education and professional work, it was a clear choice for both to start building the game using Unity 3D engine. Another fact was that in 2014 when they started the project, Unity 3D was the only free engine available.  It was only about a half a year later when Unreal started to have talks about Unreal Engine becoming a free engine. Due to financial constraints, it was not really possible to think about Unreal becoming Dreamloops working engine.

The team felt that Unity was a great selection to begin with due to Unity's functionalities. Unity worked nicely as a prototyping tool and it was easy to create a simple prototype for the game to test out the feature ideas they had for the game in the beginning. Team never thought of creating their own engine for the game due to lack of high level programming knowledge needed. Additionally, they felt that creating their own engine for the game would have been a financial black hole that would also require great amount of resources and future upkeep.

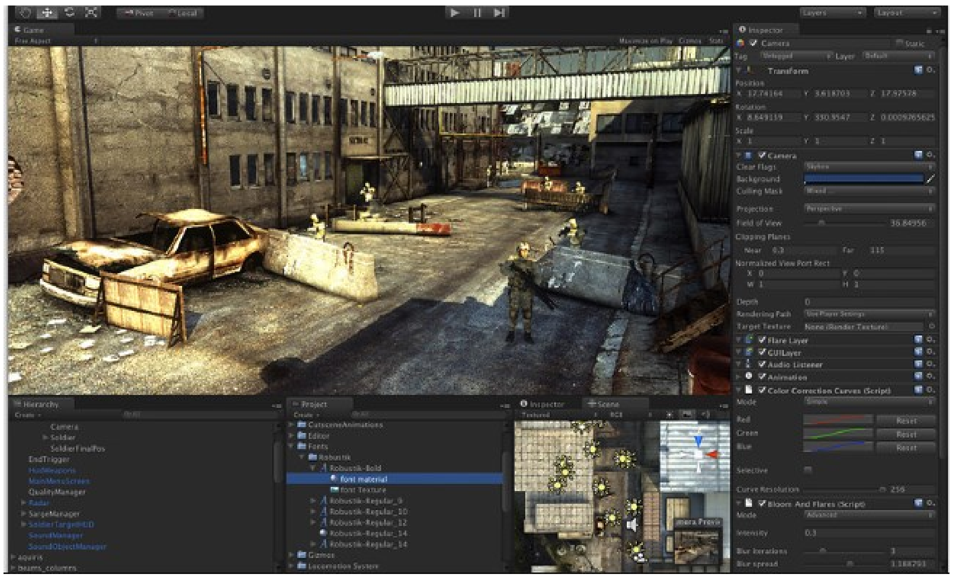


Figure 18. Editing scenes in Unity 3D [Bestwinsoft.com]

After about six months into pre-production and prototyping Unreal Engine was becoming a free engine and the team had discussion about moving from Unity to Unreal. Their interest in Unreal was mainly due to it being very high quality engine, it had great visual output possibilities and visual scripting, which also made prototyping easier. However, the team thought that the transition to Unreal would be a moderate risk for the project due to their lack of experience with the engine. It would have also taken a considerable amount of time to transition to another engine and learn how to use the new engine efficiently, so the idea was eventually denied. Additionally, Unity shop was a blooming source in 2014, providing different assets and plugins for development while Unreal's own store was lackluster due to being new.

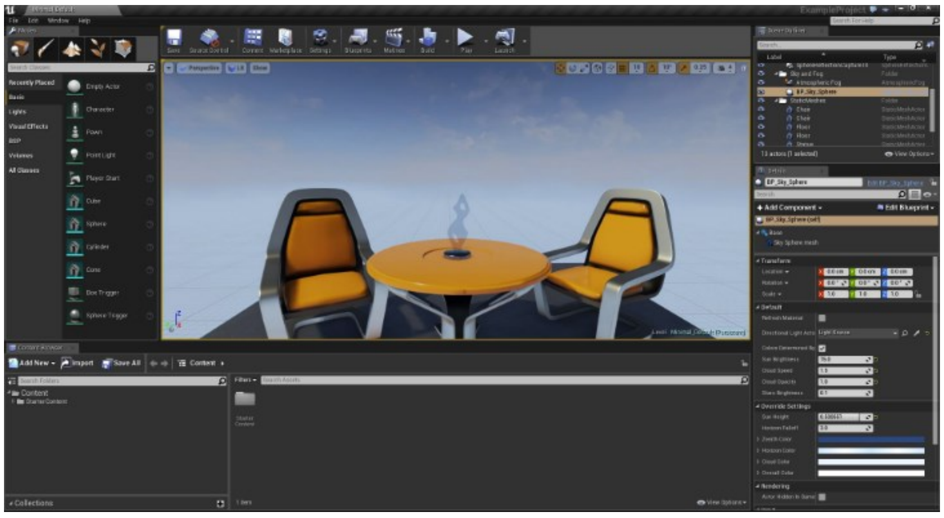


Figure 19. Unreal Engine Scene edit [Unreal Engine documentation]

### 3.1.1.2. Additions to the team

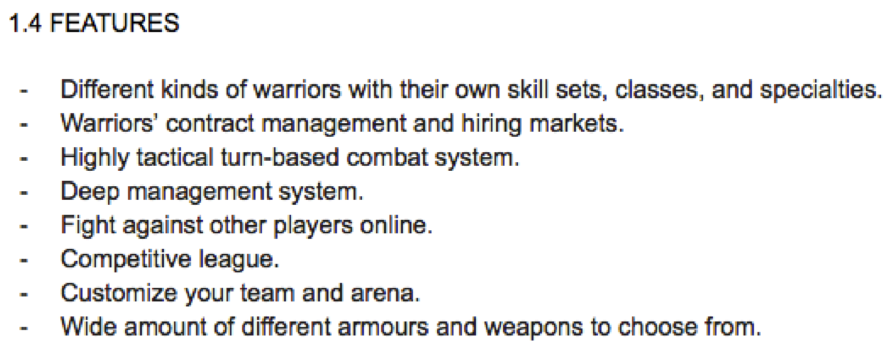
In early 2015 more programmers and artists got involved in the project and after four months, Dreamloop consisted of seven people including a lead designer, technical artist, marketer, two programmers and two artists. Due to lack of funding, Dreamloop had to apply for a loan in order for the project to continue.

Except for the marketing person, the team worked at a local startup building which offered the premises and supportive working environment for the team. The founders considered the premises essential for the project's success as otherwise communication with people working at their own homes could have been extremely problematic. As the project had just started, the actual company was formed only at the end of August 2015.

### 3.1.1.3. Glory Design Document (GDD)

During early 2015, Game Design Document for the project had grown to 59 pages and it contained almost every section mentioned in Chapter 2.1.4. The document started with an overview containing the idea, platform and a quick recap of the envisioned features included in the game as seen in figure 20. The platform focus was PC but a possible Mobile adaption was also planned.

The GDD continued by carefully listing the gameplay elements of the team by starting from the team management. The point of these parts was to essentially explain logic, reasoning, and rules of specific gameplay

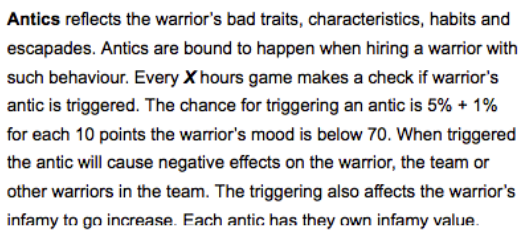


1.4 FEATURES

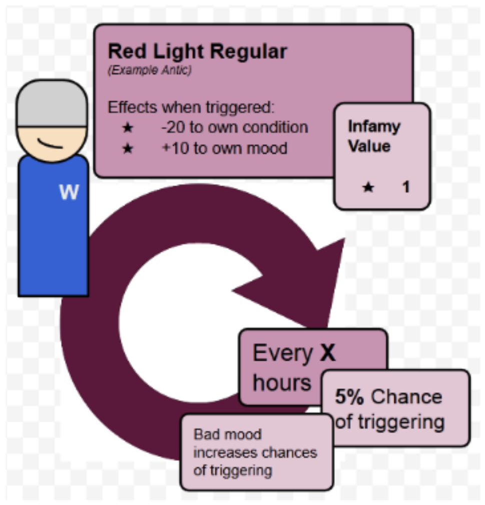
- Different kinds of warriors with their own skill sets, classes, and specialties.
- Warriors' contract management and hiring markets.
- Highly tactical turn-based combat system.
- Deep management system.
- Fight against other players online.
- Competitive league.
- Customize your team and arena.
- Wide amount of different armours and weapons to choose from.

Figure 20. Listed features in Glory GDD

parts. First it explained the team, what is a team, how it operates and where it operates. Warriors were explained in a similar way, how many can there be in a team, can they be trained, equipped and all other relevant information. The section continues to explain essentially everything that is to know about the warriors: Captain warrior, possible classes for all warriors and explanation and warrior budgets and contract offerings. In some areas, specific stats or skills are explained by using numbers instead of plain text, which aimed to deepen the logic behind the design.



**Antics** reflects the warrior's bad traits, characteristics, habits and escapades. Antics are bound to happen when hiring a warrior with such behaviour. Every ***X*** hours game makes a check if warrior's antic is triggered. The chance for triggering an antic is 5% + 1% for each 10 points the warrior's mood is below 70. When triggered the antic will cause negative effects on the warrior, the team or other warriors in the team. The triggering also affects the warrior's infamy to go increase. Each antic has they own infamy value.



Picture*: Antic system mind map with example.*

Figure 21: Antics system in the GDD

After the management side of the gameplay, the GDD continues to explain the match situation. It explains the stats of the warriors and how everything plays a role during the match and what the logic behind all the stats are. As an example, the antics stat and the system behind it is explained in figure 21. The explanation of the system and its design is clear and it demonstrates the main idea of the stat, when and how the effect is triggered and the stat the player sees essentially affects the triggering. Lastly the effect of the antics is explained and the whole system is also explained by visualizing it with the picture.

Classes are also individually explained. They contain the general picture of the class, what stats they require and rely on, what the class is good against and its weaknesses. Crossbows for example in the GDD are explained to be ranged weapons that do not require strength from the user unlike bows do, but instead they use stamina as their primary stat. They do a lot of damage, but they require time to be reloaded and reloading takes one turn unlike bows that can be used every turn. All other classes have same way of explaining them to create good vision of them working in the game.

Next the focus comes to the races of the game and some of their storylines and background information. Glory had five different races, where each has their own history, connections to other races and race specific stats. However, before the current races Zurahi, Kritan, Perkerian, Vathosi and Fjäll, there were some early iteration on the races.

Figure 22 represents early iteration of a Gladiator character that also has gladiator armor designed for it. It presents the basic form and body type plus the type of armor they will be wearing. In a similar sense, there is also an iteration of the pirate and the heavy armor type he is supposed to be wearing. Lot of these early iterations were not used, but the characters were taken into more specific detail later in the project.

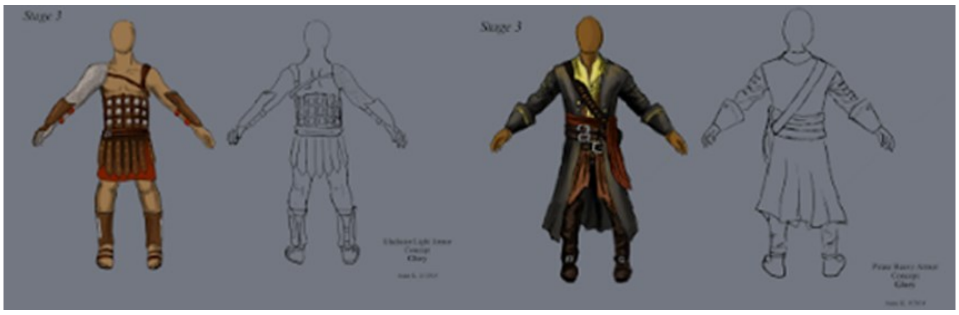


Figure 22. Early armor concepts for Glory

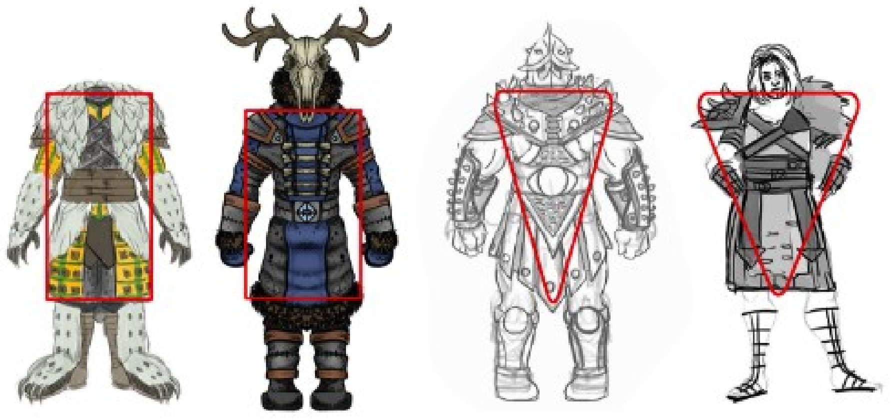


Figure 23. More advanced Armor concepts

Figure 23. presents more of what type of characters Glory had in the GDD.  They also individualized them by describing that every tribe has their own dominant shape as a starting point. For example, Perkerians are downward triangles, Fjäll are rectangles and Vathosi are upward triangles. They also stated that every tribe needs to be recognizable from afar. One armor design and related references can be seen in figure 24.

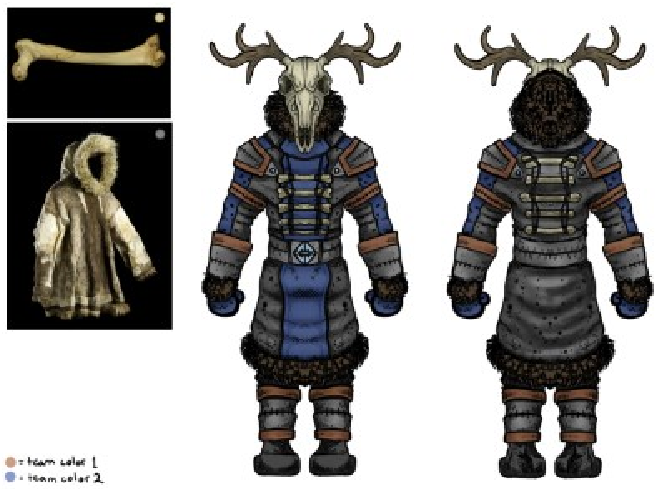


Figure 24. Armor design and references

As mentioned earlier, the GDD also had discussion about class stats.  In the figure 25. you can see the stats of Zurahi race and stat specific representation of the race that visually creates an understanding of the races strengths and weaknesses. Stamina, mobility and intelligence are marked with green, strength, condition and endurance with yellow and then the weaknesses are their initativeness (warrior’s quickness), agility and morale. After this the GDD goes more into detail about what kind of race Zurahi is, what are their connections to Vathosi for example and Zurahi's character in general.

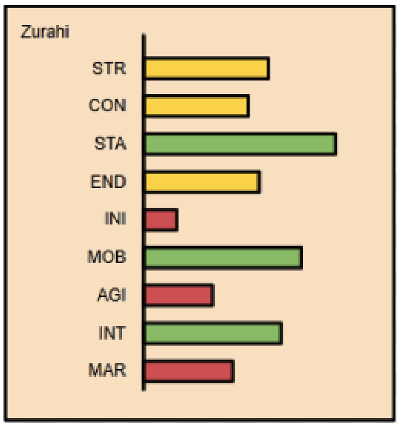


Figure 25. Zurahi stats in GDD

Lastly the GDD goes into more technical details about the art style. It describes using physically based shading with non-photorealistic features. To describe this simplistically, Physically Based Shading or PBS means modeling how light actually behaves, like reflecting from different sources such as metal or wood. Non-photo realistic features or rendering (NPR) on the other hand focuses on creating more of a drawing or painting style look for a model. The differences between the two are shown in figure 26. Then the art describes the style a little bit more by talking about how the style of the game should be more playful than realistic, but the realism should not limit armor designs in the end.

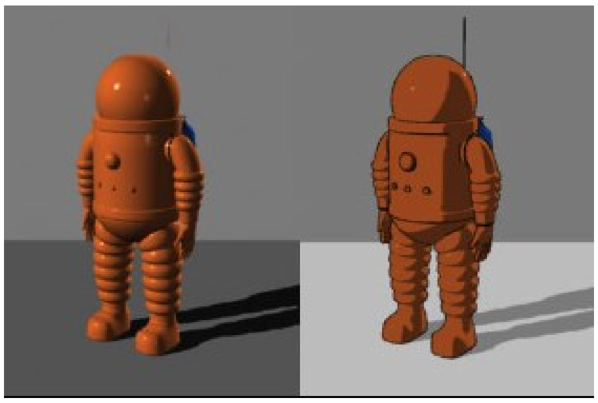


Figure 26. Photorealism versus Non-photorealistic rendering [Non-photorealistic rendering, Wikipedia]

The art section continues to describe the whole game and its style. It says that the color mood should be fresh, clean, and modern, and should not contain any overly saturated masses of color or gritty greys and browns. The picture below shows difference between real life example and then in contrast to what Blizzard created in Overwatch and mentions that it wants to draw lot of influence from Blizzard as can be seen in figure 27.

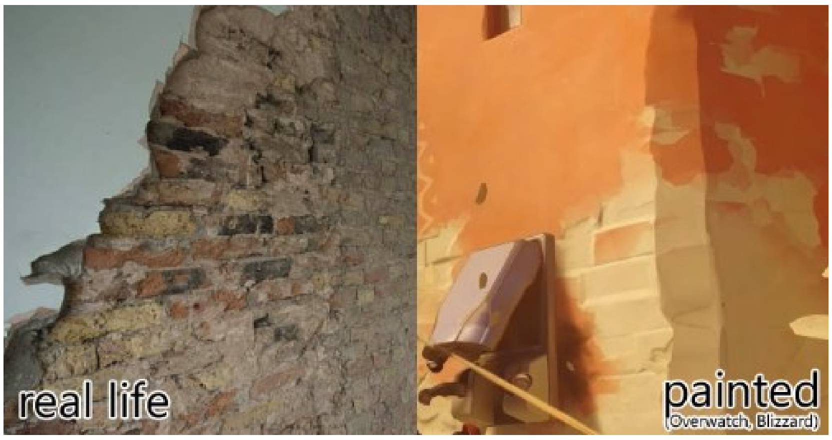


Figure 27. Real life versus painted style

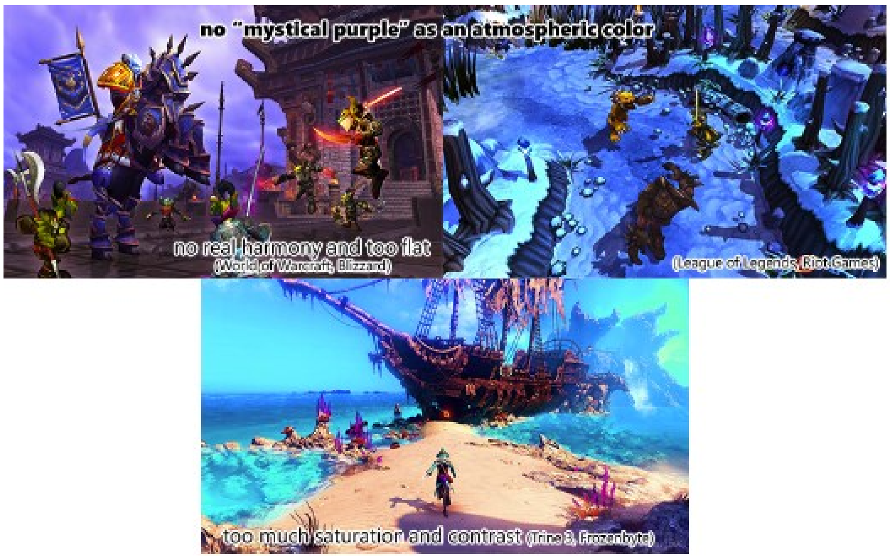

no "mystical purple" as an atmospheric color
no real harmony and too flat
(World of Warcraft, Blizzard)
(League of Legends, Riot Games)
too much saturation and contrast (Trine 3, Frozenbyte)

Figure 28. Graphical style comparison between World of Warcraft, League of Legends and Trine 3

Figure 28. shows the differences between three games, which is also used to show the oversaturation in all of the games to point out the style that Dreamloop does not want for their game. The reasoning explained is that the background should not overshadow the characters. Also, backgrounds are slightly muted to give more spotlight to the characters and the gameplay.

The rest of the document also included more information about the textures and their technical details, but because this thesis will not focus on major technical details, the section will not be covered. Most importantly, it is noted that they are included in the document, explained, and thought out.

### 3.1.1.4 - Growth of the Project in Unity

One of the first features that was done inside Unity was the grid system that was already explained earlier in chapter 3.1.1. After implementing the grid-system, more features were implemented to the game and next the prototype contained the ability for the main character to move on the gird and turn based system was implemented as well. Turn based system refers to a mechanic where each gladiator has their own turn and there is essentially no time pressure for the player to act quickly like in real time systems. These features worked as a basis for the project where additional features could be built on.

More characters were implemented to the game, which allowed that gladiator teams could be formed and tested. Warrior interaction was a major milestone for the project that included of course the ability to control your own warrior, but to also hit other warriors and kill them in order to win a matchup. Height difference was also a major feature that came in after a few months and its one of the rare features that is not often seen in tactical turn-based games. Height system is a mechanic for warriors to climb three types of ledges that can give specific advantages for the warriors. There were three different ledges designed, low, medium, and high ledges that the warrior can climb to, to seek an advantage.

After some of the core features were implemented the team had to look into creating the flight path for the arrows for crossbowmen and bowmen. They also looked more into the other classes to have more class specific features, like a spearman having the ability to attack opponents from one tile away and a similar way for crossbowmen and bowmen to be able to attack from multiple tiles away and also ignore straight lines and attack diagonally. This was the basic prototype of Glory in 2015. After the prototype was ready, they also came up with an idea of an engagement system, which would tie a warrior that has been attacked with the one warrior that is attacking. The warrior could remove the engagement, but doing so would make the warrior suffer damage as penalty. This was the last feature that was implemented to the prototype before the next one started.

During the prototyping phase, all the art were merely placeholders. Everything art related was simplistic and a placeholder that would only serve to create simplistic visuals in order to reflect some of the style and feeling. In 2015 the team was more focused on also working on the design and solidifying the design foundation of the game. They felt that creating assets was not important because if the prototype would not stick, most of the assets would have gone to waste.

#### 3.1.1.5 Project management in Glory

Dreamloop did well in terms of project management. They discussed about different issues with other teams in the startup building in 2015 and they got appraisal from other teams in terms of how well they were able to manage their project and the team. All hours and tasks at the time were logged by using basic Excel and there was always someone who was looking at the tasking and project was supervised often. The team also had frequent meetings, but they did not have separate art meetings or code meetings. Most of the programming were discussed between two programmers, but people were required to be present either in person or through Skype in order to make sure that the communication of what is being done at the moment and what will be done in the future is understood by everyone. Dreamloop implemented Lean method shortly to their project due to recommendations from other startup teams, but they felt that due to quick communication and working environment Lean felt like a rough method to be continued with. Dreamloop also used version control, which allowed them to manage their game and the changes to different files without problems.

During the fall of 2015 the Glory project came to an end due to a legal reminder from the founder's old company. Their legal notification stated that because the game with that name was pitched while both were working in the company, they would have to drop the name and create a new name for the project. This is the time when Project Glory officially ended and the Challengers of Khalea name was picked for the project.

## 3.2. The Kickstarter

After changing the name to Challengers of Khalea the developers noticed that they were in desperate need of funding. Due to lack of funding possibilities the team immediately thought of creating a Kickstarter campaign, which in 2015 was the most well-known game funding site. However, there was a major problem creating a campaign in Kickstarter. Due to Finnish laws not allowing companies to receive donations without giving compensation, it was impossible to create the campaign in Finland.

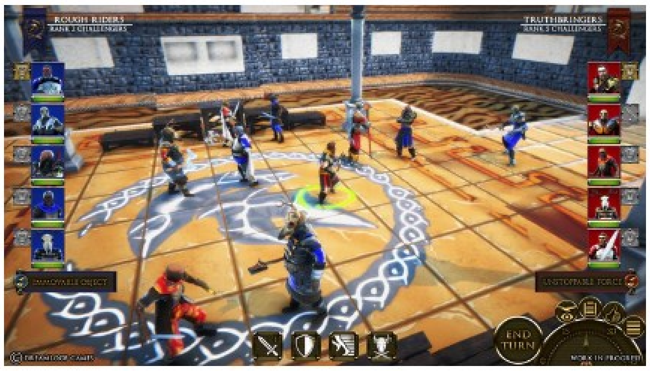
The team was able to come up with a solution after some discussions. Because their marketer was a citizen of the United States, he was legally able to create a sub-company in United States which could create a Kickstarter campaign. While it took time to meet different lawyers to set up the sub-company, the Kickstarter was finally able to start. The Challengers of Khalea campaign went through for 30 days and in the end ended up only receiving 6,611 dollars of their goal of 80,000 dollars. Due to Kickstarter's rules, if the pledged goal is not achieved, no money is received from the campaign.

The Kickstarter did not work out for the company and they had to find an alternative route to fund the project. However, what was important at this point were the reasons for the failing of the Kickstarter. Discussions with the founders revealed that they felt one of the reasons why the Kickstarter did not do well was the fact that when they had to publish their descriptive video for the Kickstarter page, the video was basically shot two weeks too early and the team was not able to put out the best quality content possible. The founders also felt that their marketer should have handled all the talking in the video, instead of splitting part of the talking to the lead designer, due to their marketer's better communication skills. It was also mentioned that the video that was put out lacked gameplay material and there should have been much more gameplay presentation in order to create a better understanding of the game to possible funders of the campaign.

The team also received a large amount of feedback of their game through Reddit and other sources. One of the biggest complaints that people had about the game was the graphical representations and the characters. People felt that the graphical style was not up to par with other similar games, which caused a lot of uncertainty. Additionally, the characters were all designed to be basic humanlike characters instead of having great amount of variation visually to create a better identity for the characters and the races. Both founders agreed that they did not tell their artists to create characters that would have a lot of variation, which turned out to be a problem in the end. Also, one of the founders who worked in the project as a technical and lead artist was not really an artist as he described himself. He felt that while he was able to program and do the technical side of the job, he was not a real artist with a proper artistic vision and partially accused himself for the problems in the art department.

Due to the feedback received the team had to start thinking about how they would continue. Lot of the feedback was concerned about the online multiplayer functionality that the game had and people questioned the necessity of the feature. After a while the team decided to take the feedback, chew it and start modifying the project to better suit the consumer needs.

About this project



- **Challengers of Khalea** is a premium **turn-based tactics** and **team management** game which blends complex **warrior contract management** from region-wide talent pools, **turn-based team combat,** and exciting, cerebral **online multiplayer.** ***It has no microtransactions, and never will.***

Figure 29. Challengers of Khalea campaign (Challengers of Khalea Kickstarter campaign page)

## 3.3. Transition from Multiplayer to single player experience

After the Kickstarter campaign was not funded, Dreamloop Games had to look into their game and think of solutions on how to continue the project. Lot of the feedback they received questioned the online multiplayer functionality saying if it would actually work and they felt single player mode was more interesting for that type of game instead of multiplayer. After a few weeks of planning the team decided to ditch the online multiplayer functionality and create a single player adventure. The team thought that the transition out from the multiplayer felt like relief, because the online programming and design over the course of the project turned out to be problematic and difficult despite the online functionalities working. Secondly, the characters needed to be redesigned and the graphical style of the game refurbished.

In order to come up with a set of ideas and core features, the team decided to take a new approach into design. Dreamloop Games took some inspiration from Volition, a company that created Saint's Row Series and split their team of nine members to three different teams. Each team had the same tasks, to design and think what would be the core elements and interesting features for the single player version of Challengers of Khalea. All groups worked with the task for two days and then presented their designs to other groups. Within a week all the new features were gathered and then the core ideas were presented to everyone. Dreamloop felt that this was a great method since getting everyone to the same table to discuss the issue would have been difficult and it would have been hard to manage the discussion. Time was the other problem and having a such a big discussion would have taken multiple days from the team. Additionally, would have been difficult to manage the discussion and make sure that everyone in the team gets heard and all the important propositions would have been given.

In smaller groups the task was easier to swallow and some of the validation was already done is those groups. Validation in this context means that all the ideas are taken, considered and the best ones are taken from inside the group and collected for a presentation. Dreamloop agreed that this method saved them a considerable amount of time, as they were essentially doing three weeks' worth of design in a single week. Dreamloop considered the method to be successful and would like to use it again in future projects.

In the end, the new single player experience was divided into four different parts. Firstly, Challengers of Khalea would get a proper story mode, meaning that the player will start the game as a captain warrior who will along the game make contracts with other warriors. Warriors that fight for you are just plainly business, you hire them, make them stronger, and then sell them with a higher price. Each playthrough is supposed to be a bit different by randomly generating warriors and teams. Game will also continue after all the matches have been won and warriors can fight through different seasons.

Secondly, the team thought about the characters. They wanted to make the characters more memorable and decided that staff members would be the storyline characters with the player's captain warrior. Staff members in Khalea are the ones that offer the player different services like medical treatment, weapon upgrades and ability management.

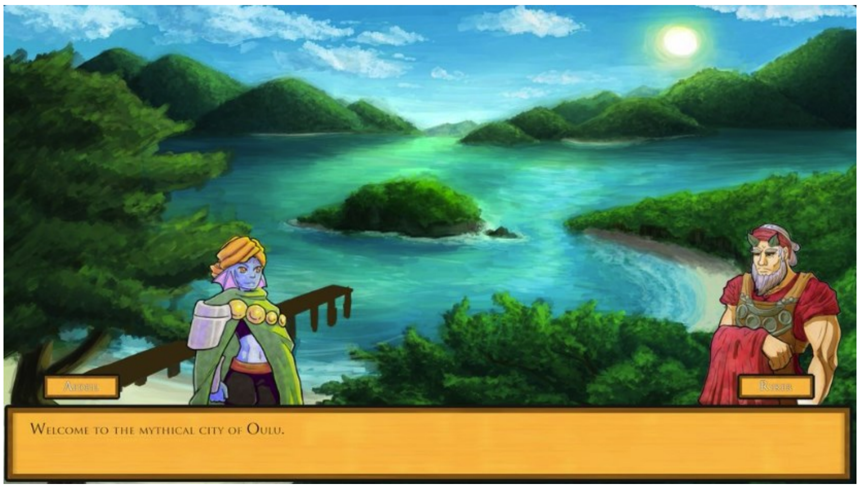

Welcome to the mythical city of Oulu.

Figure 30. 2D storytelling in Khalea

Thirdly, the team considered that presenting the story was something they struggled with and they decided to create a 2D still image presentation for the dialogues that can be seen in figure 30. The game also got a sixth race called Makapoa presented in figure 31.

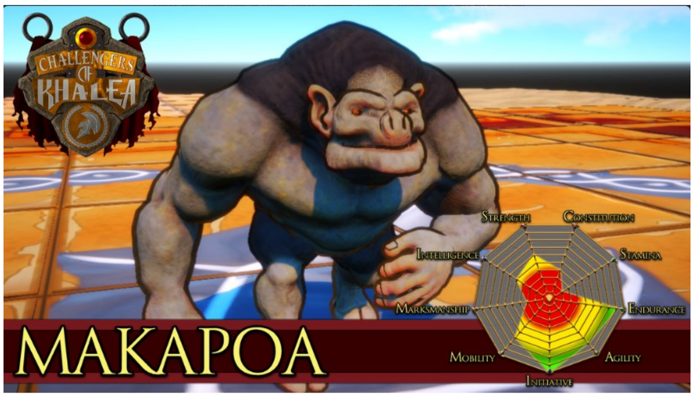


Figure 31. Early Makapoa iteration

### 3.4. Curious case of StrangeIoC

Dreamloop Games also had a problem with their programming inside of Unity. Due to the way lot of programming is often done inside Unity, the games code had an incredible high amount of dependencies. Dependencies refer to the way different parts of the code are connected to each other. Jenkov [2014] explains that essentially a class A might use another class B and if class A cannot carry out the work without class B, it creates a dependency. Jenkov explains that generally dependencies are bad because they hurt reuse. The problem with Unity dependencies was that basically each change in the code would cause a huge chain reaction due to the dependencies. Programmer would change something in the code, and suddenly they would have seven other files or sections of code that would have to be fixed as well because they were connected. This would cause lot of time consuming tasks for the programmers as they would change something or fix something in few minutes, but would then spend 30 minutes examining other parts of the code and fixing broken connections.

The lead designer admitted that it was very problematic and that there were not really many good solutions on working otherwise inside Unity. This is because of Unity engine not fully supporting the type of game they were making. Lead designer explains that Unity engine has been built by keeping eye on more general game archetypes so anything built outside of those archetypes will suffer more or less similar problems. In 2015 however they found out that there was a plugin called StrangeIoC for Unity 3D that promised to solve these types of problems. Dreamloop implemented StrangeIoC plugin to their system and started to rebuild their code base from the ground up.

Even before finding Strange the team had to go through reprogramming of the code base for two times throughout the project. Their lead designer explained that this was due to when they first started programming and prototyping, the goal was to get everything working and the proof of concept out as soon as possible. There was not really much time to fully think of the code architecture, so lot of what was written in the first two to three months was not really optimal, even though it worked. After the first rewrite the project grew and more features started to come in and the base was rewritten again due to similar issues of prototyping. He also felt that when prototyping you want to see constant advancement and if you would take more time to fully think of the code structure, it would feel very demotivating. Additionally, he felt that it is almost impossible to see the full product until you build it. Even though paper prototyping will show you a thing or two, a simple paper prototype does not necessarily show all the qualities.

Dreamloop felt that StrangeIoC was a great solution for them and basically freed them from the dependency problems. StrangeIoC removed the dependencies and allowed changes to be made easily without the fear of breaking other parts because in StrangeIoC different layers of code are treated as more of an independent black box. The structuring of Strange can be seen in figure 32.

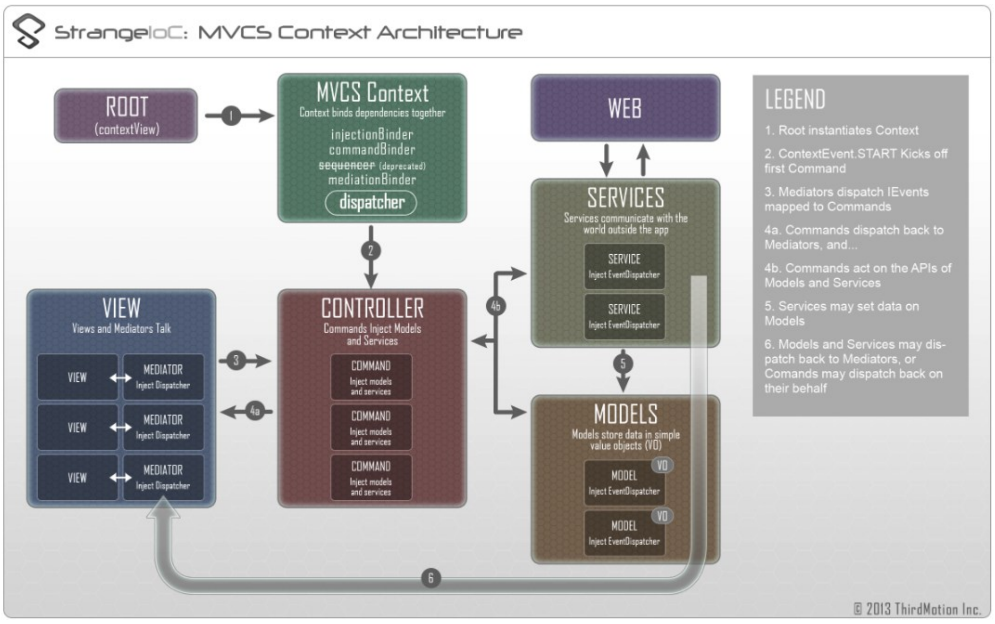

StrangeIoC: MVCS Context Architecture
ROOT
(contextView)
1
MVCS Context
Context binds dependencies together
injectionBinder
commandBinder
~~sequencer~~ (deprecated)
mediationBinder
dispatcher
2
WEB
SERVICES
Services communicate with the world outside the app
SERVICE
Inject EventDispatcher
SERVICE
Inject EventDispatcher
VIEW
Views and Mediators Talk
VIEW
MEDIATOR
Inject Dispatcher
VIEW
MEDIATOR
Inject Dispatcher
VIEW
MEDIATOR
Inject Dispatcher
3
4a
CONTROLLER
Commands Inject Models and Services
COMMAND
Inject models and services
COMMAND
Inject models and services
COMMAND
Inject models and services
4b
5
MODELS
Models store data in simple value objects (VO)
MODEL
Inject EventDispatcher
VO
MODEL
Inject EventDispatcher
VO
6
LEGEND
1. Root instantiates Context
2. ContextEvent.START Kicks off first Command
3. Mediators dispatch IEvents mapped to Commands
4a. Commands dispatch back to Mediators, and...
4b. Commands act on the APIs of Models and Services
5. Services may set data on Models
6. Models and Services may dispatch back to Mediators, or Comands may dispatch back on their behalf
© 2013 ThirdMotion Inc.

Figure 32. StrangeIoC structure [StrangeIoC website]

### 3.5. Merging with Vasara and the issues of merging teams

Dreamloop had a problem by not having the talent and resources to produce quality characters, so they decided to ask help from another team that worked in the same floor. Vasara Entertainment was a new company that had just released their first game at the end of 2015 and the team consisted of 4 people: Programmer, Technical artist, 3D modeler/animator and 3D artist/concept artist.

Approaching Vasara was an obvious choice because they were practically located on the other side of the wall from Dreamloop, and communication with a team that would be located in another city was not a great option. The team had a knowledgeable programmer and artists, and in early 2016 Dreamloop Games only had one artist at their hands and they were in desperate need of more. Dreamloop was aware that these four people had better knowledge of art, and skills to execute a better vision of characters and the graphical style. Dreamloop proposed the idea of getting help from Vasara Entertainment and the company decided to join the project and eventually merged with the company half a year later in 2016.

Even though the team from Vasara had lot of experience, the merge and getting familiar with the Challengers of Khalea project still proved to be challenging. Due to other artists not having similar skill level with the other part of the team, it took time for the art side of the project to come together and create similar understanding and guidelines inside the project for the artistic side. Art side also included animations and also the animation guidelines had to be rethought and also there the skill difference was moderate.

After discussing with Dreamloops lead designer, he thought that even to this date the merging with their artist and animator has been a rocky road and mainly due to communication issues. He thought that one of the major problems with this was the communication within the members. Often there were lot of assumptions made by the members. The other artist who they were working with should understand what they were talking about and they would understand the possible restrictions set for a specific art, but that was not always the case. Additionally, it was not always clear if skill wise everyone was on the same level and information was not always shared regards to the skill levels. Often assets came out inconsistently as it was assumed that the other half would understand the requirements. Then again, the other half did not confirm those restrictions or specific details for that art asset, which would end in assets that were not immediately usable and extra effort had to be put in. These factors came up in both interviews with the lead designer and the 3D artist/concept artist.

To this date one of the difficulties with the art side is to find an operative art leader, which according to Dreamloops lead designer has always been one of the bigger problems throughout the projects lifetime.

Currently the lead position of art has been split by two people: Technical artist and 3D/concept artist. Operative in this context means someone who knows and sets the pipeline for the art and the restrictions, and makes sure that people will follow these guidelines. The purpose of this is to create the restrictions with the set pipeline and ensure that not too many additional pipelines or ways to produce art are born.  One specific what is mentioned is the naming conventions for the art, meaning that the assets are properly named and they can be fast and easily distinguished from other assets. Additionally, when exporting files between different programs the export settings should have been decided and set so that the right settings for different asset types are used each time.

Dreamloop has created an art pipeline and guide for art that is shown for everyone in the team. The art pipeline defines the person specific pipeline on how everything works from person to person. The art guide is much more specific and technically detailed. It shows the specific settings and requirements for different shaders, naming conventions, animations, material and textures.

Lead designer said that additionally one of the problems with art side is the lack of tasking.  Art department often has to ask what they are supposed to do at that given time, because design wise it is not clear what the artists are supposed to be doing. He feels that the problem is also the fact that he is not an artist and does not consider his ideas necessarily worthy so he often gives all the tasks for the artists.

## 3.6. Art creation process

The team started a new development process for the art. In order to create better character's, the team split the development into six different parts. First the art team would think of what kind of character they wanted to make and created different ideas as a group. After those ideas Dreamloops own concept artist and animator would draw some rough sketches and also find different reference images from the internet and then come up with more specific rough concepts. Rough concept essentially means an outlined drawing that does not contain more than the rough edges and basic shadowing. After this, the rough sketches were given to another concept artist who would then take those sketches and do more high quality concept art. High quality concept art is defined by going more into detail with the rough sketches. Artist might add more clear outlines that define the character's shape better, body features, color and different poses. Figure 33. shows concept art iteration of Makapoa race.

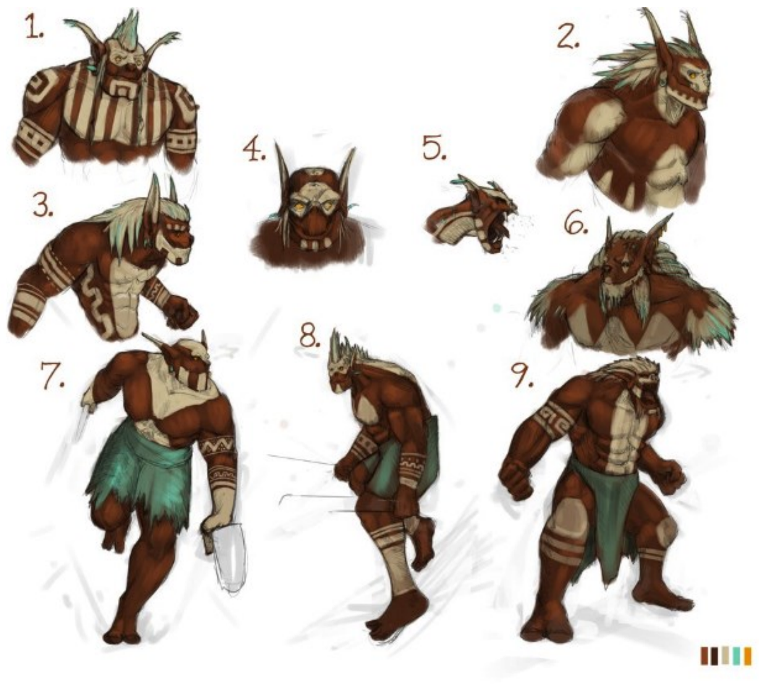


Figure 33. Makapoa Concept art

After the concept art has been made, 3D artist can start to do a high polygon sculpting of the drawn character. High polygon sculpting can be shown in figure 34. The first face shows a high polygon sculpt with 8 million polygons. Sculpting refers to a process that gives the ability to refine the shape of the model and add more detail to it.

After the high polygon sculpt has been made, it is time to do the retopology for the 3D model. Since high polygon sculpt can have several million polygons in the model it would be practically impossible to implement them because they would cause huge performance problems in the game. In order to implement the model to the game without major performance issues, retopology has to be made. Figure 34. does also show the difference between a high polygon sculpting and a model where retopology has been made. The goal of the re-topology also depends a bit on the platform the game is going to be on. Mobile games tend to have very low polygon requirements, while console and PC platforms allow much higher polygon counts per object. Additionally, the quality will also depend on the scene and how close or far away the object will be looked at.

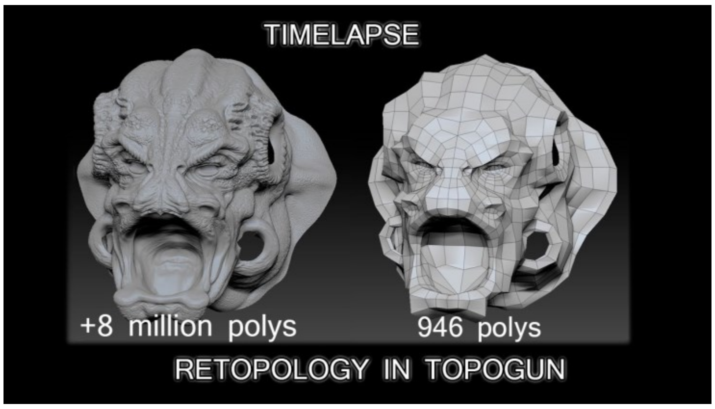


Figure 34. Why we use re-topology (Timelapse making of – Retopology in Topogun, Youtube)

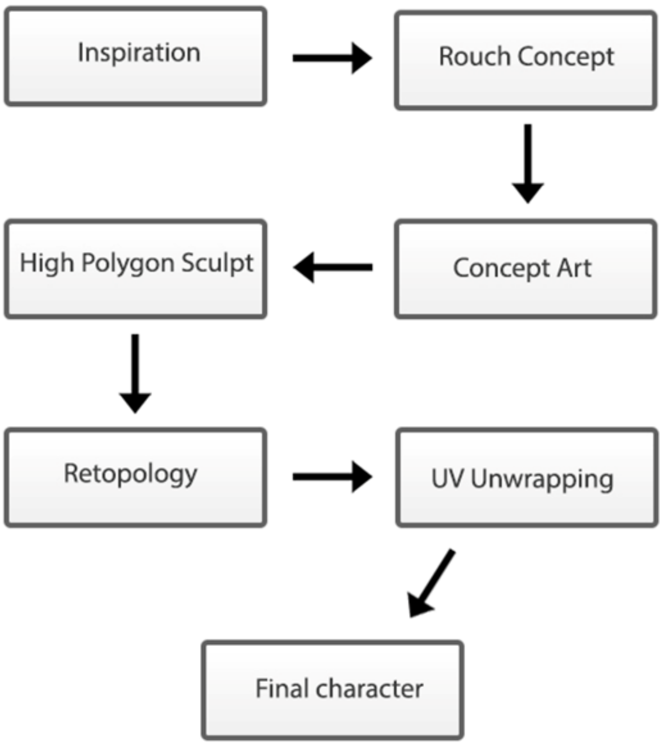


Figure 35. Character creation process

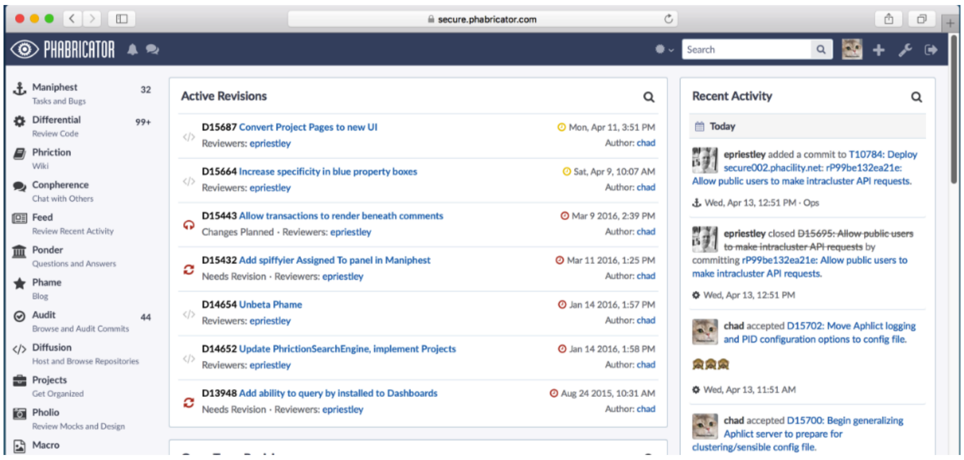
After the retopology has been made it is time for the final phase of the character. UV unwrapping will be done, which refers to turning the 3D model into a 2D picture and allows the artist to create textures that will be applied and the character will be implemented to the game. The whole process is also shown in figure 35.

### 3.7. New GDD for Khalea

During this of course the Game Design document had to be changed. The lead designer thought it was easier to copy the Glory GDD and then re-read it and make the changes accordingly. On most part the GDD stayed the same but there were sections that expanded and some of the sections saw some reiteration in terms of design. Of course, the new characters were put in there and they went through few iterations in terms of their background and story elements. Some of the management features were also scrapped and rethought because they worked better when the online elements were still intact.

### 3.8. Project Management Changes

After Vasara joined the project Dreamloop quickly noticed that their project management needed to be re-thought.  Suddenly they had four new people that required management and still using excel seemed like a lot of additional work. They thought of a solution to their problem and found a program called Phabricator, which they have been using ever since to manage their project. Phabricator basically allows better ways to give different tasks to members, review code, having own Wikipedia for artist pipeline and bug reports. What phabricator contains can be seen in figure 36.



D15687 Convert Project Pages to new UI
Reviewers: epriestley
Mon, Apr 11, 3:51 PM
Author: chad

D15664 Increase specificity in blue property boxes
Reviewers: epriestley
Sat, Apr 9, 10:07 AM
Author: chad

D15443 Allow transactions to render beneath comments
Changes Planned · Reviewers: epriestley
Mar 9 2016, 2:39 PM
Author: chad

D15432 Add spiffyier Assigned To panel in Maniphest
Needs Revision · Reviewers: epriestley
Mar 11 2016, 1:25 PM
Author: chad

D14654 Unbeta Phame
Reviewers: epriestley
Jan 14 2016, 1:57 PM
Author: chad

D14652 Update PhrictionSearchEngine, implement Projects
Reviewers: epriestley
Jan 14 2016, 1:58 PM
Author: chad

D13948 Add ability to query by installed to Dashboards
Needs Revision · Reviewers: epriestley
Aug 24 2015, 10:31 AM
Author: chad

epriestley added a commit to T10784: Deploy secure002.phacility.net: rP99be132ea21e: Allow public users to make intracluster API requests.
Wed, Apr 13, 12:51 PM · Ops

epriestley closed ~~D15695: Allow public users to make intracluster API requests~~ by committing rP99be132ea21e: Allow public users to make intracluster API requests.
Wed, Apr 13, 12:51 PM

chad accepted D15702: Move Aphlict logging and PID configuration options to config file.
Wed, Apr 13, 11:51 AM

chad accepted D15700: Begin generalizing Aphlict server to prepare for clustering/sensible config file.

Figure 36. Phabricator insides

Dreamloop split their meetings into three different ones. Art meetings are held two times a week where all artists and animators gather and discuss the current tasks and future tasks.  When required, these meetings will also have more technical meetings where also some programmers to need to be present. Programming meetings are also held two times a week.  Due to Phabricator having a feature for code reviewing, it is also a crucial part of these meetings. In these meetings programmers go through all the new features and bugs that have been discovered. The code is reviewed by three other programmers and they communicate if the problem is correctly fixed or if there is a better solution to the problem or a feature. One of the most important parts to follow in these meetings is following the bug correction, which means that fixing bugs should not create new bugs inside the code. So far Dreamloop has been really happy with the fact that it does not occur inside their project. The rate of new bugs occurring after fixing previous once has been extremely low and it is also high due to implementing and using StrangeIoC.

### 3.9. The tools for creating the content

One of the least talked topics of game development is the discussion of tools and how the content of the game is created and designed. Content is often seen as the graphical presentation, mechanics and what is directly in front of the player. For Challengers of Khalea there has been in total four different separate systems that they had to build in order to create the content of the game and manage specific areas. One of the first content creator tools that were created for Khalea were the arena and ability editors.

#### 3.9.1. Arena Editor

Arena editor is a tool that allows the designer to create different types of arenas for Challengers of Khalea by placing props and warriors inside the early built arena model. Props are basically another name for 3D assets that can be accessed through the arena editor. Warriors then are just basically spawn points inside the arena

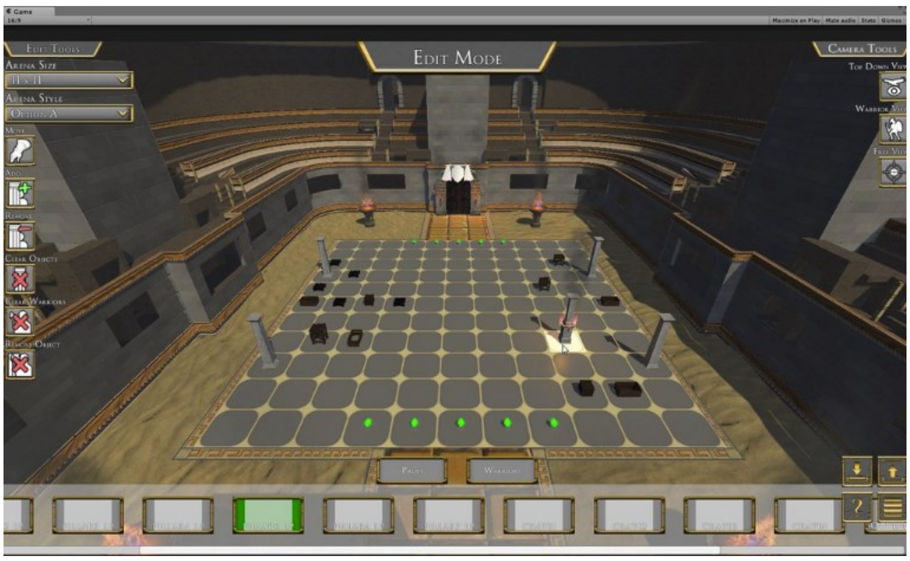


Figure 37. Early arena editor

that are divided into player’s team and the enemy team. These are the main tools inside the editor that are needed to create the actual content.

Figure 37. shows one of the first versions of the arena editor that was built for Khalea. You had options to choose the arena size, which at the time was 10 x 10 grid and only size available, and the ability to select a different arena theme. The editor also had tools so props and warriors can be moved inside the grid to different sections of the arena and there is also an option to clear all the props and warriors from the level. Additionally, the editor allowed different possibilities to view the arena. Lastly, the editor included the ability to save and load the created arenas.

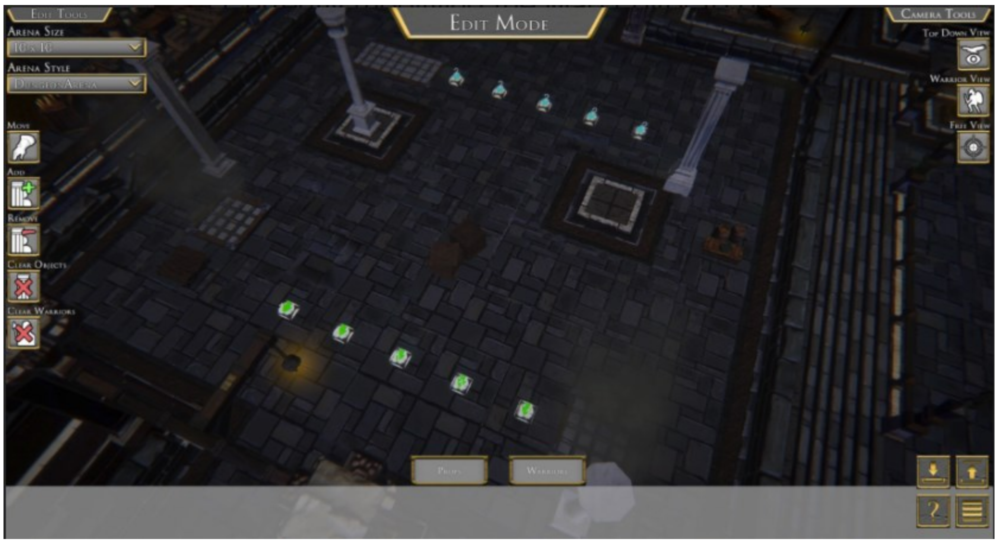


Figure 38. More current version of the arena editor

The arena editor is still changing and being modified to better fit the needs of the project. Newer version can be seen in figure 38. and the tool will still get a new user interface and new functionalities. Once the new version of the editor is ready, you will also be able to see props visually without having to first add them to the arena and you will also be able to see the current layout of the arena in a loading preview screen.

### 3.9.2. Ability editor

Purpose of the ability editor is to simply allow designers to create new abilities for the warriors. Abilities are skills that warriors are able to use during the match and there are different types of abilities based on certain factors like the class, weapon and race.

Figure 39. of the ability editor shows the information that can be controlled inside the editor. Ability ID refers a to an identification number for the ability so that it can be properly recognized and loaded inside the game. Then ability can be freely named, described and restrictions applying to that ability can be inserted. The current version is a bit outdated in terms of the design as there are more ability types coming in the near future. Damage over time spells (DoT) and healing over time (HoT) are few examples that are going to be implemented later, which of course means that in order to main them work, these types of spells need to programmed and added to the ability editor as possible spell types. The fact that more ability types are coming later brings out the fact that projects grow and change over time, new requirements come and change and it is always lot of additional work.

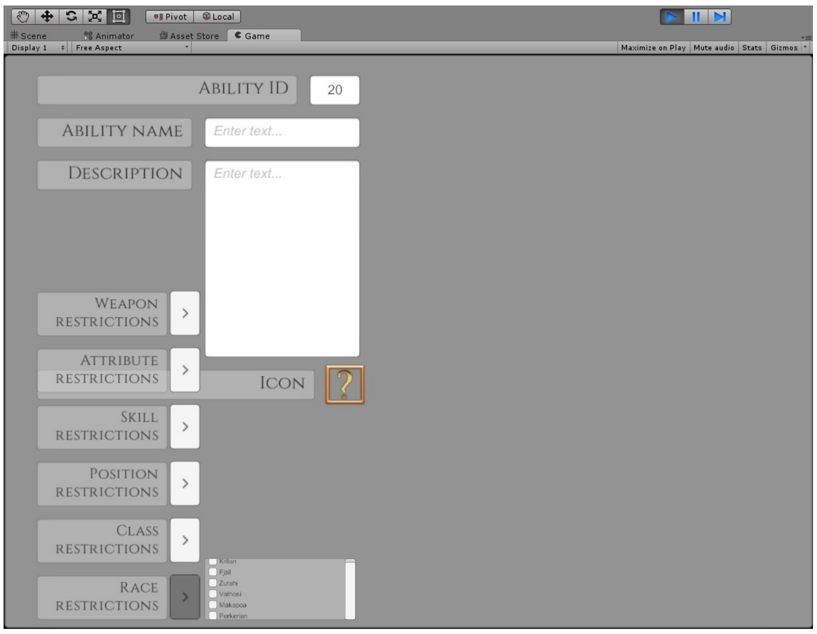


Figure 39. Ability editor system

### 3.9.3. Bring out the Puppet master

Puppet master is a plugin for Unity 3D that provides different functionalities for the Unity's mechanim system. Mecanim according to Unity is a tool that allows animating humanoid bi-ped characters and allowing building transitions from different animations to other animations. The system is also described to be able to be used for animating UI elements, opening doors and changing light for example. Puppet master was brought to the project before the summer of 2016 right after StrangeIoC was implemented.

Puppet master according to the lead designer of Dreamloop brings new interesting features for them to work with. Essential parts of puppet master plugin are that it allows physical based animations for characters. What the team did before in terms of animations was that they had death animations for characters, which can be sometimes a fair amount of additional work especially if different death animation is race specific. Physical based animation removes the requirement for these animations, because puppet master allows death animation to be created just by controlling physics elements. When a character is supposed to die, the game can simply just remove that character's weight from its feet and the character basically collapses by itself without need for specific animation. Visualization of a falling simulation can be seen in figure 40.

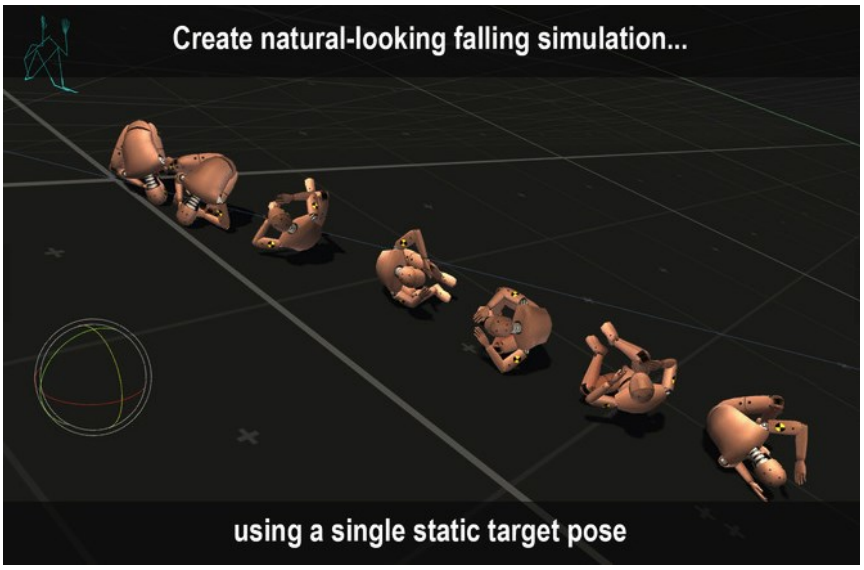

Create natural-looking falling simulation...
using a single static target pose

Figure 40. Puppetmaster introduction [Puppet Master Plug page]

Additionally, Puppet master allows using ragdolls and Inverse Kinematics (IK) in the game. Ragdoll system allows characters to react to different interactions more naturally without having to create a specific animation for it. Ragdoll system allows previously explained death animation to happen, and also allows the loose puppet in the ground to not restrict other warrior's movements due to the ability to be able to control character physics. Inverse Kinematics is a physics term and in the context of games allows for example weapons to be inserted to character's hand, or allows character's feet to be attached properly to the ground, especially if there is rough environment where the character can walk.  Additionally, it can allow focusing the camera to specific points like objects.

Implementing the plugin to the project took about one and a half months total and there was a programmer and a technical artist who worked with the plugin through that time. Both the lead designer and the programmer thought in the interviews that the process took fair amount of time and was a difficult task. The main reason for the difficulty rose from the fact that their programmer did not have much experience with animations in Unity, so learning the essential information took most of the time for him. Additionally, because StrangeIoC was just implemented it also took time to figure out how to combine these two together. The team also thought of a possibility of implementing similar system themselves but in the end purchasing the plugin would save them both time and money. They thought that creating a similar system would have taken them roughly two months and also additional time in the future as the system would have to be refined.

### 3.9.4. Weapon arc system

Weapon arc system is an additional feature that has been implemented to Khalea by Dreamloops technical artist. This system allows better visualization of the weapon trails that are created when a weapon is swung. More technically speaking the system calculates the cusp of the weapon and then creating a certain swing length and creates a trail for the weapon in real time.

Dreamloops technical artist explained that the reason why this system was implemented was the fact that because they wanted sometimes to implement very fast weapon swing animations, and such fast animations can be hard to understand if there is no trail for the weapon. However, if trail is implemented, then the weapons movement becomes more apparent as can be seen in figure 41. It also eases animators job because then only have to think of the weapon movement and not how trails need to be implemented and if they would have to be implemented in another way. It also worked as a communication tool between the technical artist and the animator, as with the system it was possible for the artist to show how everything works and how the animations need to work instead of relying on written and verbal information. The system also sets requirements for the animators and makes it easier for them to work as they do not have to think as many points while doing the animations and the rules are more clear.

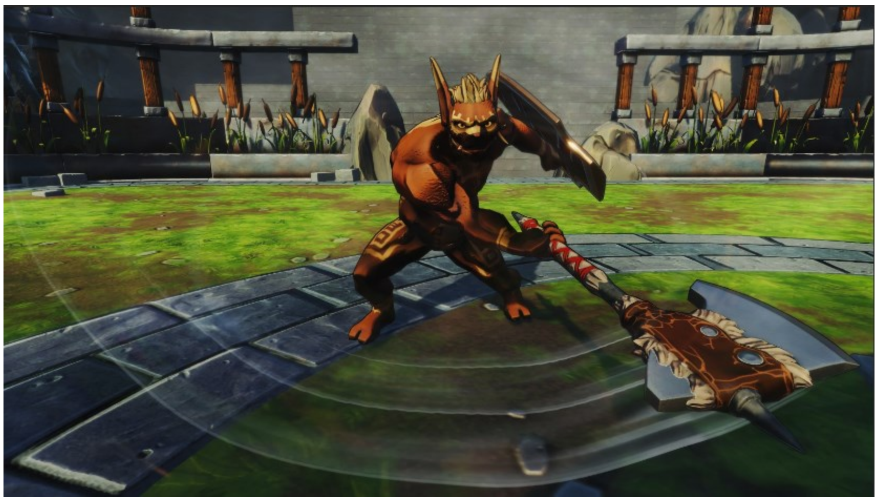


Figure 41. Weapon arc system in motion

The implementation of the system started around in early August 2016 and it took two weeks to create the first iteration of the system. However, the idea for the system was born already few years back due to the interest in using weapons and attack animations in games. The system will also see more iterating in the future according to the technical artist as he has few additional ideas on how he would like to expand the system. He said that in the future this kind of system could also be reused in similar types of games.

## 3.10. Marketing of Glory and Challengers of Khalea

The first thing that Dreamloops marketer did for the project was that he started doing competitor research. This type of research meant that he would research games that shared similarities of concept with their current game idea. Due to how the game in the first prototype played out, marketer calling it more of a chess with different units he also started to look for more turn based tactical games in order to find out what elements in the game he would like to have most attention to. He ended up researching different types of games like Gladius, which was a great influencer for Glory at the time, Fire Emblem and other turn based tactical games. He specifically wanted to find out what their game would be and the target audience for it.

Online communities were a big source of information for Dreamloop and researching different game forums were a great help in order to determine what kind of gamers would buy turn-based tactical games. The problems of course occurred with games that were quite old, essentially somewhere between 90s to 2000 games that did not have online communities to search for the information. He said that it is really easy to find out the age ranges and geographical location of the player, but it is really hard to find out which of those people is actually going to like the game. He continues that often one thinks that he has everything to determine this but in reality he feels that the only way to actually determine it, is to test it with ads in order to see people's reactions. This is something Dreamloop has started to do more during the recent year and that is because earlier they did not have the resources to do so.

Early in the project there was a clear plan for the marketing, to get information out about the game. Press coverage and articles were a big part where they wanted to aim for as they felt they were the most valuable parts on getting information out. Marketer considered that it was also essential to get out some information about the studio as well as he considered himself to be not only a game marketer but also a studio marketer. As the project progresses Challengers of Khalea will start to see more and more visual marketing in social media and other sources, but they had already had few posts in the past that have gone viral and have been a success in terms of marketing. Lot of the marketing they did and still do is free marketing as they do not have a market budget or much ability to have a constant one.

The marketer considers that they have had three different phases that have modified the way they do marketing. First there was the Glory, then post Kickstarter Challengers of Khalea and then the post Vasara Khalea project. He says that during all these phases the way he did marketing has changed a bit of course because the game changed from more Gladiator themed to historical fiction style game that was closer to realism. After the post Vasara phase the game started to turn into more fantasy based management game. After post Vasara Khalea they had three different parts that connected people together: People who like fantasy, tactical RPG's and management games so it creates a solid middle ground for the game.

Dreamloops marketer feels that competitor research is very crucial part of the development process even though every now and then successful games emerge where competitor research is basically non-existent. He feels that if not doing so you might just be shooting in the dark if you do not know which way the market is leading at that moment. He is especially interested in how their competitors work and what kind of strategies they have in order to get the information about their game out and learn from that and its results.

Dreamloops marketer also feels that sometimes competitor research does not necessarily achieve what it is supposed to. He feels that sometimes people do not like the method and they think that it is going to change their game, but he also argues that every time you are inspired by something, you are doing competitor research in some level. He feels that there has been conflicts of interest and differences of an opinion inside the team about the issue. However, he also feels that it might be due to his approach to the issue and also sometimes it is a communication issue, and he has had to change his approach to communicating these ideas forward in a way that better fits the situation and different members.

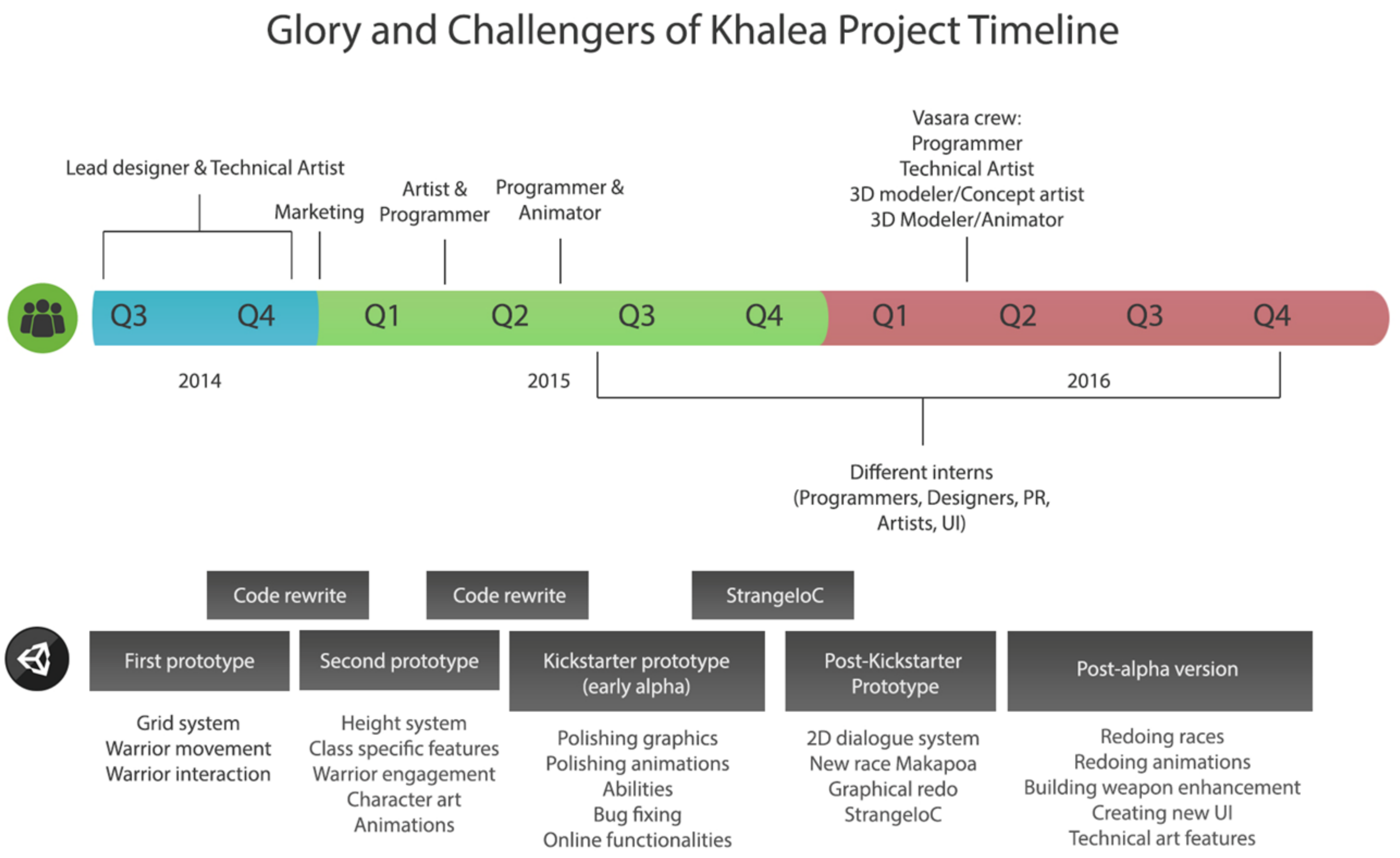


Figure 42. Glory and Challengers of Khalea timeline

# 4. Discussions

In this chapter, we will discuss different areas of the project in order to find out key differences between the case study and the literature and additionally to find problematic areas in the development process of Khalea.

## 4.1 Start of the project

Like many other game projects, Challengers of Khalea started like thousands of other game projects start, idea from a previously made game or a combination of games moulded into a new experience. The start of the project however might not be very typical as the idea was presented in a form of a pitch to a company the two co-founders worked at in 2014. Even though the company did not want to start further conceptualization of the game, the support of their peers obviously meant a lot for the co-founders and pushed them to continue with the idea. Conceptually both founders appeared to have very clear idea of what kind of game they wanted to make in terms of mechanics and gameplay in general, inspiration coming from few older games, Areena 5 and Gladius.

From the perspective of conceptualization, the question of choosing between a single player experience and multiplayer experience should be one of the bigger questions for the project in the pre-production phase. This Dreamloop never had in the beginning and the multiplayer functionality was decided very quickly. I consider this to be a big question as going multiplayer is a big risk because of two factors: There are already multiple online multiplayers on the market, Massive Online Battle Arena games (MOBAs) being most famous ones and even to this date they remain extremely popular. Online multiplayer games also require lot of resources from the project. They require someone capable of online functionality programming, setting up servers, player balancing, online security and multiple other factors in the form of testing and high upkeep after the release.

*"Sure, adding online multiplayer to your game is massively time-consuming, very costly, and not exactly the greatest fun you'll ever have -- but it can potentially bring in multitudes of extra players and sales, so it's always worth it, right?"*

*"Making and launching a multiplayer game was arguably the most stressful thing I've done as an indie developer, and I enjoy my job and my life too much to mess around with all that sort of stuff again."*
*- Mike Rose 2014*

It is also known that single player experience also has its requirements and difficulties, but I firmly believe that the amount of possible problems come in much lesser form then when adding multiplayer functionality.

In terms of conceptualization, jumping to making Glory was a shot in the dark. It had a passionate and big goal that largely came from the multiplayer perspective. Most of the literature does not talk much about multiplayer games, but they are surely a big challenge for developers, especially new ones and even bigger companies have failed with their multiplayer concepts from time to time.

Overall the project had great ground to stand on, proper professional approval, the desire of starting and knowing what they wanted to make, but I think the problem was that the conceptualization was not taken further enough at the start. This means clearly defining different aspects of the game, not only gameplay and mechanics but also the characters, stories, abilities and other necessary aspects that depend on the game type, referring to the fact that a game should have a strong conceptual ground to build on. The issue with this type of approach however was the fact that both were programmers, who did not have very artistic approach in concepting and they did not have proper artists at the start of the project. The literature however heavily emphasizes the importance of conceptualization and it should be done early and thoroughly. Conceptual problems also affected the success of the Kickstarter, because Dreamloop did not have a proper art direction and both conceptually and visually the characters were not interesting enough. The feedback also approached the fact that online multiplayer was not a very interesting concept.

## 4.2 Engine Selection

One of the least discussed topics of game development is often the matter of choosing the game engine for the project. There is no denying though that choosing the engine for the project is extremely important and understanding the limits of that engine as well. For Dreamloop this process was inevitably as their experience was tied to knowing Unity and how to use the engine, and it was an easy tool for early prototyping and Unreal Engine was not even free early in the project. After the prototyping, founders discussed the possibility of switching to Unreal engine, but due to their skills and risks taken into consideration, they decided to continue with Unity. This type of discussion was smart from Dreamloop because they would acknowledge their strengths in terms of resources and they also considered the risks that would come if they would jump into a new engine where everyone would have to spend time to create new knowledge.

### 4.3. Prototyping

Before starting digital prototyping, Dreamloop as a positive notion decided to start with paper prototyping, which was suggested heavily in the literature due to easy iteration, changing process and low costs. Unfortunately, there was no chance of presenting any pictures of the drawn prototypes from Dreamloop due to losing those papers, but general descriptions of the prototypes and results were received. Prototyping was done later in 2014 and it had two parts: creating different types of characters by spending certain amount of points for different stat and then these characters and teams formed had a paper board to fight against other players on a 16x10 grid. Dreamloop collected the results from these sessions defining how many of different classes were selected, average points per stat and defining max and min health point amount, attack etc. While I consider that the approach to start paper prototyping was good, I felt that the paper prototyping was not taken further enough and only conducting two paper prototype tests is not very extensive.

I think that paper prototyping should have been a larger process, carefully thinking of how the prototyping could help them to find out positive aspects from their ideas and then executing those plans in order to answer important questions about the game. The prototype could have been more extensive, giving players the ability to choose from five to seven different abilities per class. This could have given them a broader understanding of how everything would work out by testing different abilities in order to understand how some abilities could be problematic with different classes and to answer other ability related questions. Additionally, testing out the further combat would possibly give ideas of what kind of experience battling as a whole would be. Because Glory was very management and fighting heavy game, more extensive prototyping could have given them a better understanding of the player experience they are creating. Discovering the fun in battling and management should have been extremely important in paper prototyping.

Digital prototyping for Dreamloop has been a slow process essentially extending from 2014 to 2016 and code rewrites have been done almost between every prototype. How long prototyping takes will of course depend on different aspects such as how much content the game is going to have in terms of gameplay and mechanics, what platform will it be made and how many people are available for prototyping. However, the literature describes prototyping as a quick process, that is supposed to answer certain questions and validate different design ideas. Knowing this, I think that the frequency of creating prototypes for Dreamloop was fairly slow, even after more people started to join the project in 2015. Dreamloop did code rewrites because they prototyped in a faster phase in order to receive results faster, which essentially lead to weaker code structure. I do however think that because prototyping is supposed to be fast and not elegant, code rewrites should not be necessary in early prototypes, but they should have done so that everything functions as needed and the system allows to be built on if necessary. In its essence, the prototyping should have been generally quicker and avoiding code rewrite if possible during early prototyping process. Dreamloop's prototyping process was more about creating the prototype and then refining it one by one until the final product is created and they also tested different mechanics and gameplay ideas on the go with prototyping.

## 4.4. Game Design Document

The first parts of the GDD started as a pitch that Dreamloop did for their previous company and it was further extended over time since 2014. The information that was presented in the GDD was on the most part what has been expressed in the literature. In terms of gameplay, mechanics and general rules of the game were presented well in detail. In terms of visual style and conceptualization the GDD was lacking. While positively it had visual style comparisons in order to create a better understanding of what kind of style they wanted to create, it did not extend very far beside this. The GDD did also lack few specific elements such as a small list of abilities for the warriors. While the GDD did have few sample abilities to show what the abilities might include in terms of mechanics, there should have been a more concise list of abilities to create a textual feeling of what the gameplay would be like in the end. This would be because turn-based tactical RPG might not describe everything that is to it. For example, it is hard to understand the phasing of the gameplay in the arenas, is it slow? is it very quick phased? Does it include lot of ups and downs per match are few questions that could be asked and explained in the document.

Literature also expressed that GDD itself is also a process that slowly extends when ideas are confirmed and it should be fully completed during early production. In that regards it was ideal to notice that Dreamloop did not take the route of going into overly detail with the GDD which, according to few sources could be problematic approach and would require lot of time and input. From 2014 to 2015 the GDD extended to reasonable sized document of 59 pages. The GDD was well prepared and described the features and mechanics well in detail but then again was not too detailed, allowing room for possible changes. It positively also used lot of pictures to visually present information and form in order to make it easier for different personnel to read and understand.

Looking at the GDD from the conceptualization perspective, it becomes apparent that the GDD did not seem to cover the look and the feel of the game enough. The GDD was more about creating blueprints and documenting design ideas about the game for Dreamloop and not about writing or documenting the player experience that they wish would be there. While there was some discussion about the visual style of the game, few concept art pieces and few lore sections, the GDD felt that they were very separate pieces that would not really come together to properly describe the product. The GDD was positively made in the way that it was not overly detailed and it essentially had room for discussion in terms of the design, mechanics and features but the concept should have been more detailed.

Secondary problem with the GDD was that it was not regularly updated. Due to heavy workload, the GDD was left unchanged for a while and it was basically changed when design discussions took place about something that needed to be implemented. In some areas, also the Phabricator that was used for tasking and as a project management tool, replaced part of the GDD by having its own Wiki page and including technical details such as art and animation pipeline information. I think that the standard GDD should have been replaced by the Phabricator in total because Phabricator is much more in use by people and it is regularly under watch. It should be essential that the GDD is regularly updated in a way that everyone in the project can see what is being updated and that GDD would also be part of the wiki, where those updates could possibly be seen. Communicating changes and further ideas is essential and it should be done in a way where the GDD is more approachable.

### 4.5 Resourcing

Resourcing was problematic area for Dreamloop because they started with low resources and started to work their way up as the project progressed. Generally, the literature talks about resourcing and its connection to conceptualization, but not much about actual process of resourcing. The process varies a lot, ranging from one single person doing most of the work to more people coming in slowly. Dreamloop started with two people, then hired one more for the marketing and slowly as they got a prototype ready, more people could be hired for the project. I think that the most problematic part of the resourcing done early is the fact that the concept itself was big and it would require great amount of different personnel with different skill sets in order to make it work. The issue is expecting that you might find these people, partially relying on luck when you could start with a more realistic concept that require people that you are much more likely to find.

Resourcing is also a problem currently in the project. There are great amounts of tasks for different people, programmers, artists, designer etc. and Dreamloop has been struggling a bit especially with the design of the game because their CEO has also company related business to deal with in addition to tasking and design work. While their CEO expressed that he will be able to work with the design every now and then, the question remains that the design should be reviewed and validated. Additionally, design is also needed for creating better user interface for the game and for management games such as Challengers of Khalea, presenting information is essential and requires completed designs in order to be implemented. Few company personnel also expressed some concern about the design, but also about the resources for programming. Recently one of the features of the game was scrapped in order to allow programmers to work on other features, emphasizing the fact that lack of resources affects the game in different ways. Resourcing also affected the success of the Kickstarter by not having a proper artist to work on the visual style of the game.

## 4.6 Project Management and methods

Dreamloop received approval about their project management from other teams and rightfully so. It was positive that in terms of tasking and having meetings the management worked nicely. They clearly understood the importance of managing a project, which came through the interviews. The communication between the team seemed fluent and features were built in decent amount of time and generally the atmosphere of the project seemed to be excellent. The team had occasional meetings where they discussed problems regarding programming, art, other architectural parts and general gameplay and marketing. Regarding these, Dreamloop did a great job at managing multiple people, which could be a potential downfall of a starting company.

It is essential to understand that each project has a different style of management and different methods might not be suitable for everyone due to different reasons that come from personalities, project size, personnel size and general attitudes of different methods. When asked, Dreamloop felt that they did not consider using different methods like agile or they tried agile, but in terms of the tactics it did not necessarily appeal to them or seem to help them much. I do understand that the company was hesitant to use different methods, as using them due to applying and possibly combining them together can be a taunting and risky task because often one of the mistakes of applying these methods is not understanding how to apply them correctly. Despite this, I do think that the company should have used time to try to adopt agile into their development for few reasons. Agile is highly used in game development in a form or two and it does provide positive aspects for development especially by creating better mentality. General principle of agile is to find the fun and the knowledge early, thus confirming that prototyping is essential in the pre-production phase, which I would have wanted to see in this project as well.

## 4.7 Iteration

Iterative development is extremely essential to game development as presented in the literature and it extends to different areas of development and not only game design. Dreamloop does have an iterative process and they do iteration almost on a daily basis that came through the interviews but also through personal observations. Especially on the programming side, if there are some problems in the design they quickly give feedback and propose a solution to fix the issue. These are however small scale iterations that are done inside the company but there have been larger scale iterations as well that are done due to technical issues such as implementing new plugins to the system. So far, the iterative process has been more team focused, and so far playtesting has not been done for the game. This is an aspect that I hope will be done in the near future, early 2017 due to the amount of value playtesting can provide.

Another iterative process for Dreamloop comes from the art. Their iterative character process starts as general ideas and finding references to use and after concept art is drawn based on those ideas. What was noticed here was the fact that when the concept is taken to another artist for 3D model creation, the concept of the character ended up being very different from how the 3D model ended up being. Lead artist of Dreamloop said that sometimes he might be too agreeable to further suggestions, that then end up heavily changing the character's appearance in the end. I consider this kind of approach to be problematic for few reasons. Firstly, concept art is a communication tool that is used to represent the 3D that is about to be built. Concept art should be the phase where iteration is done by making few sketches, refining them, making from 2 to 3 different iterations of the character, taking feedback and making changes based on that feedback until you have a great concept. If concept art is done and changed immediately in the next phase, then how important is concept art? Also, if the concept art is quickly changed, then it cannot be used as a marketing tool either, making it easily wasted work. I would rather see Dreamloops concept artist and 3D artist to work together on the conceptualization in order to create any kind of changes and mutual understanding of the character early and in the concept phase.

### 4.8 Technical issues and direction

After the Kickstarter in early 2016, Dreamloop had to go back into conceptualization and also take a fresh new look at their project from the technical perspectives by looking at coding dependencies in Unity and the animation and physics problems they had. Dreamloops actions were positive in a sense that they did not ignore their problems, they took those problems and understood that if they were to continue with them it would create even more problems in the end for them, and fixing them would only get even more problematic the further they would go. While implementing StrangeIoC was a demanding task, I can say that it has been a good decision on their part to create a better basis for the game to build to. Also, choosing and implementing puppet master plugin was a smart decision as it helped them with their long term plans of development, proving that the company understands what their requirements are at the moment and what they will be in the future.

During my time at Dreamloop it has been extremely welcoming to notice that especially their technical artist is constantly interested in new technology and plugins that come available for Unity. He often browses different plugins and is interested in the fact that if they would be something that would help their project somehow. A good example of this was the founding of the NoesisGUI that provided a much better way to build and manage the user interface of the game. NoesisGUI is a Unity plugin that allows user interface to be built by using XAML

(Extensible Application Markup Language), which shares similarities to languages such as HTML and CSS. The team soon also asked the important questions about the plugin: It is built with XAML, will it work with consoles? How easily can we iterate with it? How easily can it be learned? These are all important questions that the team had and very essential ones before any kind of implementation is made.

The implementation and programming of the weapon arc system also took time but was beneficial for the game in order to create more stability for the animators and a way how to produce better quality animations and visuals. The system will likely also see some new features in the future but so far the system has everything built in it that needs to be there. It has been refreshing to observe and notice through interviews that Dreamloops strongest points have been their technical skills and understanding.

### 4.9. Merging with Vasara

Because the company merged with Vasara Entertainment, the company had to change the way communication and project management was done due to increasing amount of people in the project. The meetings were changed and split. Two times a week artists and programmers would have separate meetings in order to discuss issues and new content related to these areas and additionally the whole team would have weekly meetings in order to clarify where everything stands for the whole project. Change of direction was great improvement for Dreamloop as they could communicate and discuss the matters more efficiently within different groups. Especially the way programming meetings worked were efficient because they did code reviews and had discussion of what worked and what practices could possibly be changed and done better. Code reviews also focused to follow if bug corrections would create more bugs, which was a good practice.

Previously Dreamloop did not really have a documented pipeline until Vasara joined the project and it fortunately changed so clear pipelines were created for both art and animations. Currently Dreamloop has nicely documented pipeline that includes general work flow, feedback and technical pipeline. One of the reasons why this was done was also the fact that Dreamloop currently has two trainees in the project, who are not as experienced artist as the current ones so it is very good to have documentation for them, but also to have something for all of the artists to remember.

Merging with Vasara did not come without problems and discussions with the Lead Artist and the CEO provided good information about the issue. One of the core issues was of course the fact that the general understanding, skillsets of the artistic side like modelling and animation were a bit different between both sides. It took a while for them to find a mutual understanding and that the communication becoming more fluid. In terms of understanding and skills one of the core problems was the fact that false expectations formed easily so that other side would assume that the other half would specifically know what to do in different circumstances when modelling or doing animations. This would lead into situations where assets could be done with false requirements, thus partially wasting time and work. Second issue was the fact that some of the artists worked distantly at their homes basically on the other side of the country, which would sometimes create difficulties in the communication between the lead artist and the others. CEO expressed that they are still in an adjusting period with the merge, basically deciding who the lead artist will be in the future and how different kinds of tasking will be done in terms of art assets.

## 4.10 Future aspects of Khalea

Currently Dreamloop Games is working on Vertical slicing. This means preparing and polishing the art and graphics for characters and few arenas, polishing and implementing rest of necessary animations for characters, creating more abilities for gameplay and creating more armour assets for characters. It was excellent to see that the company is aware of the state of their game and is currently polishing it to meet certain requirements. During the summer of 2016 Dreamloop approached their first publisher candidate with a playable prototype of the game and the received both negative and positive feedback from the publisher candidate. At the time the visual style of the game was still in the middle of development and it was something the publisher was not pleased about. On the positive side the publisher did enjoy the mechanics of the game.

Additionally, the presentation of Dreamloop of their concept of the game needed more work. Because there are still lot of missing components in the game such as abilities, which there are inside the prototype but even Dreamloops lead designer described them to be very primitive and boring. This is one of the problems what the lack of design resources do bring to the table. If the design is not ready it will carry on forward to most areas like marketing, art and programming. If for example marketing person cannot fully understand and interpret the concept and idea of the game, how can he make the publishers understand it?

The current estimation of Challengers of Khalea's launch is at the end of Summer 2017, which at the moment sounds very unrealistic. This comes from the facts that there are still lot of components that are not ready. The user interface of the game is currently being redone, the Artificial intelligence (AI) of the opponents is still in its early phases, the lore and the story of the game is also being produced at the moment, quality assurance does not really exist nor does outside playtesting, abilities and types are still in production and level design has no maker currently.

However, there are other question marks in the air for the project that come from the publisher side. If Dreamloop Games makes a contract with a publisher in the future, it could potentially shape the scope of the game. If the scope of the game is affected, then it will very likely lengthen the production of the game and also increase resources. Additionally, publisher coming along will require stricter production scheduling and communication with the publisher partner so it will require more resources from the production side.

Dreamloops Lead Designer said that they have moved to production during summer 2016 and the team is currently creating assets to represent the final game. The problematic part of being in production is that changes in this stage take time and in terms of programming any kind of major change is problematic. This creates an interesting problem because there is still design work that is not yet done, such as narrative development, progression, certain aspects of management in the game and they have not been yet even validated. If the designs that have been made for the game work, then there is no problem. However, going through the literature and design talks, you cannot really expect everything to go smoothly, creating the nervous feeling that there are going to be problems and fixing those problems is going to take lot of time and effort. This is a case that I would call de-synchronized development, which would be a situation where part of the development has already processed to the middle of the production and other parts such as design is somewhere between pre-production and production, basically creating a project where different areas of development are not properly synced.

## 4.11. Refined game development process for pre-production and production

I wanted to refine the process of game development based on the literature, which arose from the facts that often literature and diagrams that are provided do not capture all that is to the game development process and its phases. They provide very simplistic view of what the process is and does not really open up the phases to fully understand the process and especially larger projects that have higher scopes. Together however the literature brings up excellent points on how to more completely visualize and explain the process.

Figure 43. describes the pre-production phase more clearly. Pre-production starts with the conceptualization where the developer should figure out few of the essential parts of their game like key features and the audience for the game. Conceptualization also should include the big questions like the genre, platform and if the game is 2D, 3D or 2.5D. These are the big questions because answering these questions will provide better understanding of what kind of engine is best suited for the project based on the concept. As it was described in the theory and what was also brought forward is the fact that certain engines like Unity is great for prototyping but it also suffers from other issues like dependencies and is generally good for 3D development but not necessarily the best for 2D even though Unity has progressed in terms of 2D development.

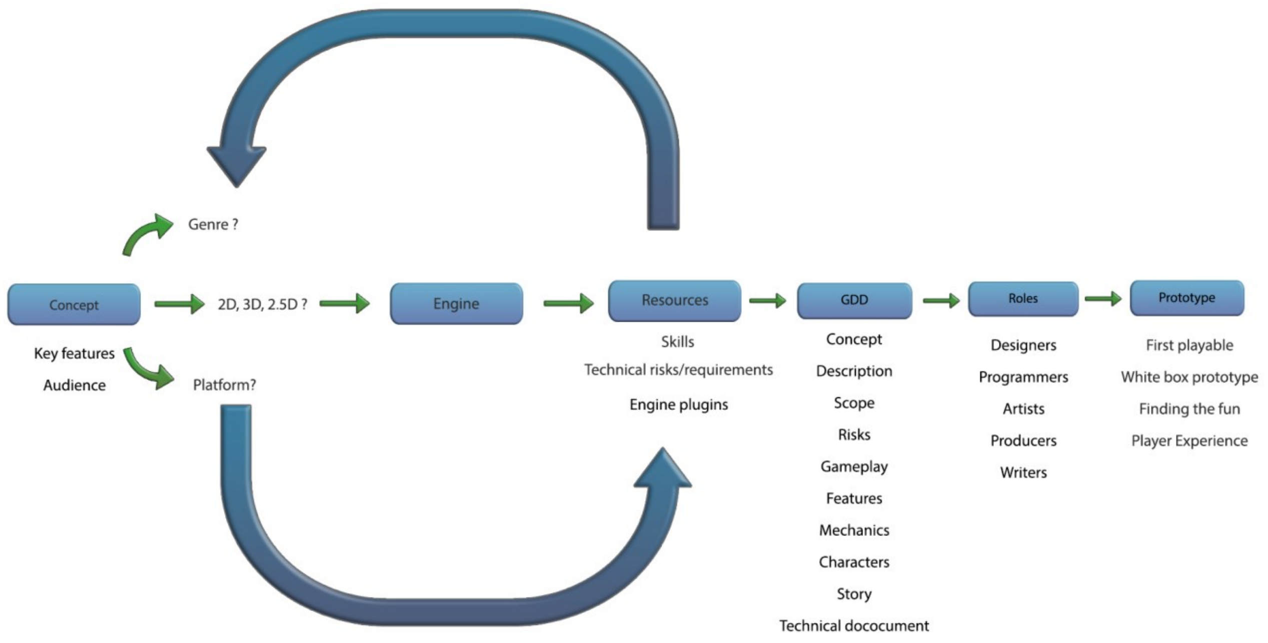


Figure 43. Refined pre-production

After the engine has been picked for the project, the resources for the project should be identified, what kind of skills does everyone have, do they have skills with the engine they picked, do they think they can learn a new engine, are there technical risks that arise from using the engine and does the project need plugins to better build the game? Resourcing is the part that should take time and thought in order to figure out if the scope and concept is realistic in contrast with the resources available. If the resources required are too high, then one should go back and rethink the concept and then come back to resources and see the results.

After the resources have been thought out, it is time for the Game Design Document. In the GDD developers should figure out the general description of their game, concept, features, gameplay and different aspects that depend on the genre and type of game that is being developed. Most importantly it is about giving a clear idea for the team and possibly publisher on what is being done. How big the GDD should be at the beginning really depends on the game and the scope of the game. There were arguments that GDD should not be too specific at the beginning which is what I agree with. The GDD should develop over the projects but should be completed in the early production where it can provide everyone in the team clear scope and focus of what is being done and what kind of tasking they are going to have. Creating a full GDD as early as possible can also create better understanding of what the actual scope is and how much resources are needed.

After the GDD has been created it is time to give the roles for everyone in the project. It is also possible that roles are decided before conceptualization, but specifications for everyone's tasks and role can come after the GDD has been created and the project is understood better.

At the end of the pre-production comes creating the prototype of the game. What the literature and developer talks showed us was that even prototyping is often its own process. Prototyping starts with creating early version that include basic movement and basic interaction, which then develops into more advanced prototypes that are supposed to develop and answer different questions. What is essential about prototyping is finding the fun and the player experience that the game is supposed to give and validate different design ideas. What the developer talks showed was that finding the fun and the experience is not just about the players, it is about the developers and the team too. What was often described is that the way the team can move to production requires that they are happy with what they are creating and they would be happy to move into production with the product.

After pre-production, we will see the refined production phase shown in figure 44. Content creation is about creating the editors and the tools to create aspects of the game, assets are made that consist of the art, animations, music, sound effects etc. that the game needs. Iteration is a crucial part of the process, but even iteration becomes more and more problematic the longer production goes, which is why the production has been split into two phases. Most of the iteration and playtesting should be conducted early as possible and then feedback should be taken in and changes should be made accordingly. Production phase two marks the time when making change becomes more problematic due to the increased size of the project and coding structure, but the process itself stays the same. Testing is not only meant for the game itself but also for the tools that are used for content creation. Everything that for example is used to create levels and other sections of the game needs regular testing and fixes for problems when they are encountered. Game testing refers to Quality assurance and while QA often starts during post-production, it is generally suitable to start early testing and problem validation in the later parts of the production.

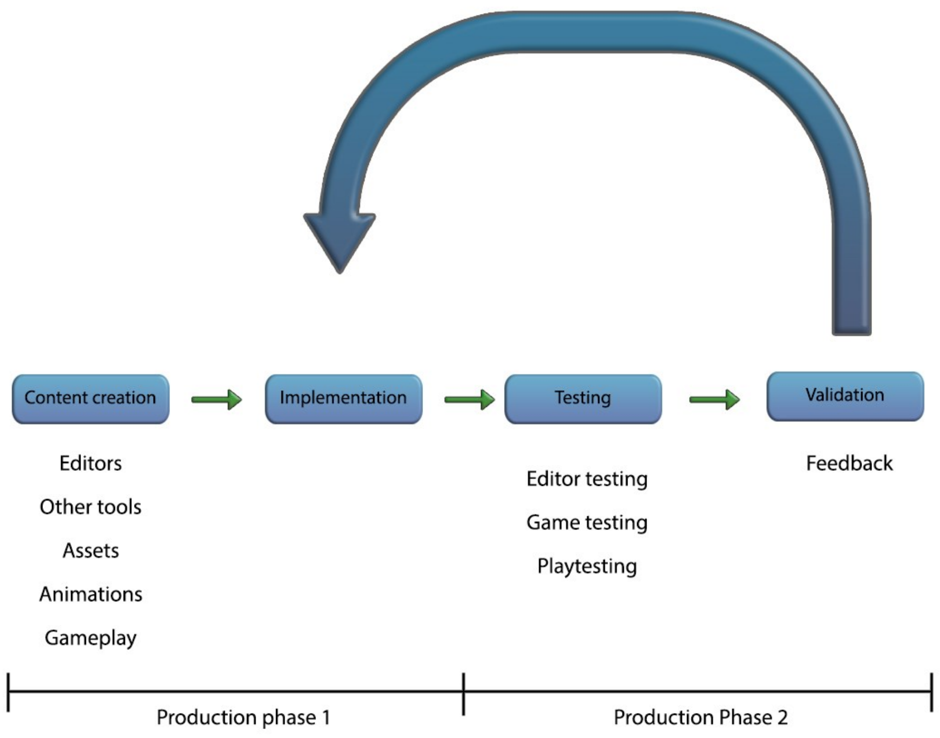


Figure 44. Refined production phase

## 4.12 Difficulties of Game Development

When you look at the game post-mortems, you will quickly notice that every project has issues somewhere down the line. Following and interviewing Dreamloop Games revealed that they have good amount of positive aspects in their development process, but also negative aspects. Table 1. shows problems identified during different development phases for both projects by comparing the theory and what has been observed from the interviews. Few of the positive aspects to notice are also the fact that there are improvements that have happened between the projects such as technical management and knowledge increasing and conceptual improvements. However, the project does still have its own problems, like does any other project and it is probably not a shock that indie developers often have it very rough. Dreamloop does work under pressure coming from insufficient resources that is of course affected by the fact that that they do not have much to offer full salaries and Challengers of Khalea does require lots of resources due to still having high requirements conceptually.

Khalea showed the truth that projects change, requirements change over the time due to different reasons that can be internal, external or both. Dreamloop did not have all the knowledge and tools early in the process. StrangeIoC was only released shortly after their Kickstarter was done and it was clear that if such tool would have been available for the project earlier, they very likely would have implemented it from the start. The team is also still unfamiliar with everything that is to StrangeIOC and they are still learning efficient ways to program by using the framework. Through designing and iteration, they also realized that they would need better tools for implementing animations and puppet master plugin was implemented. Making both these changes took great amount of time and effort and it essentially set the project backward for a while. The engines such as Unity are not perfect and they never will, they do slowly get better and adapt to better fit designers, but additional effort for bigger projects is always needed. These tools will not provide everything to make the game complete and also plugin dependency can be problematic as well.

| **Glory** | **Positive aspects** | **Negative aspects** |
|---|---|---|
| **Pre-production** | Paper prototyping was considered and done | Paper prototyping was not extensive enough |
| | Not writing too extensive GDD | Conceptually GDD was not well thought out |
| | Early prototyping without much commitment | Lack of risk management |
| | Good project management | Overambitious concept |
| | Concept validation by peers | Lack of development methods |
| | | Lack of resources |
| **Challengers of Khalea** | | |
| **Pre-production** | Better conceptualization | Lack of resources |
| | Good project management | Lack of documentation |
| | Good Programming related control | Lack of validation |
| | Good technical management | Lack of player experience validation |
| | Vertical slicing | Concepts change rapidly |
| | | Slow prototyping |
| **Production** | | Desynchronized development |
| | | No playtesting yet |
| | | Jumping into production too early |

Table 1. List of positive and negatives aspects gathered from the study

Reading the literature, interviewing and observations have shown that video game development is undoubtedly very broad field that combines multiple areas of expertise together. Having such a broad field of interests does bring multitude of problems as well. We have conceptual problems ranging from mild mistakes to concepts that can be flawed from the beginning being uninteresting and not necessarily what the players would like to see. Often projects start with lack of resources and those problems still stick throughout the project, but It remains as a common problem for indie developers and possibly also for bigger companies who have even bigger projects. Developers can also often be very impatient, wanting to jump into development very quickly which might lead to lack of conceptualization and into a mindset that they will figure everything out during the development. We also have to manage people, their personalities can become a problem causing clashes between members, keeping everyone focused and on schedule by using different management methods. Development has problems because of technology, even with technology that we already know or think that we know and then the new technology that we desperately try to implement. We put our trust in free engines such as Unity, GameMaker and Unreal but even they do not always do what we want and they do not always function the way we want. Engines such as Unreal at least are free to tinker if they have problems but often engines require us to patiently wait if there is a glaring problem, so these engines are not perfect as you still need high amount of knowledge in order to use them to make high quality and larger scale games.

We often also forget that what we see on the screen is not just assets that are easily placed there, we need tools to make the content and we need those tools to create the vision we have for the game visually but also structurally. Level editors are important tools that we use to create these levels and they often might need separate systems so we can construct these levels the way we want them to work and appear. These tools need their own building, testing and validation and they need to be usable. In order to create better and clearer animations we might need animation specific tools, our game might need abilities that we need to create and design efficiently with proper tools. Tools are often not just for one person to play with, they need to work for everyone and these tools often also carry on to future projects.

Even though design has not been the primary focus here, I do consider that understanding the development process is crucial knowledge for game designers. The way we think in terms of design does not always work, the solution might not be the best for the player even though maybe technologically it might be or resource wise. Designers are often met with difficult questions how will I change something without causing much extra work in different areas? How do these different ideas go together and how can they be implemented efficiently? What will change due to my design in terms of programming, user interface, visuals, story or something else? How will my actions affect the players experience and overall game experience? Designers should consider these problems and in order to better understand these problems, they need to understand the process.

Lot of older games had problems of their own during the development, but they were able to overcome those problems and make the game successful. Game development process will always have problems, but the answer possibly lies in the fact that in order to overcome them we need experienced people and management who thoroughly understand the process and its difficulties and are able to make smart decisions when problems arise, and they eventually will.

# 5. Conclusions

This chapter will present the conclusions from the study by first reviewing the research questions of the study and then presenting the key findings from comparing the case study with the theoretical literature. Additionally, the chapter presents the limitations of the study and ideas for further research.

## 5.1 Summary

Video game development process has been under discussion for years, essentially starting from the early 2000 and slowly the gaming industry has grown rapidly thus creating even more requirements for understanding the whole process.

1. What kind of process is video game development from the perspective of a company?
2. What problems can be found from the company's development process in comparison to the literature?

The purpose of this study was to first of all examine the video game development process from a company's perspective and secondly finding out key problems in the process of the company when compared with the literature. The first question was approached mostly from the game professional's perspective, inspecting game development and game research literature and two Game Developer Conference talks and presentations. Due to general lack of academic articles provided about the whole process, they offered a much smaller perspective on the issue.

As a conclusion, defining game development process was a difficult task because the process can have small changes based on the type of game, people, methods, fields and setbacks. The game professional's literature however did provide excellent information in order to create great picture of the key elements of the process. Generally, the differences between professional literature were not extensive and the process was described similarly, but through inspecting the literature from multiple sources and especially the professional talks, they provided interesting additions to the whole process. The most intriguing parts were prototyping, game design document and iteration. Literature about prototyping and game design document both suggested that they themselves are a process instead of a single type of task during the pre-production. Designer talks brought out the fact that prototyping is a process itself, having from two to four phases depending on the company's approach on the issue. Inspecting the design document also showed that game developers do have different definitions of the contents and preciseness of the game design document. The design documentation has started to stray away from a single big document to a more readable and sensible cluster of separate documents that vary from general concepts to gameplay, narrative and technical documentation. It was also concluded that often developers have the tendency to create very precise documentation, which often ends up in wasted work due to the unpredictable nature of game development. Definition of iteration was straight forward, but what was concluded was the fact that iterative process does extend to other areas such as programming, design, art and other fields as well, making iterative process very extensive area to look at.

In order to answer the second question, case study was conducted with Dreamloop Games. Interviews with the developers were the primary way of inspecting the company's process between years 2014 and 2016 and occasional textual discussions and observations when working in the company. The case study provided great data in order to answer the questions and key problems were found by comparing the case with the literature. The key differences were in prototyping, iteration and the game design document. Prototyping was found to be slow for Dreamloop Games even during pre-production by producing a prototype approximately every six months, when prototyping was supposed to be a quick process. Additionally, paper prototyping by Dreamloop was not very extensive and it could be considered extremely short by only having few testing sessions. In iteration, it was specifically art iteration that stood out from the sources. Art iteration and especially concept art is used as a tool of communication and the iteration is done by changing the concept. Dreamloop strayed from this approach by creating concept art for a specific character, but that concept was immediately changed in the next phase of the character creation process. The game design was conceptually not extensive enough and it focused heavily on the gameplay and mechanics aspects. Furthermore, the documentation did not include precise technical documentation and the aim of the concept at the start was found to be not very realistic. When looking at further problems, resourcing was and has been a problem from the start and in addition to prototyping, iterative problems and lacking game design document the general scope of the project is still passionate and large. An interesting find was also the fact that the project appeared to have de-synchronized development due to resource problems, which could create further problems in the future for the project.

In addition to these primary questions, few more findings were made during the study. Interestingly the game design document was on small part replaced by project management program called Phabricator and its Wikipedia functionality, but large portion of the design documentation has not yet been transferred there. Also, the importance of the design document proved to be large but not necessarily in the beginning. While design documentation should be extensive, it became apparent that it needs to be made extensive early in the production and not in pre-production. Secondly, it was an interesting find that game engines are not hugely discussed topic in the literature. The finding from few sources proved that game engine selection is extremely important during the pre-production due to the differences and restrictions between the free game engines and different types of games might not always be favoured in certain engines. Everything also suggested that creating own engine for the project is not favourable unless resources, knowledge and specific requirements allow this approach.

During the study, it was also concluded that all the three phases of video game development are crucial in their own way as it was presented in the literature. It became an interesting question to ask if a certain phase would stand out with its importance, but making such a conclusion proved to be extremely difficult. All areas shared importance and it was suggested that every single phase should generally be handled with care. It became interesting task to combine all the information gained from the literature, and try to combine them together in order to further specify the process. The conclusion from the refining is that game development process still needs clarifications.

As a final notion, it became clear to conclude why video game development is hard. Due to video game development including such a broad field of expertise, the problems also exponentially increase. Even more successful game projects had their own problems, but they were surpassed as large amount of these projects included multiple different problems during the development.

To summarize these findings, researching and observing game development process has gotten more precise and it is better understood. Due to the growth of the industry, the requirements for understanding the process and also the amount of data has grown substantially, which has led to better literature and understanding of the process. However, due to the differences that happen in each project and due to different aspects of development, the understanding of game development process still has room for growth.

## 5.2 Limits of the Study

The research question was large and extensive for one master's thesis, but the findings have proven to give more descriptive perspective of the whole picture. The limitations of the study arise from this fact that the game development is a very large process to study because it includes so many fields of expertise and the thesis has its own limitations as well. Most focus was obviously put on researching the pre-production and production because the game project in question has not yet seen post-production and most material gathered was for the first two phases. Due to the general length limitations of the thesis itself, such topics as music, animations, communication, project management methods such as Lean and Kanban were left out among series of other topics. For example, music for the project is in its very early phases and most of the music production will come much later down the line. Additionally, other systems and tools such as game save system and SQL database implementation were left out.

## 5.3 Future research

In terms of future research on game development process I hope that slowly research would start to split and separate these processes based on their size and possibly also based on platforms, and would better differentiate the processes and how everything changes through the size of the project and the team. While the diagrams of development processes are often able to capture the essence of the process, I do feel that in order to truly understand the process, they should include more detailed and refined diagrams on the process to reflect it and create better understanding of the development process, not only for researchers but also for starting game developers and gaming communities. Due to the limits of this thesis, prototyping section could have had more theoretical background in terms of programming. Additionally, for future research it would be interesting to look at the prototyping process from the perspective of a programmer to understand if programming should be a quick process and if extensive code refinements are necessary during early prototyping.

## References

Adams, E. (2009). *Fundamentals of game design*. Pearson Education.

AMA with Redhook Studios. (2014)
https://www.reddit.com/r/Games/comments/2030kf/we_are_red_hook_studios_developers_of_darkest/
Referred in November 12th, 2016.

Andruko, Corey & Everman, Dusty (2009). The Iterative Level Design Process of Bioware's MASS EFFECT 2.
http://www.gdcvault.com/play/1324/The-Iterative-Level-Design-Process
Referred in November 5th, 2016.

Areena 5.
https://fi.wikipedia.org/wiki/Areena_5
Referred in September 23rd, 2016.

Bethke, E. (2003). *Game development and production*. Wordware Publishing, Inc..

Cohen, D. S., & Bustamante, S. A. (2012). *Producing games: from business and budgets to creativity and design*. CRC Press.

Craitoiu, Sergiu (2016). Marketing Ideas for your indie game.
http://www.gamasutra.com/blogs/SergiuCraitoiu/20160119/263694/Marketing_ideas_for_your_indie_game.ph
p
Referred in November 15th, 2016.

Deetman, Koen (2014). Un Unreal Experiences Proves to be a Real Unity
http://www.gamasutra.com/blogs/KoenDeetman/20140924/226358/An_Unreal_Experience_Proves_To_Be_A
_Real_Unity.php
Referred in November 10th, 2016.

Donovan, Greg (2015). The Vertical Slice Challenge.
http://www.gdcvault.com/play/1022329/The-Vertical-Slice
Referred in November 7th, 2016.

DellaFave, Robert (2014). Marketing your indie game: The Single Most Important Thing to Learn
https://gamedevelopment.tutsplus.com/articles/marketing-your-indie-game-the-single-most-important-thing-to-
learn--gamedev-7157
Referred in November 16th, 2016.

Dreaming: Seriously Lucasarts…where is Gladius 2?
https://www.destructoid.com/blogs/GrimmTrixX/dreaming-seriously-lucasarts-where-039-s-gladius-2--
225236.phtml
Referred in November 14th, 2016

Fullerton, T. (2014). *Game design workshop: a playcentric approach to creating innovative games*. CRC press.

Gladius
https://en.wikipedia.org/wiki/Gladius_(video_game)
Referred in September 23rd, 2016.

Gregory, J. (2016). *Game engine architecture, Second Edition*. CRC Press.

Hamari, J., & Tuunanen, J. (2014). Player types: A meta-synthesis. *Transactions of the Digital Games Research Association*, *1*(2).
Ismail, Rami (2016). Patch the Process.
http://ramiismail.com/2016/08/patch-the-process/
Referred in November 14th, 2016.

Jacobson, Brian & Speyer, David (2015). Classic Postmortem: The Making of Half-Life 2.
http://www.gamasutra.com/view/news/259479/Classic_Postmortem_The_making_of_HalfLife_2.php
Referred in November 5th, 2016.

Leadbetter, Richard (2015). Among Friends. How Naught Dog Built Uncharted 2.
http://www.eurogamer.net/articles/digitalfoundry-among-friends-how-naughty-dog-built-uncharted-2
Referred in November 14th, 2016.

Janz, Jason (2016). A father, a dying son, and the quest to make the most profound videogame ever.
https://www.wired.com/2016/01/that-dragon-cancer/
Referred in November 12th, 2016.

Jenkov, Jakob (2014). Understanding Dependencies.
http://tutorials.jenkov.com/ood/understanding-dependencies.html
Referred in November 14th, 2016.

Keith, C. (2010). *Agile game development with Scrum*. Pearson Education.

Kissner, Michael (2015). Writing a Game engine from scratch, Part 1: Messaging.
http://www.gamasutra.com/blogs/MichaelKissner/20151027/257369/Writing_a_Game_Engine_from_Scratch_
_Part_1_Messaging.php
Referred in November 12th, 2016.

Kultima, A. (2015, September). Developers' perspectives on iteration in game development. In *Proceedings of the 19th International Academic Mindtrek Conference* (pp. 26-32). ACM.

Lawrence, Nathan (2011). The problems of defining a hardcore gamer.
http://www.pcauthority.com.au/Feature/260885,the-problems-of-defining-a-hardcore-gamer.aspx
Referred on November 12th, 2016.

Lucero, A., Holopainen, J., Ollila, E., Suomela, R., & Karapanos, E. (2013, September). The playful experiences (PLEX) framework as a guide for expert evaluation. In *Proceedings of the 6th International Conference on Designing Pleasurable Products and Interfaces* (pp. 221-230). ACM.

Macklin, C. & Sharp, J. (2016). *Games, Design and Play: A detailed approach to iterative game design.* Pearson Education inc.

Mills, A. J., Durepos, G., & Wiebe, E. (Eds.). (2010). *Encyclopedia of case study research*. Sage Publications.

Moran, Dylan (2016). 5 Leading Game Engines for Indie Developers.
http://www.gamasutra.com/blogs/DylanMoran/20160729/278145/5_Leading_Game_Engines_for_indie_game_developers.php
Referred in November 12th, 2016.

Mayer, Andrew (2016). Making Games Better: The Art and Process of Game Design and Development.

Petrillo, F., Pimenta, M., Trindade, F., & Dietrich, C. (2009). What went wrong? A survey of problems in game development. *Computers in Entertainment (CIE)*, *7*(1), 13.

Quijano, Augusto (2014).The Art of Making Guacamelee! - From Folklore to Finish.
http://www.gdcvault.com/play/1020419/The-Art-of-Making-Guacamelee
Referred in November 4th, 2016.

Rouse III, R. (2010). *Game design: Theory and practice*. Jones & Bartlett Learning.

Sayenko, Alex (2015). How (and why) to write a Great Game Design Document.
https://gamedevelopment.tutsplus.com/articles/how-and-why-to-write-a-great-game-design-document--cms-23545
Referred in November 9th, 2016.

Schell, J. (2008). *The Art of Game Design: A book of lenses*. CRC Press.

Schell, J. (2014). *The Art of Game Design: A book of lenses*. CRC Press.

Sterling, Jim (2015). SLAIN! - Broken Promise.
https://www.youtube.com/watch?v=GCuv4Ip8_6w&t=496s
Referred in November 6th, 2016.

Sylvester, T. (2013). *Designing games: a guide to engineering experiences*. " O'Reilly Media, Inc.".

The 3 Best Video Game Engines. 2016.
http://www.gamedesigning.org/career/video-game-engines/
Referred in November 12th, 2016.

Tynkkynen, Johanna (2013). Supercell CEO and founder Ilkka Paananen: “Finland is a great place to be an entrepreneur”. https://teknet.tek.fi/arkisto.lehti/content/supercell-ceo-and-founder-ilkka-paananen-%E2%80%9Cfinland-great-place-be-entrepreneur%E2%80%9D.html
Referred in November 9th, 2016.