GDD as Development Methodology

Kim J.L. Nevelsteen
Abstract

**GDD as Development Methodology**

*Kim J.L. Nevelsteen*

The Game Design Document is a well defined set of requirements for a game design and a methodology for game development. The most pertinent aspects of the GDD are described and the relative research collected, in order to better the reader's understanding of the concept. This is a work in two parts, theoretical and practical.

Research was an ongoing effort while attempting to produce a game design according to the content prescribed for the GDD. The resultant design was for a fictitious game dubbed Arcane Dominion.

Knowledge gained about the GDD is used to make a comparison and attempt to identify a possible evolution towards existing software engineering methodologies. Some differences are brought to light between the GDD and a popular contemporary methodology, Agile Development.

To compliment the research, one aspect of the designed game was implemented in a prototype and thereafter analyzed in order to obtain needed feedback and be confronted with the complexities of game balancing pertaining to one chosen game strategy.

No knights or soldiers were hurt in the making of this thesis.
Acknowledgements
I would like to thank my thesis supervisor, Prof. Justin Pearson, for his confidence in my work and letting me think out of the box. I would also like to acknowledge Martin Ericsson and Leif Olvång for long discussions allowing me to bounce many ideas off them.

Dedication
I would like to dedicate this work to Mama for her undying support and numerous sacrifices over the years for my higher education and making the writing of this thesis possible; to Martin Wehlou for his seemingly endless wisdom, inspiration, support, teachings and our frustrations since our meeting; to Nonkel Frans, Tante Yolanda, Nonkel Willy and Tante Chris for being my surrogate parents in Belgium; to Prof. Edward Soetens, Prof. Brigitte Verdonk, Prof. Roland Bol, Prof. Ole Glimo, Prof. Annie Cuyp, Prof. Frans Arickx and Prof. Konstantinos Sagonas, the mark worthy professors; to Rina Hillen, Ulrika Jaresund, and Lydia Janssens for dealing with my administration nightmare over the years; to my brother David, for being my childhood idol; to Hania Uscka for being my mathematics guru; to my father for lightening the financial burden; and definitely not least, all my friends for making my university life interesting enough to relive life as a poor student.
GDD as Development Methodology

(part 1 of 2)

June 15, 2008

Creation: 2007-11-30, Revision: 221
## Contents

1 Introduction 5

2 Game Design Document 5
   2.1 Purpose and Content .......................... 5
   2.2 Design using the GDD .......................... 6
   2.3 A Living Document .............................. 8

3 Comparison with Software Engineering 9

4 Prototyping 13

5 Game Balancing 13

6 Analysis 15
   6.1 GDD: Arcane Dominion .......................... 15
   6.2 Prototype: The Build Process .................... 16
   6.3 Game Balance: First Defensive Unit ............... 21

7 Conclusion 26

A Source Code 27
   A.1 A Prototype in Prolog .......................... 27
1 Introduction

In comparison to software engineering, video game development lags behind with respect to formal development methodologies. The reason for this is largely due to the creative nature of game development and past tendency for games to have been developed by programmers who fulfilled multiple roles in development, without any formal methodology. Video games are essentially only 25 years old and as the entertainment software industry matures, more formalization can be expected [17]. Due to the serious growth of the game industry, different development roles are being split into specific employment positions. Where the game developer of the past also possibly filled the role of designer, producer, artist and programmer, contemporary teams consist of individuals with a more specialized skill set [17].

Two roles of particular importance to this paper are that of the visionary and of the game designer (who’s role is sometimes filled by the producer). They are responsible for initially creating and maintaining a document called the Game Design Document, a document paired with a non-standardized methodology frequently employed in game creation.

Use of the Game Design Document, often noted as simply GDD which will be used throughout the remainder of this text, is only now being widely adopted by major companies in the gaming industry¹, which contributes to the fact that the methodology has not been standardized.

2 Game Design Document

2.1 Purpose and Content

The content written in the GDD can have many intended audiences. It is not uncommon for a game designer to write different versions or incorporate different sections of the same design in a single design document to address various audiences. A document with the sole purpose of convincing management of success will differ and bring to light other aspects of the game than a document intended for public relations for advertising purposes [5].

“Design documents tend to fall into one of two formats: they either loosely describe a game concept so upper-management can sign off on the idea, then get dropped into someone’s filing cabinet never to be read again or they are the size of your local Yellow Pages, filled with every imaginable detail that no-one but the person who wrote it cares about or is willing to read.” [11]

In essence, the GDD serves as a communication medium between audiences.

“A complex project involves many talented people. Skilled programmers and artists tend to have minds of their own. While you intend to create a horse, the artist may be envisioning a unicorn and the programmer a highly efficient camel.” [9]

¹Even Blizzard has proclaimed not using the GDD in the past, contrary to the present [18].
Due to its lengthy detailed nature, people will not likely be inclined to read the document [11, 2]. In order to stimulate readers, it is important that the GDD is presented in a clear and readable fashion for each particular audience [5] and attractive, if possible [17]. Note, the format considered readable might vary greatly between audiences.

Discovering content the GDD should contain is not difficult research. There are many references, each with their own list on what should be present in the GDD. Most state that the material is relative to the type of game being produced and give some brief examples. This can be expected given some games have very little in common. The following is a short list of possible content for the GDD [1] . . .


- Game Descriptions: an over all game concept description, roles of the players in the game, proposed gameplay modes, genre of the game, description of the target audience, competition modes, description of the game world, licensing, and the target platform the game will run on.

- Game Aspects: the characters, the storytelling and narrative, the user experience including the user interface, the level design and, the two most important, the gameplay and the core mechanics.

Something rather peculiar is that design documents contain an estimated production cost for the game. It is uncertain if it is a task deemed for the designer to make this estimate together with the design. If it is not the designer’s responsibility, who’s responsibility is it?

### 2.2 Design using the GDD

Because the GDD needs to describe each aspect of the game in detail, the designer must be able to logically order document content so that nothing is overlooked. Structuring the document correctly to allow this proves difficult [2]. Structure can be achieved through different approaches, such as either listing all the game’s responses relative to all possible player actions or by documenting each game token. Several references state they prefer a “feature-oriented” document [11, 5] which happens to lend well towards versioning.

> “Many of the attempts to define games have focused on the mechanistic elements of the game, such as the primitive actions that the system allows the player to perform or the tokens that the player manipulates. The approach has been to treat games as self contained logical system.” [7]

The Ultimate Game Design is the ultimate imaginative creation of the designer. It contains a plethora of ideas which gives the game enormous depth. Problem with the Ultimate Game Design is that it requires an infinite amount of resources. In other words, the project team that attempts to implement the Ultimate Game Design will spend an unacceptable amount of time creating a
first release of the game or even a working prototype. This poses a problem for those seeking a return on investment for the project since the game’s success remains unknown during the development process. Also, because of the lengthy development process, other producers might have produced a similar product in a shorter amount of time rendering the ultimate game partially or fully obsolete. Setting short term goals by defining a feature set for progressive versions avoids the Ultimate Game Design and is the basis for iterative development.

“My notebook is filled with feverish schemes that add layer upon layer to the original concept, buildinmonstrosity” monstrosity (sic) called The Ultimate Game Design. Stellar mechanics, physics, fusion weapons, special encounters, crazy plot twists! Just thinking about it gives me shivers of pure intellectual pleasure.” [7]

At the start of the project, according to Rollings and Morris [17], the visionary or designer should create the Designer’s Notes, which contains all brainstorming and wishes for the game, and the Gameplay Specs which contains the initial design to be implemented in the first iteration of development. The initial design is an important basic feature set which determines when a minimal working prototype will be available providing proof of concept.

“To achieve conceptual integrity, a design must proceed from one mind or a small group of agreeing minds.” [2]

According to the Spiral Model, to which the GDD resembles closely, each iteration or tier of the development process updates the GDD [17] defining another feature set to be implemented in the next and successive versions of the game. Having distinct versions or milestones in the development provides the following benefits [4].

- Having a minimal working prototype provides proof of concept with a minimal time investment, which may help in getting project approval, if needed.
- Shorter feedback cycles allow for adjustments in the design reducing the risk of creating concepts that are technically impossible or not entertaining.
- Through having a working prototype rather than just a thick design specification to represent the game, improved marketability can be gained.

Testing the prototype produced at each iteration will provide feedback indicating problems in the design. This will lead to changes in the GDD through adding, removing and modifying features. These changes are tracked by the team or by an individual responsible for the maintenance.

Since the GDD must touch upon all game aspects without becoming a size comparable to that of the Yellow Pages, how large can the GDD be expected to become and how much time needs to be allotted for its creation? Because game designs vary greatly, only rough estimates to these questions can be given. Two independent sources state [4, 13] that initial designs can run from 20 pages, for a small design, to 60 pages, for a large and detailed design. By the end of the design process, after many iterations, a complete game design is noted to become anywhere between 75 and 200 pages or more; with a time scale of 2 or 3 weeks for a team or 150 to 400 hours for a lead designer.
2.3 A Living Document

The GDD itself must be under revision control so that design changes can be tracked and document input from different people can be merged and maintained. Discussions pertaining to features don’t necessarily need version control but having a record could prove beneficial. Version control is also particularly handy when it has been decided that a feature is to be removed. Either the feature can be left in the document with a clear indication that it is obsolete and why or the version control system can be used to track deleted text, the latter possibly providing more clarity.

If the GDD is to be a living document and the primary purpose to allow for communication between different parties, then the right medium for documenting must be chosen to facilitate this. In the last decade we have seen the rise of many different technologies which might be suitable. Three prime examples of this are a blog, a wiki and a forum. Each has its advantages and disadvantages. It is uncertain, however, if any of these is the ideal tool. The following is a list of some of the characteristics that are beneficial for such a tool to support [5].

- Ease of authoring: If the GDD is to stay current during the development, the medium that holds the GDD must lend well to authoring. If making a change in the GDD requires too much time or effort, the GDD will most likely be neglected. The rationale, “I shall fill in the documentation later.”, might be used leading to obsolete and neglected documentation.

- Linearity: There is some contradiction in this characteristic. The GDD must at the same time be a narrative (linear) and allow for hyperlinking between text. This is one domain where a blog, a wiki or a forum might excel for the GDD. Each section of the document should be discussable by the development team, preferably through the same medium that holds the GDD, so that it is also documented. When a conclusion has been drawn on a particular issue it should be noted and finalized, so that others need not search through long heated discussions to find results. The collection of all finalized issues should be ordered and will constitute the linear GDD. Hyperlinking should be possible within the text, but should not take away the possibility for the text to be called upon in a linear fashion where important concepts or features are grouped together. All this might, of course, be asking too much.

- One voice: Ideally the GDD would be written and maintained by a single author, promoting narration and coherency. This can be achieved through having all finalized text collected by one individual, revised and placed in the actual GDD [2].

- Linking and tagging: Along side hyperlinking capabilities, the medium for the GDD should allow for external direct linking as well. A direct link should be available to any section in the GDD allowing for the links to be communicated through a variety of other mediums. Auto-linking (automatically setting a link for a known key word) is very useful to link semiotics to their definition. Semantic labels (tags) can also be set on different sections of text to help developers search or gather information pertaining to a certain subject.
One technology of particular interest which might have a slight edge in comparison to others, is the wiki. It allows for multiple editors to edit the same document and hyperlink to other texts within the wiki and external sites. But, the most obvious advantage is its built in revision control, also known as source or version control.

3 Comparison with Software Engineering

In the past, video game development was carried out by a group of individuals with diverse skill sets, creating emotionally and visually satisfying products [17, 6]. Due to its immense growth, the game industry has acquired the talents of individuals with specialized skills, each in a particular area. Teams with one hundred people are no longer an oddity. To guide these teams of individuals production methodologies are needed. What could be easier than to borrow the methodologies from another domain of development.

There is a tendency to interchange the concepts GDD and game design, however, care must be taken. Game design can be somewhat of a misnomer or simply be used to signify a wide scope of the game development process. In order to compare with software engineering, the concept of game design must be broken down. Game design is often used to describe what in software engineering is formulated as two concepts. Requirement analysis is the act of determining exactly what the product must do, which is often paired with a use case specification. Aside from this, it must be decided exactly which technologies will be combined to create the product, the so called system architecture. Both of these concepts seem to be collected into the term game design. The GDD can be compared to the requirements specification with the exception that a development methodology is paired with it. In terms of the GDD, the requirements specification is what is determined at each iteration in the development process.

It is possible to approach a requirement analysis from different perspectives, the most well known in software engineering being that of the use cases. A comparable technique in game development would be that of the Scenario-Based Design. Both attempt to play out how a product shall be used in order to guarantee that each case is handled appropriately in the design.
Since the primary focus of game development is entertainment, the problem domain differs from that of software development. Although a game might be a technical marvel, it may not necessarily be entertaining. This affects development in two major ways; it is desirable to have a working prototype as soon as possible in order to obtain needed feedback to whether the prototype is actually entertaining and psychology plays a very large role in determining the game design. How does one measure a requirement when the goal is subjective? The best game designers will take into account the psychological model of the player, but it can not be discounted that this will influence the methodology of game design.

“Mechanics and aesthetics are certainly important pieces of any model of game design, but in the end, such analysis provides little insight into what makes a game enjoyable. You end up with a set of fragmented pieces that tell you almost nothing about the meaningful interactions between the game as a simulation and the player as an active and evolving participant. Games are not mathematical systems. They are systems that always have a human being, full of desires, excitement and immense cleverness, sitting smack dab in the center. To accurately describe games, we need a working psychological model of the player.” [7]

Figure 1: Spiral model[19]
To compensate for the Waterfall Model’s inability to cope with rapid changes in project demands an iterative development process was introduced. The Spiral Model (Figure 1), which resembles game development using the GDD [17], is an iteratively based model where each iteration is considered one pass through the Waterfall Model. The Spiral Model does not promote a fixed project specification, but rather an iterative one resembling the GDD, and is primarily used for risk management and prototyping [19]. This allows for quick addressing of critical points early in the project.

“Ultimately, game design is a heavily iterative process that involves frequent prototyping and an evolution of crude game mechanics into a reasonable form.” [4]

Adams[1] describes the game development process to have the Concept, Elaboration and Tuning stages. The Rational Unified Process (RUP), which is the most elaborated and well documented Unified Process derivative, defines similar stages, but counts four stages namely, the Inception, Elaboration, Construction and Transition stages.

The Inception stage is notably short and parallels the Concept stage of game development, which can best be kept concise [13] so that prototyping can begin early. At the end of the Inception stage, the basic requirement set must be complete. RUP clearly defines the end of the Inception phase as a critical point to determine whether the project should be developed or not. At this point there should only be a marginal loss if the project is cancelled. The game development process doesn’t state the like explicitly, but rather describes this in terms of cost determination and other factors. The next two stages of the RUP are captured in one phase of the game development process. The reason being that RUP is defined for projects of large robust software applications where such rigorous formal definition is needed. RUP has an Elaboration stage which promotes continual prototyping and refinement of the product requirements followed by a Construction stage when the actual product is produced. In game development only the Elaboration stage is defined. Both of these last phases refine/tune and test the product before release, however, multiple releases of the game are internal to the company, for Beta testing, instead of for a client. In the past, games were smaller and disposable with few public releases or updates. Contemporary games are increasingly large with long life spans meaning many releases to the public. With the advent of massive online games, the disparity between games development and software engineering can be expected to diminish further [17].
Agile software development techniques [14] are widely used today and if a comparison is made with game development, the difference with respect to the project requirements is the most obvious. Agile software development places far less emphasis on requirements in the first iteration and welcomes changing requirements even late in development. Agile development is centered around the customer or client, working closely with the client throughout the project. Delivery of working software to the client is presented on short intervals. These principles make Agile software very awkward for game development where the client is lacking and the visionary holds the idea for the finished product [17].

With respect to documentation, Agile Programming doesn’t lie far from the GDD. Documentation is urged to be kept short and concise, but emphasis is placed on conveying information face-to-face. The GDD should not become bloated either, but face-to-face communication of information might prove difficult in games development when teams responsible for very different aspects of the game are not located in the same place. It is also rather comical that Agile programming promotes sustainable development; the maintaining of a constant pace indefinitely. The so called “crunch time” of games development certainly proves to break this principle firmly. The last difference with Agile programming is based on how teams are organized. Agile programming teams promote the sharing of responsibilities amongst a self-organizing team. This might be a difficult principle to follow for a game programming team where individuals might have highly specialized talents.

The acronym SDD seems to be associated with three obscure concepts. The Internet only hints to the existence of the System Design Document and Software Design Document; the latter possibly correlating to the GDD, but nothing substantial was found. It is possible that a software development methodology has appeared based on the success of the GDD, but this is only speculation. IEEE publications exist with a formal explanations of documents describing software entities in detail under the name Software Design Descriptions [12], but the descriptions are explicitly stated not to be paired with a development methodology and therefore show little resemblance with the GDD.

Cost appraisal is a part of both software engineering and game development. In the process of software engineering, different roles have been identified to contribute to the product design. The software engineering process might be guided by the visionary or customer, promoted and advertised by marketing, and designed by a systems architect who would also be responsible for a cost analysis of the different system components. In game development, these roles seem to remain rolled into one individual with the GDD containing a cost appraisal.

Software engineering and game development share beneficial aspects which are the driving force behind product creation. It is rather comical that they also seem to share a same negative aspect. In both disciplines there is a tendency for the designer to be lead astray and attempt to incorporate too much into the initial design. In game design it is sometimes referred to as creating The Ultimate Game Design, whereas in software engineering it is often referred to as Feature Creep. Feature Creep is the slow addition of features to what would be the most basic design until the design is comparable to the Ultimate Game Design. The end effect is the same, a bloated design.
4 Prototyping

According to the Adams[1], prototyping is done primarily in the Elaboration stage. Whether the prototype is of the entire system or a partial implementations of features in a working product, the prototype must be gradually replaced by the end product.

“The Stub Game Mechanic: A simple, easily implemented placeholder mechanic that fills in holes in the game design and lets you play the game. It is generally not well balanced, nor is it the ultimate desired system. Its key benefit is that it lets you get a working iteration of the game play up and running quickly.” [4]

The lead game designer, possibly together with a team, produces the initial draft of the GDD [17]. The designer(s) can determine the rules that govern the game, however, due to the fact that the design simulates a working model, it is extremely difficult for someone to be able to predict the exact outcome of the model when so many variables are present. After the design team has created the initial GDD, implementors can start a prototype that can serve to test the rules of the game. Prototyping allows makers to test how the rules behave in the simulation to see if they are sound. The more the design can be tested, the more uncertainties can be eliminated early on in development.

“Much of gameplay is emergent, in that you don’t know quite how all those rules will work in practice until you try them out.” [17]

One extreme uncertainty in game development is whether the game shall be fun and rewarding to the player [17]. This uncertainty must be removed quickly. Since entertainment is a subjective matter, a working prototype provides needed feedback in order to ascertain this non-technical property.

“The greater the interaction between the computer and the user, the greater the benefit is that can be obtained from building a quick system and letting the user play with it.” [8]

5 Game Balancing

“In one infamous example, an early version of Starcraft had what was called a “Zerg problem.” In that game, the Zerg Guardians were devastating to anything on land, but they were vulnerable to air units. In order to get Zerg Guardians, however, players had to have already researched Zerg Mutalisk units. Mutalisks could act as air cover for Guardians, meaning that a combined Mutalisk/Guardian force pretty much ran over everything in its path. The imbalance was quickly addressed, but if hadn’t been, the “Zerg problem” might have killed the game.” [16]

Game Balancing remains one of the hardest problems in game development. Game balancing is also done iteratively and continually in the development process. Each new version brings with it a new sets of rules which must be rigorously tested. Each alternation in the rules ripples through the model creating,
possibly adverse, effects. Game balance includes: avoiding dominant strategies, including the roles of chance properly, making player-vs-player fair, making worlds persistent and managing difficulty. Good game balance makes fair competition between players possible. The epitome of this is Starcraft; so renown for its balanced game play that it is used in competitions in South Korea[15].

Not a lot of documentation exists pertaining to Game Balancing. In the documentation that does exist, different game aspects, such as fairness and difficulty, are discussed, but concrete approaches for achieving balance are lacking. Obscurity in the subject is most likely due to the nature of a game having a unique combination of elements that need to be balanced. If the game is symmetric then mathematical analysis of game elements proves particularly beneficial [3].

Mathematical analysis can be used to attempt to calculate, given values on a particular scale for each of the different game units, how the game model might behave over a period of time. Although it is theoretically possible to calculate this behavior, in practice the results of mathematical analysis are highly interpretable depending on which play strategy that is employed.

“The real art of balancing comes in watching how people actually play.” [16]

Although a lot of skepticism exists that a generic system can be created to compute game balance, one company advocates that they have done exactly this and offer the product for sale [20].

Mathematical analysis might prove very handy for symmetrical games, but some games are not symmetric. In order to balance these games a different tactic must be employed. A popular technique to ensure game balance is by using an intransitive relationship between elements, commonly referred to as Rock-Paper-Scissors.

Intransitive relationships achieve game balance by ensuring no dominant strategy exists. This means there is no there is no dominant strategy to defend with and also none to win with. A player choosing a move randomly has an equal amount of chance to win as a player choosing the same move repeatedly. Only predictability is penalized; if an opponent realizes a player is predictable a winning counter move can be played. If more than three game elements are involved in an intransitive relationship or the cost and reward of each move is not equal, it might be useful to calculate an interaction matrix to obtain game balance [17].
6 Analysis

6.1 GDD: Arcane Dominion

The basic concept of for my design is not new. After perhaps the game Dune II from Westwood Studios many other game titles followed hallmarking the genre real-time strategy games. That these games having moved into an Internet browser setting is rather new. For my design, I used a real-time strategy game called Tribal Wars [10] as the basis for my design. The game’s design, in my opinion, is seriously flawed which gave me the inclination to attempt a more professional design.

My thesis revolves around the concept of using the Game Design Document. I have a background in software engineering, which makes iterative development in a team using source control a known concept. I was completely oblivious to how to create the initial draft of the GDD, the Gameplay Spec, and the specifics of game design. In order to start my research properly, I began writing the GDD without any prior knowledge of how it should be structured; only having read one source on what the content of the GDD should be. This approach allowed me to see what methodology I would devise and what pitfalls I would encounter while working isolated.

Through attempting to create a GDD for a fictitious game, I dubbed Arcane Dominion, with only the knowledge of what should be contained in the document, it was interesting to try and convey brainstorm ideas into a concrete game design. The entire part two of this document is the resulting Game Design Document which I created for Arcane Dominion. I stopped writing the design when the task felt tedious, I had surpassed the concept of learning design and faced just filling in details of specific game features. It was only then that I discovered a case study about testing a blog as medium for the GDD [4]. The case study reinforced conclusions I had already drawn about the writing the design.

Even though the objective was clear, it was difficult to know which approach to take in order to maintain some kind of mental image of what had been discussed and what still needed to be done. This coincides with research stating it is difficult to structure the GDD properly. At first, I contemplated discussing the flow of events a player would go through in the game, but since the direction in Arcane Dominion is not linear, this did not bode well. I fell into using the “feature-oriented” approach simply because it seemed like the natural way to handle the design. Perhaps, my previous experience in software engineering influenced my choice.

Different concepts of the game intertwined raising the question of where to place information in the document. Yet again, this concurs what the research has stated about the desirability of hyperlinking. I have gotten the impression that the medium chosen is very important for the GDD. A flat file does not lend the flexibility needed.

One very brutal mistake I made during the design was the lack of defining feature sets to be implemented at each iteration. I was on the way to making the classical error of designing The Ultimate Game! The thought of versioning did cross my mind, but entirely too late.

In creating the GDD, I also sorely underestimated the amount of work that the design would require. This was compounded by the fact that I didn’t set
milestones in the project. I was under the impression that it was a reasonably sized task, manageable by a single individual. The dragging of the design sent me searching for a time estimate for an initial and completed GDD for a typical game. The results found and I mentioned previously were rather surprising.

After a long period of writing the design, I started to notice that it was difficult to describe certain features in detail further. It was hard to imagine with any certainty what implications design decisions would have on the later stages of the game. This also indicted that milestones were lacking, but more importantly I lacked a prototype of basic feature in order to test how successive features might behave.

6.2 Prototype: The Build Process

My previous experience playing real-time strategy games and the availability for real-time strategy games at my disposal, gave me a consider amount of knowledge for designing Arcane Dominion. However, in order to test new or changed concepts in the design a prototype was needed. At the basis of real-time strategy games is the build sequence of all the buildings and units. In the design of Arcane Dominion, I have devised a modified version of the classical build process. It is this modified version that I chose to implement as prototype.

To briefly recapitulate the workings of the build process specified in the GDD of Arcane Dominion; the player is given a build queue with slots which he/she may fill with a number of optional buildings. Each building type has a minimum build time and prerequisites which must be fulfilled before it can be built. The player is slowly charged resources as the building is being constructed. If at any point in the build process, the player’s resources are inadequate, the build process is briefly stalled until resources are available again. The minimum build time for a building can be divided amongst a number of workers in the BUILD_CREW, but I shall disregard this feature for now. A player can choose to build Quarries which increase the acquisition of resources.

There are two ways I could have approached the implementation of the build queue for Arcane Dominion. I chose to derive a mathematical formula for the behavior of the build queue, but I could have implemented the behavior of the queue programmatically in an algorithm.
I devised a formula which could be used to calculate the outcome of the build process after a number of builds. Implementing the formula in Prolog allowed me to combine prototyping and game balancing, by making use of Prolog’s backtracking capabilities.

Below are the devised formulas for the build process, with the top level formula being the formula for the total time $t_t$ it takes to build the building $r$ and all preceding buildings.\(^2\) The top level formula makes use of all the other component formulas.

$$
t_t = \sum_i t_i, \quad t_0 = 0
$$

$$
t_r = \max\{bt_r, rt_r\}, \quad bt_r = \frac{tq_r}{\text{BUILD CREW}}
$$

$$
rt_r = \frac{rq_r - rsv_r - 1}{\text{gain}(l)}, \quad rsv_0 = \text{START GOLD}
$$

$$
rsv_r = \left( \max\{bt_r, rt_r\} - \frac{rq_r - rsv_r - 1}{\text{gain}(l)} \right) \times \text{gain}(l)
$$

$$
rsv_r = (t_r - rt_r) \times \text{gain}(l)
$$

There exists an extreme case for the formulas above, namely the start condition. When the player has no Quarries in play, the gain per hour is zero. Since $rt_r$ divides by $\text{gain}(l)$, which would be equal to zero, the result is undefined. To handle this extreme case I rework some of the formulas and define some limiting conditions.

$$\text{if } \text{gain}(l) = 0 : \quad rt_r = \begin{cases} 
(rq_r - rsv_{r-1}) & \text{if } (rq_r - rsv_{r-1}) \leq 0 \\
\infty & \text{if } (rq_r - rsv_{r-1}) > 0 
\end{cases}
$$

$$
rsv_r = \max\{bt_r, rt_r\} \times \text{gain}(l) - (rq_r - rsv_{r-1})
$$

$$
= 0 - (rq_r - rsv_{r-1}) = rsv_{r-1} - rq_r
$$

\(^2\)Please refer to the Game Design Document for the semiotics pertaining to the game, i.e. the constants that are typewriter typed such as BUILD CREW and START GOLD.
To test the formulas above, I chose values for each of the needed constants and then calculated a few recursions to see how the formulas progressed. This quickly became tedious and inspired me to implement the formulas to automate the process.

\[
\begin{align*}
\text{rq}_{\text{quarry}}(1) &= 100G \\
\text{tq}_{\text{quarry}}(1) &= 8m = 480s \\
\text{rq}_{\text{center}}(1) &= 400G \\
\text{tq}_{\text{center}}(1) &= 23m = 1380s \\
\text{rq}_{\text{yard}}(1) &= 300G \\
\text{tq}_{\text{yard}}(1) &= 14m = 840s
\end{align*}
\]

\[
\begin{align*}
gain(0) &= 0G/hr = 0G/s \\
gain(1) &= 60G/hr = 1G/s \\
gain(2) &= 70G/hr = 1.16G/s \\
gain(3) &= 81G/hr = 1.35G/s \\
gain(4) &= 94G/hr = 1.56G/s
\end{align*}
\]

\[
\begin{align*}
r &= 1 = \text{quarry}(1), \quad \text{Test: START\_GOLD} = 0, \ gain(0) = 0 ! \\
b_{t1} &= 480s/1, \quad \text{let BUILD\_CREW} = 1 \\
r_{t1} &= (100G - 0G)/0G/s = 100/0 = \infty, \quad \text{gain(0) = 0} \\
t_1 &= \max\{480s, \infty\} = \infty \\
r_{sv1} &= 0G - 100G = -100G
\end{align*}
\]

\[
\begin{align*}
r &= 1 = \text{quarry}(1) \\
b_{t1} &= 480s/1, \quad \text{let BUILD\_CREW} = 1 \\
r_{t1} &= (100G - 100G)/0G/s = 0/0 = 0s, \quad \text{gain(0) = 0} \\
t_1 &= \max\{480s, 0s\} = 480s \\
r_{sv1} &= 100G - 100G = 0G
\end{align*}
\]

\[
\begin{align*}
r &= 2 = \text{center}(1) \\
b_{t2} &= 1380s/1 \\
r_{t2} &= (400G - 0G)/1G/s = 400s, \quad \text{gain(1) = 1} \\
t_2 &= \max\{1380s, 400s\} = 1380s \\
r_{sv2} &= (1380 - 400s) \times 1G/s = 980G
\end{align*}
\]

\[
\begin{align*}
r &= 3 = \text{yard}(1) \\
b_{t3} &= 840s/1 \\
r_{t3} &= (300G - 980G)/1G/s = -680s \\
t_3 &= \max\{840s, -680s\} = 840s \\
r_{sv3} &= (840s - 680s) \times 1G/s = 1520G
\end{align*}
\]
Prototype Analysis...

Below are several Prolog listings. In each listing build sequences are shown in brackets with the build order going from left to right. Immediately after the equal sign the total build time $tt_r$ is shown with resource reserve displayed in between parenthesis. A build time of infinity means the gain is zero and there is not enough reserve to start building the sequence. In later listings, I reduce building names to a single letter for clarity; since build sequences can become quite long. The four DIVISIONS are marked with abbreviated terms, Rgm, Mag, Mil and Eng, denoting Regime, Magic, Military and Engineering, respectively.

Before building prerequisites were implemented, the prototype already revealed a problem with the initial chosen values. Using the gain(l) and START_GOLD values above revealed that gain values were excessive. Listing 1 shows all buildings being built within the same amount of time with only reserve resources varying. It should be noted that there is an implied restriction present, forcing the player to build a Quarry first before any other building.

1. gain(0.0).
2. gain(1.1.0).
3. gain(2.1.1).
4. gain(3.1.3).
5. gain(4.1.6).

6. \texttt{nb_setval(rsv,300)}.

[center, quarry, quarry, quarry, yard]= infinity (200)
[center, quarry, quarry, yard, quarry]= infinity (200)
7. [center, quarry, yard, quarry, quarry]= infinity (200)
[center, yard, quarry, quarry, quarry]= infinity (200)
[quarry, center, quarry, quarry, yard]= 3660 (2680.0)
8. [quarry, center, quarry, yard, quarry]= 3660 (2512.0)
[quarry, center, yard, quarry, quarry]= 3660 (2428.0)
[quarry, center, yard, quarry, yard]= 3660 (2818.0)
[quarry, center, yard, yard, quarry]= 3660 (2428.0)
9. [quarry, center, center, quarry, yard]= 3660 (2650.0)
[quarry, center, center, yard, quarry]= 3660 (2650.0)
[quarry, center, center, yard, yard]= 3660 (2842.0)
[quarry, center, center, yard, center]= 3660 (2926.0)
[quarry, center, center, center, yard]= 3660 (2926.0)
[quarry, center, center, center, center]= 3660 (2842.0)
10. [quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)
[quarry, center, center, center, center]= infinity (200)

Listing 1: initial prototype

The next prototype iteration has lowered gain values. I decided to use Fibonacci numbers divided by ten for successive gain levels and set the initial Quarry level to one, removing the implied restriction of having to build a Quarry first. In order to vary resource gain, I allow for more than one Quarry to be created to give the option of increasing resource gain. Listing 2 reveals an interesting characteristic of the chosen model. Because a CITY Center is resource expensive, it is advantageous to build many Quarries early in the build sequence in order to gain resources faster towards the purchase of the CITY Center. Listing 2 shows the build sequence with all the Quarries built first having the lowest overall build time and the most resources in reserve. This hints to the existence of a saturated system where resources are over abundant. I search for this saturation point below.
With the prerequisites implemented the lowest build time of 3993.33 produced by the build sequence [f, q, q, q, y, c] is no longer valid. The lowest build time, as can be seen in listing 3, is now again the sequence with all the Quarries built first. The implied restriction forcing the player to build a single Quarry first is again clearly visible.

To exaggerate the effect of having the Quarries early in the build sequence and to ensure the system is sound, I configured the system with a single Quarry already in play and a second optional Quarry at the very end of the build sequence. Through Prolog’s backtracking, all the different permutations of the build queue are evaluated. Listing 4 shows the optional Quarry gradually move towards the beginning of the build sequence paired with a lower overall build time.
Having completed the build queue prototype, the lowest possible total build time for each DIVISION to build one of each building with the given constants and one Quarry is shown in listing 5. To reduce complexity and program run time I have computed the common build times separately and they must, therefore, be added to each of the DIVISION total build times. How each of the total build times relate to each other is dealt with in Game Balancing.

Listing 5: lowest total build times for each DIVISION

<table>
<thead>
<tr>
<th>DIVISION</th>
<th>Total Build Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comm</td>
<td>20580.0</td>
</tr>
<tr>
<td>Rgm</td>
<td>64000.0</td>
</tr>
<tr>
<td>Mag</td>
<td>96000.0</td>
</tr>
<tr>
<td>Mil</td>
<td>39000.0</td>
</tr>
<tr>
<td>Eng</td>
<td>22800.0</td>
</tr>
</tbody>
</table>

6.3 Game Balance: First Defensive Unit

In order to ensure that each player has an equal chance to succeed in Arcane Dominion, it is important to make sure build times do not give one DIVISION guaranteed tactical advantage over the others.

As I have pointed out above, Quarries play an important role in the build process. It was suggested by previous results that a point of resource saturation could exist. The first task is to determine if such a point exists and if so, at which point. For testing, I have chosen the DIVISION with the largest overall build time, Magic. In listing 6 the number of Quarries is printed on the beginning of each line, with 2+1 meaning two Quarries immediately configured in play and one optional quarry available in the build sequence. Each successive run of the simulation results in a lower total build time until a total of six Quarries are in play, at which point the result is the same as the previous run. At this point, minimum build times have been reached. The build queue is designed to keep constructing buildings only if resources are available. Each building has a base BUILD_TIME time associated with it. If resources are abundant, the minimum build time for a building defaults to the base BUILD_TIME for the building. In listing 6, the run with five total Quarries in play shows that the optional Quarry is still entirely needed since it finds itself at the very beginning of the build sequence. Six Quarries in play results in a saturated system suggesting five could be the point of saturation.

Listing 6: resource saturation point

<table>
<thead>
<tr>
<th>Quarries</th>
<th>Total Build Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>96000.0</td>
</tr>
<tr>
<td>1+1</td>
<td>48740.0</td>
</tr>
<tr>
<td>2+1</td>
<td>32493.3</td>
</tr>
<tr>
<td>3+1</td>
<td>19592.0</td>
</tr>
<tr>
<td>4+1</td>
<td>15360 (2444.0)</td>
</tr>
<tr>
<td>5+1</td>
<td>15360 (2588.0)</td>
</tr>
<tr>
<td>6+1</td>
<td>15360 (10268.0)</td>
</tr>
</tbody>
</table>
It is mathematically possible to balance a game, but only pertaining to a certain play strategy. The strategy that I will pursue is where each player attempts to build defense units as fast as possible to protect him/herself against attack. The four buildings that grant defense units for each of the four DIVISIONS are the Guild (gu), the Institute of Magic (im), Barracks (ba) and the Workshop (wk). Running the simulation for permutations of all buildings required to obtain the first defensive unit, listing 7 shows the initial build times with one Quarry in play. Note the Quarry level is zero before running the simulation, forcing the first built to be a Quarry.

Listing 7: 1 total Quarry in play

1 Rgm [q, c, y, f, gu]= 10480.0 (0.0)
2 Mag [q, c, y, f, th, at, im]= 59480.0 (0.0)
3 Mil [q, c, y, f, fo, ba]= 13480.0 (0.0)
4 Eng [q, c, y, f, fo, wk]= 23480.0 (0.0)

The initial build times are very dispersed. I did chose the resource and build times for Magic to be high so that the Magic units could be made powerful and hard to obtain initially in the game, but having such a long build time would be unfair. There is no way to compensate through other game aspects the imbalance between Magic’s total build time and that of the Military, for example, with these initial values. The build times above, however, are not entirely representative of the build process. The number of Quarries in play early in the game strongly influences the build process and the system seems to be resource saturated with six Quarries in play, so I will observe the model progression as the number of Quarries in play increases to six total. Listing 8 shows two Quarries in play and a dramatic decrease in build times.

Listing 8: 2 Quarries: 1 given, 1 optional

1 Rgm [q, c, y, f, gu]= 5240.0 (74.0)
2 Mag [q, c, y, f, th, at, im]= 29740.0 (0.0)
3 Mil [q, c, y, f, fo, ba]= 6740.0 (0.0)
4 Eng [q, c, y, f, fo, wk]= 11740.0 (0.0)

With a total of three Quarries in play, listing 9 shows the Regime and Military DIVISIONS already having reached resource saturation by attaining the minimum build time. The optional Quarry is still highly needed for the Regime, but the Military postpones its build until the third slot. This indicates the Military’s optional Quarry is only partially needed to obtain the minimum build time.

Listing 9: 3 Quarries: 2 given, 1 optional

1 Rgm [q, c, y, f, gu]= 3740 (74.0)
2 Mag [q, c, y, f, th, at, im]= 19826.7 (0.0)
3 Mag [q, c, y, th, f, at, im]= 19826.7 (0.0)
4 Mil [fo, ba, q, c, y, f]= 5240 (2.0)
5 Eng [q, c, y, f, fo, wk]= 7826.67 (0.0)
Four Quarries in play (see listing 10) shows Magic as the only DIVISION still entirely dependent on having the optional Quarry built first. Engineering only partially benefits and the other two DIVISIONS are saturated with resources.

<table>
<thead>
<tr>
<th>DIVISION</th>
<th>Build Time</th>
<th>Time Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rgm</td>
<td>3740</td>
<td>122.0</td>
</tr>
<tr>
<td>Mag</td>
<td>11992.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Mil</td>
<td>5240</td>
<td>272.0</td>
</tr>
<tr>
<td>Eng</td>
<td>6720</td>
<td>276.0</td>
</tr>
</tbody>
</table>

Listing 10: 4 Quarries: 3 given, 1 optional

Listing 11 confirms the point of resource saturation to be five Quarries. All DIVISIONS have enough resources to obtain the minimum build times and only Magic is still partially dependent on the optional Quarry. One additional Quarry will saturate the entire system with resources. This is shown in the next listing.

<table>
<thead>
<tr>
<th>DIVISION</th>
<th>Build Time</th>
<th>Time Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rgm</td>
<td>3740</td>
<td>870.0</td>
</tr>
<tr>
<td>Mag</td>
<td>8900</td>
<td>314.0</td>
</tr>
<tr>
<td>Mil</td>
<td>5240</td>
<td>1320.0</td>
</tr>
<tr>
<td>Eng</td>
<td>6720</td>
<td>1060.0</td>
</tr>
</tbody>
</table>

Listing 11: 5 Quarries: 4 given, 1 optional

At this point every DIVISION waits to build the optional Quarry until the very end of the build sequence. The optional Quarry is more of a burden than a benefit. Listing 12 truly confirms the minimum times. And, these build times are those which are important to balance, since each player can technically obtain them. Unfortunately, the result also indicates the presence of a dominant strategy, but that is another issue.

<table>
<thead>
<tr>
<th>DIVISION</th>
<th>Build Time</th>
<th>Time Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rgm</td>
<td>3740</td>
<td>1992.0</td>
</tr>
<tr>
<td>Mag</td>
<td>8900</td>
<td>1220.0</td>
</tr>
<tr>
<td>Mil</td>
<td>5240</td>
<td>2892.0</td>
</tr>
<tr>
<td>Eng</td>
<td>6720</td>
<td>3076.0</td>
</tr>
</tbody>
</table>

Listing 12: 6 Quarries: 5 given, 1 optional
To sum up the data obtained in all six runs, all values are plotted on one graph in figure 3 so that the growth curve for each DIVISION relative to the others can be seen. The growth rate of Magic proves inordinate.

![Figure 3: build time progression](image)

Before I can conclude this section on game balance. Two more tests are relative to my analysis. In order to double check that the Magic DIVISION can indeed compete with the others, it is important to be sure that Magic can create its buildings and the extra needed Quarries in a time comparable to the other DIVISIONS. Listing 13, shows that Magic can create four extra Quarries in about the same time that Military can create its buildings.

Listing 13: uneven build

<table>
<thead>
<tr>
<th>Division</th>
<th>Build Time</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>10480.0 (0.0)</td>
<td>q, c, y, f, gu</td>
</tr>
<tr>
<td>Magic</td>
<td>13064.0 (0.0)</td>
<td>q, q, q, q, c, y, f, th, at, im</td>
</tr>
<tr>
<td>Military</td>
<td>13480.0 (0.0)</td>
<td>q, c, y, f, fo, ba</td>
</tr>
<tr>
<td>Engineering</td>
<td>23480.0 (0.0)</td>
<td>q, c, y, f, fo, wk</td>
</tr>
</tbody>
</table>

The last remaining test pertains to boosting. If a player is presented with an over abundant amount of resources from an long established player, then the model must be controlled to see if it still holds under these circumstances. It shouldn’t come as a surprise that the player is still subjected to the minimum build times similar to a resource saturated system. To confirm this, I have run the simulation, but have set START GOLD to 300,000, ensuring that all buildings can be bought from the reserve. Listing 14 again shows the minimum build times, but this time for a boosted player.

Listing 14: the effects of boosting

<table>
<thead>
<tr>
<th>Division</th>
<th>Build Time</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>3740 (298700)</td>
<td>c, y, f, gu</td>
</tr>
<tr>
<td>Magic</td>
<td>8900 (293800)</td>
<td>c, y, f, th, at, im, q</td>
</tr>
<tr>
<td>Military</td>
<td>5240 (298400)</td>
<td>c, y, f, fo, ba, q</td>
</tr>
<tr>
<td>Engineering</td>
<td>6720 (297400)</td>
<td>c, y, f, fo, wk, q</td>
</tr>
</tbody>
</table>

24
To conclude, it is apparent that the minimum build times are obtainable for each DIVISION. How should these numbers be taken into account? The calculated times are for each DIVISION to obtain defensive units, allowing each player an equal chance to defend themselves against attack. If I normalize the build times to the Military, the relative defense capabilities between the different DIVISIONS is represented by a single factor.

\[
\begin{align*}
\text{Regime:} & \quad 3740 \times m = 5240, \quad m = 1.4 \\
\text{Magic:} & \quad 8900 \times m = 5240, \quad m = 0.588 \\
\text{Military:} & \quad 5240 \times m = 5240, \quad m = 1 \\
\text{Engineering:} & \quad 6720 \times m = 5240, \quad m = 0.779
\end{align*}
\]

In my opinion, I chose the build times and resource requirements rather well, because they reflect what I wanted to achieve; Military is the fundamental combat type, Regime has a weak fighting unit that should be cheap to produce, Magic has a powerful hard to obtain unit and Engineering a comparable level to that of the Military. I could change the build times so that all build times are equal, but that would cause my game to lose a lot of character. Instead, I would choose the unit build times or combat defense ratings so that they compensate the factors above. This gives yet even more character to the game through diversity.

I have only analyzed one aspect of the build process for balance. Other aspects such as the balance of a player’s obtainable attack strength compared to defense strength will also influence unit build times and combat ratings with possibly conflicting situations. Other variables or restrictions might have to be introduced in order to obtain a true balance. Therein lies the delicate and tedious process of game balancing.
7 Conclusion

The main aim of this text has been to better the reader’s understanding of game design when employing a Game Design Document as development methodology.

To summarize, the GDD serves as a communication medium and a contract of sorts between different parties in the development process. Though the content of the GDD might vary heavily for each game, structure is needed in order to ensure that all game aspects are covered properly in the GDD. This structure can be obtained through different perspectives such as a feature-oriented, storyboard, or game token perspective. Versions or tiers should be specified so that each iteration of the development process adds another layer of features to the product, with the first version being a bare minimum to promote prototyping and immediate feedback. To promote versatility, the GDD must prove to be a living document. A medium such as a blog, forum or wiki might prove useful to hold the GDD, with an obvious preference towards a wiki, for its native ability of version tracking.

An extensive comparison with software engineering has been discussed in an attempt to identify a possible evolution towards software engineering. Software engineering has many development paradigms already and although game development is a specialization, commonalities are found in the requirement specification and development phases between the GDD and long standing methodologies such as RUP and the Spiral Model.

The practical aspect and most time consuming task has been creating a GDD first hand. The design of a fictitious game gave needed “hands-on” experience and underpinned literary findings. The importance of prototyping to obtain needed feedback quickly has been stressed. The build queue of the design was mathematically expressed and then implemented in Prolog. Through Prolog’s backtracking capabilities a mathematical analysis was done on the implemented build queue to ascertain if the chosen model was indeed balanced. That game balancing is one of the hardest tasks has been exemplified by changes cascading through the implemented model and the discovery of a dominant strategy.

As the game industry evolves further, other strategies may be adopted as development trends fluctuate. That games development is based in entertainment and therefore coupled to the psychological model of the player will undoubtedly keep game design a specialized segment of development.
# A Source Code

## A.1 A Prototype in Prolog

```prolog
%------------------------------------------------------------------------------
gain(0, 0).
gain(1, 0.1).
gain(2, 0.2).
gain(3, 0.3).
gain(4, 0.5).
gain(5, 0.8).
gain(6, 1.3).
gain(7, 2.1).
gain(8, 3.4).
gain(9, 5.5).
gain(10, 8.9).
gain(11, 14.4).
gain(12, 23.3).
gain(13, 37.7).
gain(14, 61.0).
gain(15, 98.7).
%------------------------------------------------------------------------------

rq(c, 1, 400).
rq(q, _, 100).
rq(y, 1, 300).
rq(f, 1, 100).
rq(s, 1, 210).
rq(p, 1, 1200).
% Regime

rq(gu, 1, 400).
rq(th, 1, 1600).
rq(dc, 1, 600).
rq(at, 1, 1800).
rq(ke, 1, 2300).
% Magic
%rq(th, 1, ...) *see Regime*
%rq(at, 1, ...) *see Regime*
rq(im, 1, 1900).
% Military

rq(fo, 1, 300).
rq(bq, 1, 1500).
rq(ba, 1, 400).
rq(mn, 1, 1200).
rq(ar, 1, 800).
% Engineering
%rq(fo, 1, ...) *see Military*
rq(tr, 1, 60).
rq(vl, 1, 80).
rq(wk, 1, 1400).
rq(gh, 1, 600).
%------------------------------------------------------------------------------

tq(c, 1, 1380).
tq(q, _, 480).
tq(y, 1, 840).
tq(f, 1, 320).
tq(s, 1, 180).
tq(p, 1, 920).
% Regime
tq(gu, 1, 720).
```

27
% Magic
% preq (th,1) * see Regime*
% preq (at,1) * see Regime*

% Military
% preq (fo,1) * see Military*

% Engineering
% preq (fo,1) * see Military*

% Regime

% Magic
% preq (th,1) * see Regime*
% preq (at,1) * see Regime*

% Military
% preq (fo,1) * see Military*

% Engineering
% preq (fo,1) * see Military*

preq(c,1).
preq(q,_).
preq(y,1) :- lvl(c,LVL), LVL >= 1.
preq(f,1) :- lvl(y,LVL), LVL >= 1.
preq(s,1) :- lvl(y,LVL), LVL >= 1.
preq(p,1) :- lvl(f,LVLf),
           lvl(p,LVLp),
           LVLf >= 1,
           LVLp < LVLf.

preq(gu,1) :- lvl(y,LVL), LVL >= 1.
preq(th,1) :- lvl(c,LVL), LVL >= 1.
preq(dc,1) :-
           lvl(th,LVLth), LVLth >= 1,
           lvl(s,LVLs), LVLs >= 1.
preq(at,1) :- lvl(th,LVL), LVL >= 1.
preq(ke,1) :- lvl(th,LVL), LVL >= 1.

% Magic
% preq (th,1) * see Regime*
% preq (at,1) * see Regime*

preq(ca,1) :-
           lvl(at,LVLat), LVLat >= 1,
           lvl(ke,LVLke), LVLke >= 1.

% Military

preq(fo,1).
preq(hq,1) :- lvl(c,LVL), LVL >= 1.
preq(ba,1) :- lvl(fo,LVL), LVL >= 1.
preq(mn,1) :-
           lvl(p,LVLp),
           lvl(mn,LVLmn),
           LVLp >= 1,
           LVLmn < LVLp.
preq(ar,1) :-
           lvl(hq,LVLhq), LVLhq >= 1,
           lvl(fo,LVLfo), LVLfo >= 1.

% Engineering
% preq (fo,1) * see Military*

preq(wk,1) :-

lvl(fo,LVLfo), LVLfo >= 1, lvl(y,LVLy), LVLy >= 1.

preq(gh,1) :- lvl(wk,LVL), LVL >= 1.

preq(tw,1) :- lvl(wk,LVL), LVL >= 1.

%------------------------------------------------------------------------------

lvlup(R,LVL) :-
    LVL0 is LVL - 1,
    retract(lvl(R,LVL0)), assert(lvl(R,LVL)).

set_basic_lvl(0) :-
    assert(lvl(c,0)), assert(lvl(q,0)),
    assert(lvl(y,0)), assert(lvl(f,0)),
    assert(lvl(s,0)),
    assert(lvl(p,0)).

set_basic_lvl(1) :-
    assert(lvl(c,1)), assert(lvl(q,1)),
    assert(lvl(y,1)), assert(lvl(f,1)),
    assert(lvl(s,1)),
    assert(lvl(p,1)).

reset(B) :-
    nb_setval(rsv,300),
    assert(lvl(xx,0)), % dummy so forall always succeeds
    forall(lvl(R,L), retract(lvl(R,L))),
    set_basic_lvl(B),

% Regime
    assert(lvl(gu,0)),
    assert(lvl(th,0)),
    assert(lvl(dc,0)),
    assert(lvl(at,0)),
    assert(lvl(ke,0)),

% Magic
    assert(lvl(im,0)),
    assert(lvl(ca,0)),

% Military
    assert(lvl(fo,0)),
    assert(lvl(hq,0)),
    assert(lvl(ba,0)),
    assert(lvl(mn,0)),
    assert(lvl(ar,0)),

% Engineering
    assert(lvl(tr,0)),
    assert(lvl(wl,0)),
    assert(lvl(gw,0)),
    assert(lvl(hq,0)).

%------------------------------------------------------------------------------

build_common :-
    reset(0),
    build(q,_),
    build(c,_),
    build(y,_),
    build(f,_),
    build(p,_),
    build(s,_).

build_regime :-
    build_common,
    build(gu,_),
    build(th,_),
    build(dc,_),
    build(at,_),
    build(ke,_).
build_magic :-
  build_common,
  build(th, _),
  build(at, _),
  build(im, _),
  build(ke, _),
  build(ca, _).

build_military :-
  build_common,
  build(fo, _),
  build(hq, _),
  build(ba, _),
  build(ar, _),
  build(mn, _).

build_engineering :-
  build_common,
  build(fo, _),
  build(tr, _),
  build(wl, _),
  build(wk, _),
  build(gh, _),
  build(tw, _).

build_all :-
  build_regime,
  build_magic,
  build_military,
  build_engineering.

% bt(+R,+LVL,-BT).
bt(R, LVL, BT) :- tq(R, LVL, BT).

% rt(+RQ,+RSV,-RT).
rt(RQ, RSV, RT) :-
  lvl(q, Q),
  gain(Q, GAIN),
  ( GAIN =:= 0 ->
    if (RTO > 0 -> fail ; RT is RTO)
  ; RT is (RQ - RSV)/GAIN).

t(+BT,+RT,-T).
t(BT, RT, T) :-
  max(BT, RT, T),
  BT >= RT, !, T is BT ;
  T is RT.

% rsv(+T,+RT,+RQ,+RSSVO,-RSV).
rsv(T, RT, RQ, RSSVO, RSV) :-
  lvl(q, Q),
  gain(Q, GAIN),
  ( GAIN =:= 0 ->
    RSV is (RSSVO - RQ)
    ; RSV is (T-RT)*GAIN).

% build(+R,-T).
bUILD(R, T) :-
  nb_getval(rsv, RSSVO),
  lvl(R, LVL),
  LVLUP is LVL+1,
  preq(R, LVLUP), !, bt(R, LVLUP, BT), !, rQ(R, LVLUP, RQ), !, rt(R, RSSVO, RT), !,
t(BT,RT,T),!
rsv(T,RT,RQ,RSV0,RSV),!
nb_setval(rsrv,RSV),
% writef(‘%w(%w)= %w (%w)
’,[R,LVLUP,T,RSV]),
lvlup(R,LVLUP).

% exec_build_seq(+0,+(List),-T).
exec_build_seq(Acc,[],Acc).
exec_build_seq(Acc,[Head|Tail],T):-
build(Head,TH),
TAcc is TH + Acc,
exec_build_seq(TAcc,Tail,T),!.

% minimum_build(+Lbl, +Array, +R).
minimum_build(Lbl,Array,R):-
assert(min(9999999)),
forall(T1:min(T)),retract(T1),!
permutation(Array,Perm),
reset(R),
% forall([lvl(L),retract(lvl(L)))],% add these 2 lines to have the
%assert(lvl(q,6)),
exec_build_seq(0,Perm,T),
tmin(T0),
T < T0,
retract(Tmin),
assert(Tmin),
nb_getval(rsv,RSV),
writef(‘%w%w=%w (%w)
’,[Lbl,Perm,T,RSV]),
fail.

min_build_all:-
Common = [q,c,y,f,p,s], minimum_build(‘Common’,Common,0); Magnitude = [th,at,ke,ba,ar,em], minimum_build(‘Mag’,Magnitude,1); Mil = [fo,hq,ba,ar,em], minimum_build(‘Mil’,Mil,1); Eng = [fo,tr,wk,gh,te], minimum_build(‘Eng’,Eng,1); true.

min_unit_time:-
Rgm = [c,y,f,gu,q], minimum_build(‘Rgm’,Rgm,0); Magnitude = [c,y,f,th,at,ke,ba,ar], minimum_build(‘Mag’,Magnitude,0); Mil = [c,y,f,fo,bq,em], minimum_build(‘Mil’,Mil,0); Eng = [c,y,f,fo,wk,te], minimum_build(‘Eng’,Eng,0); true.
References


Analysis: Game Design using the GDD

(part 2 of 2)

June 4, 2008

Creation: 2008-01-07, Revision: 217
1 High Concept Document

1.1 Game Concept Description

Strategic warfare from an isomorphic perspective has almost been perfected by Blizzard, however, competitive games are limited in the number of players that can participate in a war in one single game. This game is an attempt to go beyond that limitation by offering players an infrastructure and a set of rules that guide warfare in today’s MMO craze.

In order to break the limit in the number of players, we expand on a concept that has already been used in small browser games. These browser games target an audience of players that log on periodically to check their account. Their player account remains active in the game even when the player is not logged on. To compensate for the fact that players are in different time zones, we must slow game time down to a point that players can log in on an average of one or two times a day in the least to manage their account.

The ultimate goal of this design is to create a hybrid game which tries to fill the niche between real time strategic warfare and low quality browser games. Also, a major issue is to reward the player properly. Browser games reward online activity and/or mundane bookkeeping tasks, when reward should be given to those players which are the best strategists. Versatility is also something that will be kept in my mind. Browser games often contain a dominant strategy which once learned reduces the game to mere labor.

The audience for the game are those that have a limited amount of time to log on, but feel that they have the smarts needed to eliminate opponents. The interface and infrastructure of the game should automate mundane maintenance to zero. It is possible that the game will fit an older age group of individuals. This has the advantage that they have the means to purchase the game, but also the disadvantage that there are as of now, less of these players in the gaming community. The target audience are those individuals who have diplomatic skills, who are war strategists or virtual city builders. Players will have to have social skills in order to work together towards a common goal.

The game should be symmetrical on the level of groups, implying a level of fairness by giving each group equal opportunity through having access to the same game elements. Players that are not in a group will be easily defeated because they will not have someone to defend for them when they are not online and because of the intrinsic one-sidedness a single division characterizes.

1.2 Challenges

A player wanting to excel in this game must overcome numerous challenges as the scope of the game broadens. At first the player is presented with the
simple task of starting a city. He/she must continue building the city while guarding against being conquered by enemies. The scope the game is rather limited at this point since it will be centered around one village and a very small portion of the map. At this point the player will start to interact with other players through friendly communication and conflict. The scope of the game shall broaden as the player has constructed more and more of his/her city and attempted to conquer and destroy neighboring enemies.

As the game progresses, the player will eventually join a group of players striving towards a common goal. Leaders of such groups will be required to have diplomatic skills in order to balance relations with other groups of players. And, players will have to learn how to cooperate in order to reach a common goal.

To sum up the challenges . . .

- **Interface:** The player must first learn the basics of the interface in order to manipulate game elements. A tutorial could help the player, but the interface must be largely intuitive or similar to previously “standardized” game interfaces.

- **City Building:** The basic game elements are buildings which must be built in the player’s city. He/she must learn the purpose of each of the buildings, the new interfaces they present and how to use them correctly.

- **Basic Warfare:** In the beginning of the game a player will have access to a limited amount of combat elements. These must be put to good use to eliminate possible threats in the immediate environment.

- **Social Skills:** Basic communication skills are needed in order to determine if other players in the immediate area are friend or foe. A player must also start negotiating whether or not to join a group of players.

- **Diplomatic Skills:** A player who is not the leader of a group will only have to balance personal relationships. Those players who are elected to lead a group of players will be required to handle relations with other groups of players. This can be a very delicate and/or tedious task.

- **Cooperative Building:** Certain buildings in the game are only available when a group of players have successfully filled cooperative requirements. Also, certain buildings of a division in the game must be built to the advantage of group member. This is an altruistic aspect. Players will have to learn how to build efficiently and cooperatively.

- **Cooperative Warfare:** In order to rise above as a group of individuals working together, players will have to learn how to work together as a fighting unit, defending and attacking in unison.

- **Cooperative Diplomacy:** As the game evolves, certain groups of players will dominate the game. If these dominant groups are left unchecked, they will be unstoppable. Smaller groups of players can work together to defeat a much larger group of players if an accord can be reached.
1.3 Gameplay Modes

Welcome Area: When the player enters the game, this is the mode which greets the player. As is customary with most games, this mode will have options to move to other game modes such as the Options Screen.

Options Screen: This mode allows the player to control aspects of the game according to personal preference, such as interface settings and feedback behavior. Graphical resolution and sound volume are good examples of settings belonging in this game mode.

Account Sitting: If a player is absent for an extended period of time another player should be enabled to account sit for the absentee. ¹

Warfare Mode: This is the primary game mode. Here the player will manipulate all game elements in order to conquer the world. This game mode is extremely large and in order to present the player with only a reasonable amount of feedback, to go against input overload, it might be beneficial to have some levels of indirection in the interface. This indirection will give rise to Detailed Control modes with a limited scope of information.

Detailed Control: There can be many of these modes. Each shall contain the scope of one game aspect which is a subset of of the primary game mode.

¹Actually, it is questionable if this mode will be needed at all if others can defend for the absent player. Not logging on should only be a minimal disadvantage, but rather no gain
2 Game Script Document

2.1 The Rules

2.1.1 Semiotics

AVATAR depending on the DIVISION represents the player in the game: Delegate, Militant, Medium or Architect.

SEGMENT Mind or Body.

DIVISION Mind:Regime, Mind:Magic, Body:Military, Body:Engineering

DOMINION any sized domain controlled by one or more players.

CITY village/town/city/metropolis...

INHABITANTS people in CITIES that do not have a specific profession.

COMMISSIONED those in the CITY with a specific profession.

POPULATION the total number of people in a CITY, INHABITANTS and COMMISSIONED.

CITY FOOD the maximum number of sustainable people in a single CITY relative to the amount of FOOD produced by Farms.

TECH a technology upgrade to better a game aspect that is already possessed by the player.

COMMAND AVATAR experience points measured in INFLUENCE, MANA, AUTHORITY, BUILD POINTS

FOOD commodity that determines how much POPULATION can be supported

TIME COST cost in time before a game element is available.

COST cost in resources in order to be able to purchase the game element.

BUILD CREW the number of workers available to do labor.

TECH LEVEL regulates how high a player’s technology rating may be.

CELL one hex on the world map

2.1.2 The Sequence of Play

The player must choose which a DIVISION he/she wishes to play. It is best to delay this choice as long as possible until the player has a slight understanding of what the differences are. This gives better immersion. When the player enters the main game mode, he/she is represented by an AVATAR that corresponds to the DIVISION chosen.
When the player first enters the main game mode, he/she is presented with the
world map (hidden by the Fog of War). He/she can move around to find a
nice location for their first CITY. The player is free to establish a CITY at any
available and legal location. The player may, however, decide to join an existing
DOMINION before creating their first village. (see 2.2.11)

Q ⇒ How, exactly, is available and legal defined above?

Q ⇒ If the player must actually move his AVATAR to explore and find a nice
location for his/her first CITY on the “real time” time scale of the game, then
the player may actually have to wait quite a long time before actually being able
to start the CITY, i.e. it will take forever to scout out a nice location. Is there
an alternative to speed the process? Should we limit the distance a player can
move based on FOOD supply? Or, shall we just make the process instantaneous,
since there is no disadvantage to other players at this point?^2

The starting condition (see 2.1.2) will allow for the player to at least start
his/her first village with ease. Once the player has selected a nice location on
the map for their CITY, the space is updated to display it as occupied by the
CITY.

The BUILD_TREE pertaining to the player’s chosen DIVISION will determine
which buildings will be immediately available to him/her. It is assumed under
normal game player circumstances that they player will build each building on
the BUILD_TREE until the player controls at least one of each. The player may
choose to build more than one of each type to reap advantages.

Q ⇒ Build sequence must be determined by prototyping. Also, the fastest
build sequences must be determined and game balanced. The game start is
particularly important to get balanced. In South Korea a game of Starcraft
lasts an average of 15 minutes. –Movie: Gamer Revolution

TODO: ...

• And the story goes like this ...

  Every player is represented in the game as an AVATAR. He/she is born
  somewhere random. In the beginning he/she receives enough resources
  (followers instead of resources?) to establish a CITY. When entering the
game he/she can walk anywhere and start a CITY. He/she can also choose
not to start a CITY, but join a DOMINION and help run it. The player must
walk(initially, really walk?) his diplomat to that DOMINION and place him
in some CITY.

  Once in a DOMINION a player can move his/her AVATAR to any CITY in
  the DOMINION. This means that when a AVATAR joins a DOMINION, his/her
  CITIES are given up to the DOMINION. If an AVATAR has enough political
  sway (see below) in a CITY, then he/she can split off the CITY from the
  DOMINION if the relationship goes bad. Really, you can’t move the city?

^2Instantaneous is not an option if the player’s AVATAR was killed and re-spawned
Otherwise, just maybe take followers along with him. Each AVATAR has a percentage of political sway. If an AVATAR is exiled, they can possibly take a mass of people/followers with them (part of the army perhaps too, depending on the type of the DIVISION.

If a CITY is taken with AVATARS in it. Every player who’s AVATAR was captured is frozen from making any communal actions. The attacker can choose to kill or keep the AVATAR captured? Captives can be bargained for? Players can AVATAR their diplomat and restart. Can he/she restart? Or is the player just removed from the game completely (reduces the number of players over time)?

A AVATAR/player can be banned from a commune. He/she can then join another or start a new CITY/DOMINION. If the AVATAR does not have enough resources/followers, he/she must wait until his influence(COMMAND) gradually grows until he is able to make one. Can he/she be picked off, while standing in the middle of a field? Can he/she be harbored by DOMINION for a while?

The Starting Condition …

The starting condition for each player will be a certain amount of resources and COMMAND. Resources allow him/her to build the basic buildings needed to start a CITY. The starting COMMAND allows for him/her to have enough reign over people in order to have a minimal BUILD_CREW and starting POPULATION.

Q ⇒ Make sure there is not a condition where the player is stuck without the means to at least start a new CITY, e.g. he/she has lost all of their CITIES and does not have the means to establish a new CITY and gain resources. A possible solution could be having players receive an absolute minimum income periodically.

Q ⇒ What is the actual starting condition? Should we give the players a possible choice between lots of resources, lots of offense or ample defense to guard the new found CITY?
2.1.3 Goal of the Game

The game in its entirety should lend itself to conducting warfare; whether it be the race to arms, the building of massive defenses to guarantee existence, the sabotage of other players or the trading of goods to purchase safety and protection.

Termination Condition...

The termination condition in which a single player or even multiple players together in team win the game is a situation that shall seldom occur. The purpose of the game is to reign over the entire world. But, this is by far an absolute condition which is difficult to achieve in a MMO setting. It can be safely assumed that the game world will always contain small players and DOMINIONS. A termination condition for winning the game could be placed at the moment when a single DOMINION controls more than a certain percentage of the world. In contrast to having this be set in stone, since the game spans an extended number of months, the world can simply be monitored by staff to see when a world needs to be reset.

Q ⇒ At what point is the player forced to leave the game? If the player loses all his/her CITIES? If the player loses his/her AVATAR? (see 2.2.1)
2.2 Features

TODO: ...

- what makes Eve Online less appealing than TribalWars? Is it the fact that it is a full screen client making it impossible to do something else?

- what percentage of the population can be drafted for a profession? Surely, not everyone.

- On a broad spectrum, each village becomes a specialized building block for your realm, similarly like a barracks is a specialized building block for your village (a realm also, but lower tier). This broadening of scope could be paired with the gradually broadening scope of the game in general.

- Choice: upgrade a low level resource +1 at low cost or a high level resource +1 at high cost. That means that the resources produced must be proportional to the amount spent. Otherwise it might be more beneficial to invest in lots of small resources or few high level ones.

- slave labor -vs- employed laborers? If someone develops the slave labor skill, then they can employ captured villagers until their morale swings far enough so that they can be assimilated in with the new villagers.

- DOMINION tech tree instead of a player tech tree. Each player must contribute resources to have it evolve. Do big players have a greater influence then? Since, they have more resources to their disposal?

- 30 second variable reward (0-60 second interval) or 30 fixed rate reward. (On an elongated time scale such as we are using, research needs to be done to see if reward rates are slowed as well. WoW is on real time scale. What is the rate of the reward there?) Drop in rewards = punishment, can lead to quitting. New and exploration is a drop in reward. It is the average rate of reward that matters, not the interval or size of reward.

http://www.gamasutra.com/features/20020204/hopson_01.htm

2.2.1 The Player’s AVATAR

The AVATAR is a representation of the player in the game. It is one of the objectives of the game for each player to protect their AVATAR from harm. The player’s AVATAR must always be placed in a specific CITY. This can be either the very first CITY a player creates, a city in the DOMINION he/she is a part of or another renegade CITY.

Q ⇒ This aspect of protecting the AVATAR could be delayed for later versions.

An initial idea was to allow for the AVATAR to advance in level. We could even require the player to pay for advancement to the next level. However, player COMMAND has been introduced instead (see 2.2.3).
TODO: ...

- AVATARS in renegade CITIES are considered in hiding or exiled with a high likelihood of being detected and/or captured in the process.

- AVATARS can be required to perform special tasks. Military’s AVATARS can be heroes with super stats. Magic, specialized magic-users? And, indeed allow for AVATAR advancement. (see above)

- Quote: "A noble’s ransom could be very high, and indeed some made a living by capturing and ransoming nobles in battle. Even peasants, who did not share the bonds of kinship and culture, would often avoid killing a nobleman, valuing the high ransom that a live capture could bring, as well as the valuable horse, armour and equipment that came with him.” [7]

- Allow for AVATAR escorts to guarantee their survival while traveling.

- A player must send his/her AVATAR to the opposing DOMINION for negotiations. The AVATAR can be captured or held hostage? If the diplomat must move to the other DOMINION to establish contact and the AVATAR movement speed is low, then it will take a long time (delay) in order to establish pacts, no? Possible, just for the initial transaction?

- Make it so that people can add another AVATAR to their account? To go against people making more than one account just to have another AVATAR. If you give the player all the options of having more than one account in one account, then people will not have the tendency to create another account, will they? But, if COMMAND is based on time in game, then two accounts means double the amount of time in game.

---

dependent on resource cost. This is not a good idea because a player can easily receive resources from others in the game, i.e. boosting. AVATAR advancement must be based on in game experience.
2.2.2 The DIVISIONS

Allow players to swap roles with another player? What if someone started the Magic DIVISION, but really *is* the leader of the DOMINION? A player can trade away all assets to another player and then change AVATAR type. His COMMAND could be converted. How in actuality this will occurs remains to be explained.

MIND: REGIME . . .

- In charge of diplomacy
- Set village build templates
- Can establish a DOMINION
- ONLY division type that can be elected LEADER
- Can change their alias so that they can actually play the role of the spy? TECH?

COMMAND Gains: pacts, alliances, recruits(one credit per player?), votes cast by players on vote created by the player,

LIMITS:
- Number of players possible in the DOMINION (drop smallest players from tribe if number of players exceeded) -OR- TOTAL population of DOMINION
- ? Number of total population under the player’s control that is not military/magic/etc...?

MIND: MAGIC . . .

- Magic units
- Heal units
- Citywide Spells: (magic-users must walk to the cities)
  - Good Healer Spells: Wall of Stone (your own or opponents =), can not attack or be attacked except by Monster from Monster summoning)
  - Evil Heal: Creeping Doom
  - Mage: Teleport (units), Monster Summoning, Rain of Fire
- Magic units are NOT under the control of the military

COMMAND Gains: per spell cast, magic attacks in battle, magic defends in battle, units trained,

LIMITS: Number of magic units under player’s control,
BODY: MILITARY . . .

- Only DIVISION type that can be elected GENERAL
- Controls military units, including structural offensive (NOT structural defense)

COMMAND Gains: attacks in battle, defenses in battle, units trained,
LIMITS:
  - Number of military under player’s control
  - GENERAL determines largest army to command

BODY: ENGINEERING . . .

- Can build all the buildings (loss of functionality if not enough build level)
- Defense structures
- Advanced build crew
- Build crew must walk to destinations to build certain structures

COMMAND Gains: kills by defense structures, number of buildings built,
LIMITS: Number of buildings under player’s control (including defensive structures),

2.2.3 COMMAND

The more COMMAND the player gets the more POPULATION he/she can control. A CITY gains in POPULATION, so in order for the player to control more CITIES, he/she must gain COMMAND. 4

Q ⇒ Does a general need to have so much AUTHORITY that he can control ALL the troops? If so, then he needs a way to get MASSIVE amount of AUTHORITY (in combo with MIND?). There needs to be another limiting factor to keep him from creating the massive army for himself only. Something like if you are commander then you build different buildings than those allowing for fast troop recruiting.

TODO: . . .

- Like ”Small Gods”. The more you have accomplished the more you can have under your control. The more battles won, attacks defended, buildings built, diplomacy done, etc... the more people you can have under your influence.
- A player can choose to create as many CITIES as he/she desires as long as the population remains under his/her COMMAND.

4It shouldn’t be too hard to maintain your COMMAND. It should be a fluidic kinda thing.
• The more **POPULATION** a player can control, the more civilians, workers and military he/she can control. Civilians can be trained into workers, scientists, military, etc...

• If the population gets to be beyond the players **COMMAND**, the populations of **CITIES** will start to revolt. A completely revolted city can become Barbarous.

### 2.2.4 The Build Process

A player can choose from any of the buildings in the build tree and assign it to the build queue. There is no limit to the number of buildings that can be placed in the queue, even if they player does not have the resources or prerequisites to have that building created.

\[ \text{Q} \Rightarrow \text{queue is not endless, but can contain enough for a build process that is 24/48 hours?} \]

The build process only starts when a building is next in queue, resources are available and prerequisites met.  

\[ ^5 \text{When the build process starts, resources slowly start to be charged from the player as the build process progresses. The build time of the building is set according to the number of BUILD_CREW available to work on the building during the entire build process.} \]

Because a player might not always want to build at full speed (to save resources perhaps), the interface should have a gauge, which shows the percent of the build capacity should be used. 100% means the build capacity is set to the maximum the BUILD_CREW allows. 0%, the lowest setting means full stop.  

\[ ^6 \text{Buildings in the build queue can be moved up or down, or simply removed completely. An insert function might be required, but if a player can add a building to the queue and move it up or down the desired place, an insert might not be needed. This of course depends on how long the build queue typically becomes and how easily (task and time consuming) it is to move a building far in a long queue. One more option might be needed if the build queue becomes particularly long. A player might want to remove all items from the build queue from a certain point all the way until the end. When a player gives the command to remove a building from the queue, the player could be presented with a question if he/she wishes to remove all items after the item being removed as well (in other words, invalidate the rest of the queue).} \]

Actions taken on the building currently being built (the actual build process) must also be discussed. Once a building has been started, resources are considered to spent on the creation of it until its completion or cancelation. The

\[ ^5 \text{For visualization purposes certain buildings can be withheld from the player as options until at least the prerequisite buildings have been schedule or built. This simplifies the game and hopefully makes the player more eager to see the advanced buildings.} \]

\[ ^6 \text{It will be up to the interface to equate 0% to a pause of the build queue discussed elsewhere in this section} \]
player may pause and resume the build process without effecting production at all. Resources simply stop being spent on the building and production is halted. If the player would like to cancel the build process, he/she has the two options that are identical to the destruction of a building.

The destruction of a building can also be scheduled by the player. When destructing a building, the player has two choices.

- The building can be deconstructed or dismantled using the **BUILD_CREW**, at the same or nearly the same **TIME_COST** of the building, allowing the player to recuperate a certain percentage of the spent resources.

- The building can be scheduled for demolition also using the **BUILD_CREW**, at a small fraction of the **TIME_COST** of the building. No spent resources are recovered and possibly a small set fee can be charged.

**BUILD_CREW ...**

The number of workers in the **BUILD_CREW** is a multiplication factor against the building’s build time to give an actual build time.

**TODO:** ...

- If the **BUILD_CREW** factor is not linear such as life (adding more people to a project does not linearly increase the project speed [4]), then it shouldn’t be a disadvantage to just serialize the builds instead of having more than one build queue with multiple build crews.

- Can the build crew be split into two teams to work on two or more buildings at once? Means builds at a later stage of the game progress much faster, but the player must be even more skillful and aware to when the build queue will be empty again. = NO! Should be equivalent to just serializing them if the same build crew is used.

- Building small buildings at fast intervals == building large buildings at long intervals, no? So, in both cases growth is linear.
2.2.5 Resources

What resources are there and how does one acquire them? Is there GOLD in the game? Certain amount of resource cost to mint one gold coin?

Add COMPETING FOR RESOURCES! Very important to build game conflict.

Resource Networks

Each village can build a Resource Node. The Resource Node is a building which can be created by the player in each village. Depending on how advanced the node is (what level it is) determines how many resources it can hold and transport. If a node graph is used to represent the entire resource net with each node of the graph representing a village having built a Resource Node, then the edges of the graph represent a connection between each of the Resource Nodes. The graph can potentially be a complete connected graph, however, the resource net should function such that a minimally spanning tree should suffice. On figure 12, nodes v1 to v8 would represent the villages, while e1 to e8 would be the weighted resource capacities between villages.

Figure 1: An undirected graph with 8 vertices and 8 edges [11].

How to realize the above? a set number of merchants in each village? or a weight on each edge between villages?
The idea behind the resource network is that it should be similar to network traffic routing in TCP/IP. When a building is scheduled for creation in a village, if resources are not readily available, they will be slowly moved through the network from different sources so that the build process can begin/continue without interruption. Of course, no interruption is the optimum condition, however, if there is network congestion due to a poorly built network or the resources are simply not available, then the build process will start to be delayed more and more.

The user will need some kind of feedback to be able to see how the network is running, which routes are heavily travelled and where bottlenecks are. Heavily travelled routes or congested routes could be colored red.

For inter-DOMINION trade to be possible, a trade pact must be created between the two interested parties. (see 2.2.12) This is done by one party offering the pact and a node by which to export resources. The other party then must agree to the pact and designate a node of his/hers through which the resources will be received. More than one trade pact can be assigned so that the flow of resources are increased. The added traffic consumes transport means of course. The added route acts exactly as an edge in the connected graph described above.

Q ⇒ That’s all nice and neat above, but how do you actually GIVE 100k resources to another player then?

A CITY that is under attack or is occupied by enemy troops CAN NOT send or receive resources. (see 2.2.6)

2.2.6 Trade

A CITY that is under attack or is occupied by enemy troops CAN NOT conduct trade.

Q ⇒ exactly what trade goods require a trade pact?

Q ⇒ An interface can be implemented that gives the attacker the right to block or allow trade, depending if he/she has isolated the city by force.

TODO: ...

- Trade everything... villages, resources, money, cows, whatever...

- increase the importance of trade so that players are inclined to make trade pacts and force other players to undergo embargoes.
2.2.7 FOOD and POPULATION

The most basic building, the CITY Center, provides an elementary amount of FOOD in order for the player to form a BUILD_CREW. 

Once the player has formed a BUILD_CREW, Farms can be built. Each Farm produces a certain amount of FOOD which increases CITY Food in the CITY. Once the CITY Food is raised for that CITY, the actual POPULATION can begin to increase at a certain functional rate, perhaps greater than linear. 2.4.1 If at any point the POPULATION exceeds the CITY Food, the reverse effect shall take place, population decrease.

Q ⇒ Who will die/be removed first? peasants? army? random? We can possibly have the general population migrate to other CITIES where food is more plentiful, sparing the player from having to wait until the population increases in one area while it is dying in another. Troops should probably not move automatically; they are under orders to stay. The general population might migrate to another CITY, but the troops will die of starvation, so maybe, just random.

TODO: ...

• sustainable population 30-120 per square mile. (6d4x5)

• And the story goes list this ...

* Villages range from 20 to 1,000 people, with typical villages ranging from 50-300. Most kingdoms will have thousands of them. Villages are agrarian communities within the safe folds of civilization. They provide the basic source of food and land-stability in a feudal system. Usually, a village that supports orchards (instead of grainfields) is called a “hamlet.” Occasionally, game writers use the term to apply to a very small village, regardless of what food it produces.

* Towns range in population from 1,000-8,000 people, with typical values somewhere around 2,500. Culturally, these are the equivalent to the smaller American cities that line the interstates. Cities and towns tend to have walls only if they are frequently threatened.

* Cities tend to be from 8,000-12,000 people, with an average in the middle of that range. A typical large kingdom will have only a few cities in this population range. Centers of scholarly pursuits (the Universities) tend to be in cities of this size, with only the rare exception thriving in a Big City.

* Super Cities range from 12,000-100,000 people, with some exceptional cities exceeding this scale. Some historical examples include London (25,000-40,000), Paris (50,000-80,000), Genoa (75,000-100,000), and Venice (100,000+). Moscow in the 15th century had a population in excess of 200,000!

---

7This essentially means that the CITY Center can not have a BUILD_CREW as a prerequisite or a maximum of one, the player’s avatar himself. Which could account for its exceptionally long build time.
• As Napoleon famously said, an army marches on its stomach, a weakness that has applied to all military campaigns in history. Medieval armies were supplied much as earlier armies had been. With the advent of castle-building and the extended siege, supply problems had to be solved on a scale seldom seen before, as armies had to stay in one spot for months, or even years. ** Should be a tactic to surround a village for a long period of time without attacking it to suffocate it to death through lack of food supply from surround farms.

2.2.8 Troop Movement

Q ⇒ It is a little difficult to judge how troop movement should occur exactly to provide interesting game play without prototyping. Rules surrounding this aspect should provide a system that logical, e.g. two massive armies standing next to each other should be aware of each other’s presence and possibly automatically do battle. Another of such an aspect is that of flanking. The hexagon map 3 used to represent the world allows for entry into each CELL from six directions; three of these could be considered as open for flanking.

Q ⇒ Should be possible to order your troops to position so that they can cut off incoming reinforcements to a battle already in progress.

2.2.9 Troop Training

Q ⇒ Since the player must wait for the POPULATION to grow, then troop training shouldn’t take all that long, perhaps.

TODO: ... 

• Player can upgrade troops. With Swords and Armor. If a soldier has less armor, then his speed will be greater... player can create a "barbarian" this way... no armor ... large weapon. Maybe have all units created like this. You could have magic-fighters.

• Train reservists for when active duty die. Faster to replace lost troops instead of having to recruit new active duty.

• In order to detect incoming attacks a player must put scouts out. A band of scouts can be knocked out by other scouts. A band of scouts has a certain radius of detection.

• If a village is taken over and the troops are out supporting another village, should the troops just become lodged in the protected village? (If the food supply is ample enough?)

• Only inactive troops should be able to be dismissed. I assume they simply disappear, so you don’t want troops suddenly disappearing in the middle of a battle.
• There is something wrong psychologically when you have to build your troops manually and after firing them they all get trashed and you get to start over [1]. Even if you did get another village. Having another village isn’t necessarily enough reward after a while.

2.2.10 Spying and Scouting

Scouting and troop visibility plays a major role in the game. When troops are placed at a certain location, they can ascertain different kinds of information, depending on the troops type, e.g. Scouts will pick up much more location information and at a far greater distance than a regular Swordsman.

Different distances will also produce different levels of information. A Scout or simply the town’s perimeter might give the player warning of a mobilized army, but will not reveal perhaps the exact size, unit types it comprises of or even who the army belongs to. A Scout can then be sent at close range to gather more exact info about the troop movement.

In order for a player to conceal his assets, he/she may also use Scouts to guard against being scouted. These Scouts are sent along with armies or positioned in particular locations which give a certain level of coverage against being scouted. The player with the most Scouts present will have the better chance of success.

At a certain point the Regime DIVISION can obtain the ability to install Spies in a CITY. This will give the spying player certain global DOMINION information (estimates). (see 2.3.1)

TODO: ...
• Spy for intell on trade pacts (see 2.2.12)
• In order to do an arms race, one must be able to estimate the size of the opponents army.
• Spies should be able to steal technology

---

It should be noted that espionage should never reveal information that would promote players to use mediums outside the game, just to avoid information discovery, e.g. revealing forum information could push players to start an external forum.
2.2.11 The DOMINION

A DOMINION is a collection of players that have decided to play together in a team. A DOMINION may be allied with another DOMINION, so it is not necessarily that a DOMINION is opposed by the rest of the world.

Players shall find that playing alone as a single DIVISION is incomplete and weak. No DIVISION alone can obtain all available game aspects, they each lack certain aspects specialized by another DOMINION. In order for each DIVISION to compliment the other, they must form a DOMINION. By having at least one strong player in each DIVISION a DOMINION can form a complete whole and also open up the possibilities to obtain DOMINION specific buildings.

At any point in the game a player may choose to petition to join a DOMINION. In order to join a DOMINION, a player must receive an invitation from someone in the DOMINION. If the player receives and accepts the invitation, then he/she shall immediately be made a member of the DOMINION.

ADVANTAGES of DOMINION:

• Communication

• Others can defend for you

• Other DIVISIONS compliment weaknesses
  • Engineering supplies Farm space ???
  • Engineering supplies majority of capacity for others to build buildings
  • Military supplies DOMINION headquarters for total command (military ranks)
  • Regime supplies resource networks, trade
  • Magic supplies ... what ???

• DOMINION Structures and Units

TODO: ...

• Asking for an invite to a DOMINION is also putting out your AVATAR for the possibility of being killed. What then? No, imagine a large nation that wants to join a DOMINION. You wouldn’t send the leader to go knocking on the front door.

• When a player joins a DOMINION, he/she will immediately be part of the group, however, until the AVATAR is placed in a city that is part of the DOMINION, certain game aspect will not be available to the player.

• How to start a DOMINION?
• How to keep all players from merging immediately into one giant tribe?
  People want to retain their group, but still want the communication benefit. So, increase inter-DOMINION communication possibilities and people want to stay in their group?

• Limiting the DOMINION members to 100 players means that it is impossible to create a single DOMINION that contains all the players in the world...
  Having the limit means there is always conflict, no?

• What should the advantages/disadvantages of having ... BIG tribe/few players -OR- BIG tribe/lots of small players? Large tribes cost more to maintain? What is the cost? Resources? Money? What? What is the downside for small tribes?

Ranks ...

Each member of the DOMINION will enter the DOMINION with the lowest possible rank or no rank at all. It is up to the DOMINION aristocracy to assign different ranks to each player.

Q ⇒ Two different types of ranks exist, assigned ranks for the DOMINION and also ranks for the player in his/her DIVISION. I’m not even sure if the first is required. Someone for magic can not be made leader?

TODO: ...

• players get different military ranks. A player of lower rank cannot cancel an order given by a player of higher rank. This means it would be good for the general to delegate instead of directly placing orders, because otherwise other players will not be able to change the order if it is needed.

2.2.12 Inter-DOMINION

Diplomacy contributes a large part of the game. It is important that the infrastructure of the game supply each player with options to communicate and bargain with so that solid agreements can be made.

When a DOMINION has decided to create a declaration, pact or embargo with another DOMINION, the creators of the pact may choose to have the event published as a global event to be posted on the Event Record (see 2.5.2), so that it is publicly visible by all. This allows a DOMINION to openly declare war on another DOMINION, for example. Another option in the creation of the agreement can be the automatic publishing as a global event when the agreement is lifted. This allows players to force a publicly visible agreement to be made that can not be lifted without going unnoticed.
Q ⇒ Both military and regime needed for declaration? Must military follow regimes law?

TODO: ...
- allow players to do a coup d’état?
- Enable political warfare. Allow a player to stage an attack from another player resulting in a fake report. This can then be used to lie and cheat.

Declarations ...

In order to keep track of different military relations with other DOMINIONS, each DOMINION can form a declaration with another DOMINION.

The following pact types are available:
- Ally (indeterminate length, non-binding/binding)
- Ally (fixed time period, non-binding/binding)
- Neutral (NOT the same as a non-relationship status)
- Non-aggression pact (NAP) (indeterminate length)
- Non-aggression pact (NAP) (fixed time period) //TODO binding?
- Enemy

Each pact will have a written agreement with it, wherein leadership can specify verbal conditions that the pact is founded on.

TODO: ...
- "... one which would be undenounceable/un-denunciable for a term of twenty-five years." If undenounceable then make it binding in the game?
- Wording: "The TENSION between Germany and Poland has become intolerable..."
- "... the dispatch of such an eminent politician and statesman emphasized how serious were the intentions of the German Government.", "... in noteworthy contrast to England, ..., had sent only an official of second-class rank to Moscow." Make it so that one DOMINION can see the rank of the official visiting to make a pact? Of course at the risk of losing such a high ranking diplomat. Can the receiving party kill the diplomat? If so, what happens if they do kill the diplomat who is abroad? It should be customary to send a lower diplomat then the leader.
Trade Pact ...

When resources become unbalanced within the DOMINION, the DOMINION might want to trade abundant resources with another DOMINION for lacking resources. In order to enable such trade, the DOMINION must set up a trade pact. Having a pact between two DOMINIONS diplomatically enables them to trade, but does NOT however guarantee that they have the game elements necessary to facilitate the trade.

Exactly what game elements require a trade pact is specified elsewhere (see 2.2.6).

One feature of the trade pact include:

- Allows for continual trade versus one time transactions //TODO How does that work exactly?

- When creating or modifying the trade pact trade prices can be set at a increased or decreased market value.

Two DOMINIONS at war with each other CAN trade, but CAN NOT have a trade pact. This allows for bribery amongst players of opposing views. //TODO again, how exactly to put this into effect?

Embargo ...

To add another dimension to the game, we include trade embargoes. An embargo is binding in the game, i.e. prevents any trade between two DOMINIONS. The purpose of the embargo is twofold. First, a DOMINION can place an embargo on trade between itself and another DOMINION. Secondly, a DOMINION can make an agreement with another DOMINION to place an embargo on trade with a third DOMINION. In the latter case, the agreement is made between the first two DOMINIONS (so that if the embargo is removed, it is known) and the third embargoed DOMINION is potentially left unknowing.

Q ⇒ knowing an ally of yours is trading with an enemy of yours should be a negative thing? Translation: Should trade agreements be publicly known?
2.3 Game Elements

<table>
<thead>
<tr>
<th>BUILDING type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY Center</td>
<td>- none -</td>
<td>1 CELL</td>
</tr>
<tr>
<td>Quarry</td>
<td>- none -</td>
<td>1 CELL</td>
</tr>
<tr>
<td>Construction Yard</td>
<td>CITY Center</td>
<td></td>
</tr>
<tr>
<td>Farm</td>
<td>Construction Yard</td>
<td>1 CELL</td>
</tr>
<tr>
<td>Stockpile</td>
<td>Construction Yard</td>
<td>1 CELL</td>
</tr>
<tr>
<td>Plantation</td>
<td>Farm</td>
<td>Serious COMMAND cost!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNIT type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laborer</td>
<td>Construction Yard</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Basic Buildings and Units

**Construction Yard**...

The construction yard is the basic structure required for building. If the player does not have a construction yard under his/her control, no buildings can be constructed except for the Quarry.

**Quarry**...

*Multiple Gain: Increased resource acquisition*

Grants resource acquisition

**CITY Center**...

The CITY Center is the main building for a player. Anything pertaining to the player’s realm that is not specific to another building will be manipulatable here.

Q ⇒ if possible, to break the clich, we could try and rid the game of a center piece like the CITY Center; just a conglomeration of functional buildings.

**Farm**...

*Multiple Gain: Increased POPULATION_MAX per Farm structure*

Increases POPULATION_MAX. This carries over to DOMINION (excess POPULATION capacity can be used by the DOMINION to support extra POPULATION).

**Stockpile**...

*Multiple Gain: Increased*

Q ⇒ Allows for storage of resources ... needed it? If income = outgoing resources then a stockpile might not be needed. Only to store reserves, perhaps.

**Plantation**...

*Multiple Gain: Increased POPULATION_MAX per Farm structure*

The plantation is an enhanced Farm structure. It will produce more food per time unit than the normal farm. Although expensive, its cost will be recuperated in the long run.
TODO: ... 

• Units (resources + **FOOD** + **POPULATION** + **COMMAND**)

• Stockpile can stock **FOOD** incase farms get destroyed?

• Plantation allows for the player to PLANT a single type of specialize crop? (see civ resources?)

• Should be something like the spy in stratego. It is imperative that you keep it hidden. It is very weak. If they enemy finds it and destroys it, then that is a GREAT loss. Could be a special infiltration unit or weapon.

**BUILD_TREE ...**

The **BUILD_TREE** pertaining to the player’s chosen **AVATAR** will determine which buildings will be available to the player at which point in the game. The **BUILD_TREE** for each **DIVISION** contains the same basic buildings (those with no prerequisites);

**TECH ...**

A **TECH** is a technology upgrade to better a game aspect that is already possessed by the player. Each **TECH** grants a unique feature to the player. Each **DIVISION** has its own unique **TECH** tree. ⁹ This easily ensures that a player is not able to obtain all the most powerful **TECHS** simultaneously. ¹⁰

Q ⇒ Is there an additional cost/requirement to obtain a **TECH**, e.g. gold, resources, **COMMAND**?

---

⁹If at any time in the game the player loses the **TECHS** that been researched or fails to meet the prerequisites to have the tech, those **TECHS** will be deemed inoperable and void the benefits.

¹⁰An alternative way to design the **TECH** trees would be to have just one general **TECH** tree with the prerequisites set so that a player can not obtain all the most powerful **TECHS** simultaneously. The first choice was taken out of simplicity and the idea that if an **AVATAR** is allowed to be in different **DIVISIONS** at some point in the future, that the tech trees are still separated, e.g. such as the style of the World of Warcraft [9].
2.3.1 REGIME

![Regime Build Tree](image)

### BUILDING type

<table>
<thead>
<tr>
<th>Building Type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guild/Syndicate</td>
<td>Construction Yard</td>
<td></td>
</tr>
<tr>
<td>Town Hall</td>
<td>CITY Center</td>
<td></td>
</tr>
<tr>
<td>Distribution Center</td>
<td>Town Hall, Stockpile ↑</td>
<td></td>
</tr>
<tr>
<td>Athenaeum</td>
<td>Town Hall</td>
<td></td>
</tr>
<tr>
<td>Keep</td>
<td>Town Hall</td>
<td></td>
</tr>
</tbody>
</table>

### UNIT type

<table>
<thead>
<tr>
<th>Unit Type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent/Spy/Sleeper</td>
<td>Guild</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3: Regime Build Tree**

**Q ⇒** Scout in combo with military (horses)?

**TECH ...**

- At certain tech level grant the ability to send anonymous messages.
- DOMINION espionage. At certain level be able to import spies into enemy CITIES in order to get a rough estimate of the total DOMINION stats. Something like Civ4 where you can see the persons military strength graph or TW where you can see their point total. Should however be a rough estimate.

**Q ⇒** Should be two leader positions: "A team of two, with one leader, is often the best use of minds." [4]
2.3.2 MAGIC

<table>
<thead>
<tr>
<th>BUILDING type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Town Hall</td>
<td>CITY Center ↑</td>
<td></td>
</tr>
<tr>
<td>Athenaeum</td>
<td>Town Hall</td>
<td></td>
</tr>
<tr>
<td>Institute of Magic</td>
<td>Athenaeum ↑</td>
<td></td>
</tr>
<tr>
<td>Cathedral/Sanctuary</td>
<td>Regime: Keep, Athenaeum</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNIT type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mage/Sorcerer(ess)</td>
<td>Institute of Magic</td>
<td></td>
</tr>
<tr>
<td>Healer/Cleric/Priest(ess)</td>
<td>Cathedral/Sanctuary</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Magic Build Tree
2.3.3 MILITARY

**Figure 5: Military Build Tree**

- **Foundry**
  Add a description here...

- **Headquarters**
  The headquarters is player’s center for commanding his/her vast army. If they player does not have a headquarters in play, the play does not have control of the army for attack. The units will defend, but no troop movement, battle plans, etc will be possible without the headquarters.

- **Barracks**
  Multiple Gain: Increased unit build speed.
  Add a description here...

- **Manège**
  Multiple Gain: Increased unit build speed.
  Add a description here...

- **Armory**
  Multiple Gain: Increased research speed.
  If the player loses all armories in the game, all research technology is lost?
<table>
<thead>
<tr>
<th>BUILDING type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foundry</td>
<td>- none -</td>
<td></td>
</tr>
<tr>
<td>Headquarters</td>
<td>CITY Center</td>
<td>↑</td>
</tr>
<tr>
<td>Barracks</td>
<td>Foundry</td>
<td></td>
</tr>
<tr>
<td>Manège</td>
<td>Plantation</td>
<td>↑</td>
</tr>
<tr>
<td>Armory</td>
<td>Headquarters, Foundry</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNIT type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pikeman/Spearman</td>
<td>Barracks(??)</td>
<td></td>
</tr>
<tr>
<td>Archer</td>
<td>Barracks(??)</td>
<td></td>
</tr>
<tr>
<td>Swordsman</td>
<td>Barracks(??)</td>
<td></td>
</tr>
<tr>
<td>Cavalry/Knight (light/heavy)</td>
<td>Barracks(??)+ Stable(??)</td>
<td></td>
</tr>
<tr>
<td>Reservist</td>
<td>Barracks(??)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: Military Build Tree

TECH

- Heavy/light cavalry (HC more expensive to make)
- Horse archer: Since using a bow requires a horseman to let go of the reins with both hands, horse archers need superb equestrian skills if they are to shoot on the move.
2.3.4 ENGINEERING

![Engineering Build Tree](image)

<table>
<thead>
<tr>
<th>BUILDING type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foundry</td>
<td>- none -</td>
<td></td>
</tr>
<tr>
<td>Trench</td>
<td>- none -</td>
<td></td>
</tr>
<tr>
<td>Curtain Wall</td>
<td>Quarry</td>
<td></td>
</tr>
<tr>
<td>Workshop</td>
<td>Foundry, Yard</td>
<td></td>
</tr>
<tr>
<td>Gatehouse/Fort</td>
<td>Workshop</td>
<td></td>
</tr>
<tr>
<td>Tower/Bastion</td>
<td>Workshop</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNIT type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ballista¹</td>
<td>Workshop</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TECH type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moat</td>
<td>Trench</td>
<td></td>
</tr>
<tr>
<td>Spike Filled Moat</td>
<td>Moat</td>
<td></td>
</tr>
<tr>
<td>Fortified Wall</td>
<td>Curtain Wall</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7: Engineering Build Tree

¹This is a static unit. I must be placed on a map location and can not be moved afterward. It could, however, be disassembled and reassembled elsewhere.
**Foundry** (*see MILITARY: Foundry*)

**Trench/Moat** . . .
Add a description here...

**Outer Curtain Wall** . . .
- **Multiple Gain:** Will add defensive capability for each map **CELL** it is built in.
- Add a description here...

**Workshop** . . .
Add a description here...

**Inner Curtain/Fortified Wall** . . .
Add a description here...

**Gatehouse/Fort** . . .
Can only be placed on a map location that has been walled.

**Turret/Tower/Bastion** . . .
Add a description here...

**Ballista** . . .
Add a description here...

**Q ⇒ Workshop provides advanced build crew? TECH: organized labor?**
2.3.5 DOMINION

DOMINION game elements are only obtainable through the cooperation of players that have formed a DOMINION. They belong to the DOMINION. If at any time the DOMINION is not in possession of the requirements for the DOMINION game element, the game element is rendered inoperable (Units can be automatically disbanded).

Q ⇒ Should only tribal leaders be able to order them?

<table>
<thead>
<tr>
<th>BUILDING type</th>
<th>Prerequisites</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stronghold</td>
<td>Keep, Workshop</td>
<td></td>
</tr>
<tr>
<td>Treasury</td>
<td>Stronghold</td>
<td></td>
</tr>
<tr>
<td>Citadel</td>
<td>Stronghold, Cathedral,</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8: DOMINION Buildings

TODO: ...

- Tribal Captial cities?
- Roads are pretty simple; click on a hex square of the map to upgrade all roads in that hex. All traffic going through that hex will benefit from the roads built. Roads can be upgraded to higher and higher levels.
- In order to restrict which DOMINION buildings can be built when, AVATAR levels or COMMAND of largest player can be used as a requirement.

Q ⇒ In order for the player to have "personal combat" before joining a DOMINION, that means each player from each DIVISION needs at least some kind of unit to be able to do combat with. But, the build trees are not setup as such. Partisan basic fighting unit for MIND? No, they have spies from the guild. What about Engineering? They need a basic fighting unit. tech: Belligerent Laborer/Partisan.

Q ⇒ For each building you need to go through and determine that the player can do *IN* that building. What’s the point of having it?! And what happens if the player has more than one of each type of that building?

TODO: ...

- The rich can buy mercenaries
- Scaling ladders
- Assume one functioning castle for every 50,000 people.
2.4 Core Mechanics

Q ⇒ Make the game multi-tiered somehow, like the stock market. Some people can play the game on a micro (per minute) or nano (per second) level and others on a macro (per hour) level. If you give the player the ability to control an individual battalion of troops for example, he/she could spend time and scout around manually or, perhaps, try and slip through enemies lines without being seen.

2.4.1 Game Progression

TODO: ...

• The factor of time in the game determines the speed of progress. The relative build times of different game aspects determines how fast a player can gain certain game elements initially. To balance the game effectively, relative build times must guarantee that different divisions are of equal strength until players join a dominion, where this balance is not important any longer. We must make sure players in each division are able to survive for a certain amount of initial game time.

• How to make sure the player can’t obtain an exponential growth rate through massive build crew?

Time  ...

It is important to realize the time frame could span a few months, since the game is based on real time and time costs are rather lengthy so as to be proportional to the time frame.

Population Growth  ...

With farms being the number of Farms belonging to a single city, level the level of each individual Farm and a function, food(level), the amount of food produced at a certain Farm level, city_food for that city is defined as

\[ \text{city}_\text{food} = \sum_{f=1}^{\text{farms}} \text{food(levelf)} \]

giving a player’s total sustainable population of

\[ \sum_{c=1}^{\text{cities}} \left( \sum_{f=1}^{\text{farms}_c} \text{food(levelf)} \right) \]

\[ 11 \text{This is probably only of use as a statistic, since city_food is our limiting factor.} \]
TODO: ... 

- If the POPULATION slowly increases to CITY FOOD and we want to move an army to a CITY to be stationed there, then it must be sustained. It must be supplied by the sending CITY (the resource net plays a role?) or the troops can be placed there permanently causing them to be added to the CITY’s POPULATION. But, I believe this is already mentioned elsewhere.

- Each unit can take exactly one POPULATION/FOOD slot, large units will consume lots of COMMAND instead of lots of FOOD.

- 1 hex Farm = 30-120 food (lvl 1, more levels?)

- We must determine which Farms belong to which CITIES

- If ENGINEERING supplies the Farms, then REGIME must supply the resource net to distribute the FOOD. Resource net can also be responsible for supplying armies at long distances.

When a player has increased the CITY FOOD above the current CITY’s POPULATION, the POPULATION shall slowly start to increase. We therefore need a function which represents this growth rate. It will take the form of

\[ P(t) = P_0 e^{kt} \]  

with \( k \) the relative growth rate, e.g. 0.02 for 2\% and \( P_0 \) the POPULATION at time \( t_0 \). If the POPULATION ever exceeds the CITY FOOD, the POPULATION will slowly start to decline. We can reuse equation (1) above, but use a negative constant \( k \) to give us the decay function. When a player has reached his/her CITY FOOD limit, both equations above will come into effect simultaneously canceling each other out.

**Resource Growth ...**

If the player has quarries Quarries in play, each with a level equal to \( \text{level} \) and each producing an periodic resource income based on a function \( \text{income}(\text{level}) \), then the players total resource income at time \( t \) is equal to

\[ \sum_{q=1}^{\text{quarries}} (t \cdot \text{income}(\text{level}_q)) \]  

Q ⇒ The COST to raise a single Quarry in level is based on some function, \( \text{cost}(\text{level}) \), with \( \text{cost}(0) \) being possibly substantially higher than subsequent leveling costs. We can manipulated this function to be linear, logarithmic or even exponential. We can further manipulate the flow of possible player resources by regulating the TIME COST for each level of the Quarry. And, since a Quarry is a building like any other, it requires COMMAND in order to build.

Q ⇒ max quarry levels for a player at time \( t \) (build times), difficult eq!
The total cost of the Quarries

$$\sum_{q=1}^{\text{quarries}} \left( \sum_{l=0}^{\text{level}_q} \text{cost}(l) \right)$$

is always less\(^{12}\) than the total resource income of equation (2).

\(Q \Rightarrow\) If we make the cost of the quarries at higher levels increase proportional to the gain in resources, then we keep the total resource income linear?

Limiting Factors \(\ldots\)

If we let units and buildings be the total number of units and buildings under the player’s control, then the following must always hold at any given time in the game

$$\sum_{u=1}^{\text{units}} \text{COMMAND}_u + \sum_{b=1}^{\text{buildings}} \text{COMMAND}_b < \text{COMMAND}$$

A CITY’s POPULATION is defined as

\(\text{COMMISSIONED} + \text{INHABITANTS} + \text{AVATARS}\)

with

\(\text{COMMISSIONED} = \text{BUILD\_CREW} + \text{TROOPS} + ???\(^{13}\))

and

\(\text{POPULATION} \lesssim \text{CITY\_FOOD} \equiv \# \text{ Farms} \ll \text{COMMAND}\)

\(Q \Rightarrow\) Although not a favorable choice, we can limit the building of Quarries and Farms by having a limiting factor on the possessing hex squares on the map. I would tend to avoid this and just work with the limiting factor of COMMAND.

\(\text{total resources income} \equiv \# \text{ Quarries} \ll \text{occupied CELLS}\)

\(^{12}\)Or, almost always less, because players might get boosted by other players in the game.

\(^{13}\)Future developments might include other professionals or specialists.
2.4.2 Combat

TODO: ...

- A hasty retreat could cause greater casualties than an organized withdrawal, because the fast cavalry of the winning side’s rearguard would intercept the fleeing enemy while their infantry continued their attack. In most medieval battles, more soldiers were killed during the retreat than in battle, since mounted knights could quickly and easily dispatch the archers and infantry who were no longer protected by a line of pikes as they had been during the previous fighting.

- no survivors... no combat info?

Attacking ...

- Can you send an attack force to attack an incoming force!? Yes, attack noble train! Let allies be able to attack incoming attacks from other players too.

- Tribe members can defend another member using their troops, but can not attack with them.

- Guerilla warfare tactics. Sabotage.

- Assassination plots against players or villages?

- Sabotage requires COORDINATED precise timing of more than one player’s attacks to get a success. The defender could possibly notice the sabotage and try to knock out just enough of the attacks to cause a failure. (Ex: Two(or more) keys must be turned simultaneously in order for the safe door to open.)

- How about instead of all this fake attack stuff [1], you just don’t tell the player which village it will hit? Either that or make that you can send them easy or something. Come to think of it? What is a fake attack anyways?? Like in the middle ages there would be a giant cross indicating an attack from one lowly unit coming to knock on the city walls. How about a mass of units assembled outside a village that just don’t attack? Or just sit and wait for the opportune moment?

- Units that can attack enemies in secret. How about units that start an uprise in the other person’s villages? reduce Loyalty.

- Distance limits for attacks.
Defending . . .

- Defense can consist of active duty troops and reservists that can be called upon by the commander.

- You should be able to setup a front line of defense and also a secondary line of defense on the map. (Interface that actually allows you to DRAW a line of defense where the troops should stand?)

- Espionage and terrorism, indian night raids to attack enemy when army is too strong?

Flanking . . .

An army is located in a particular map CELL. Say, for example, this army is then attacked from one direction by an opponent, say from due north. If another army attacks the same defending army from due south while the army is still in battle from the previous attack from the north, the southern attack can be considered a FLANK attack. This multiple attack requirement alleviates the problem of having to determine each armies facing direction and makes flanking more difficult. (It is also a little more logical. An army not in battle is alert to incoming troops from any direction.)

2.4.3 Field Battle

We would like to make good use of the map for troop movement. We have two scenarios that need to be addressed. The first, where a player would like to station troops extremely close to his/her opponent, but wants to provoke the opponent to make the attack and the second, a basic attack by one player on the next. To accommodate for the first case, a player can place his/her units in the map location directly next to the target CELL. At any point, either party can take the initiative and attack making the attack fall in the second case. Namely, where the attacker is the one taking the initiative and the defender can choose the most optimal unit to defend with.

Q ⇒ Does the defender get to keep choosing most optimal defense in subsequent rounds as well?

Q ⇒ standing one CELL away allows a player to *see* the FIELD troops of the other before the attack allowing him/her to judge better what to send in first.
<table>
<thead>
<tr>
<th>UNIT type</th>
<th>Range</th>
<th>Speed</th>
<th>Armor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pikeman, stance</td>
<td>zero</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Pikeman, thrown</td>
<td>*see Archer</td>
<td>*see Archer</td>
<td>*see Archer</td>
</tr>
<tr>
<td>Light Cavalry</td>
<td>zero</td>
<td>racing</td>
<td>leather</td>
</tr>
<tr>
<td>Knight</td>
<td>zero</td>
<td>swift</td>
<td>plate</td>
</tr>
<tr>
<td>Mounted Archer</td>
<td>moderate</td>
<td>racing</td>
<td>leather</td>
</tr>
<tr>
<td>Swordsman</td>
<td>zero</td>
<td>slow</td>
<td>plate</td>
</tr>
<tr>
<td>Archer</td>
<td>moderate</td>
<td>moderate</td>
<td>chain</td>
</tr>
<tr>
<td>Longbowman</td>
<td>long</td>
<td>moderate</td>
<td>leather</td>
</tr>
<tr>
<td>Virtuous Healer</td>
<td>moderate</td>
<td>moderate</td>
<td>none</td>
</tr>
<tr>
<td>Wicked Healer</td>
<td>moderate</td>
<td>moderate</td>
<td>none</td>
</tr>
<tr>
<td>Mage</td>
<td>long</td>
<td>moderate</td>
<td>none</td>
</tr>
<tr>
<td>Catapult</td>
<td>long</td>
<td>slow</td>
<td>—</td>
</tr>
</tbody>
</table>

Figure 9: Combat Stats

Range: zero < moderate < long

Speed: slow < moderate < swift < racing

Armor: none(none/leather) < light(leather/chain) < moderate(chain/plate) < heavy(full plate)

Unit Specials . . .

Pikeman (stance): Defends effectively against incoming cavalry, light and heavy, but rather ineffective against Mounted Archers since they will ride along the pike wall and shoot arrows.

Pikeman (thrown): Threat as if they were Archers, except they throw long pikes/spears instead.

Light Cavalry: Can be used to attack Mounted Archers effectively because of their speed.

Knight: Due to the sheer weight and armor can almost ride over unmounted opponents.

Mounted Archer: The basic tactic used by most horse archers was the ‘shoot and run’, where the horse archers would come in range of the enemy lines, shoot, and retreat if they were chased or came too close to the enemy infantry or heavy cavalry. This was brutally effective against heavy infantry or heavy cavalry that were well-armed but couldn’t reach the mounted archers. The best way to counter horse archers was to use light, fast cavalry that could reach the archers, or longer range missiles where retreating didn’t stop casualties. Horse archers were also used to attack supply lines and harry retreating troops [10].
**Swordsman**: This is a heavily armored fighter, making it effective against lightly armored units. And, as always the disadvantage is that of speed.

**Archer**: The only advantage of this unit is their cost. They may be upgraded to Longbowman, if the TECH is available.

**Longbowman**: "The pike and the longbow put an end to the dominance of cavalry in European warfare, making the use of foot soldiers more important than they had been in recent years." [7]

**Virtuous Healer**: . . .

Spells:

- Healing, the Healer must be located close to the target (range? zero/moderate?) in order to use the spell. Heals the target of a certain amount of damage.

- Sanctuary, temporarily shields the target. The shield can absorb X amount of damage before being dispelled.

- Turn Undead, instantly buries a number of Animated Dead from the Wicked Healer spell.

**Wicked Healer**: . . .

Spells:

- Rot, causes the effected units to slowly rot to death. (range? moderate/long? Do the effects wear off after battle?)

- Fear, causes the target units to flee in fear for a certain amount of time.

- Animate Dead, allows the caster to animate dead units back into combat.

**Mage**: . . .

Spells:

- Fireball, this is the mage's standard distance area attack

- Blink, allows the mage to instantaneously move a group of units to the opponent for immediate attack on a specific unit group. (range? moderate?)

- Rage, temporarily increases the size and strength of the target unit group, but forces the unit into a blinding rage. (disadvantages?)

**Catapult**: Long range, slow firing siege engines which cause a great deal of damage in an area of effect. Extremely fast units are needed to reach them without being taken out.

**Q ⇒** The English longbowman used a single-piece longbow to deliver arrows that could penetrate contemporary plate armor and mail [7].

45
There are two possible models that can be explored for the combat engine. Prototyping and mathematical evaluation will reveal which is better suited.

**Model 1: The Pit** Each army is stationed on opposite ends of a region. In the middle, which we will refer to as "the pit", the attacker must decide which unit he/she wants to send into the pit to attack the opponent. The defender can then choose which unit he/she would like to defend with. Either by sending another unit into the pit to meet the incoming attackers or to wait until the attacker reaches his/her front line. Ranged units can fire into the pit with or without be required to move into the pit themselves. If long ranged units are placed in the pit, then they could possibly have enough range to target any units in entire opponent army.

![Figure 10: The Pit](image)

**Q ⇒** If there is only one pit. How to handle different movement types? Racing horses should reach the other side without actually being the pit, but briefly compared to the slow walking Swordsman.

**Q ⇒** How do we handle the different ranges. If we have one pit and 2 sides, what are the lengths of the different ranges?
Model 2: Unit Slots  If we visualize that each of the two player’s units are lined up in the order of attack and defend facing each other, with free space in between. If each unit were to take up a slot and the free space were divided into a number of slots, then as the attacking army moves to attack they gradually move through the slots toward the defending army. Distance can be calculated by range in number of slots. When the two armies meet the colliding layers can be calculated and the distance attacks.

Figure 11: Unit Slots

Q ⇒ How is movement speed handled here? Just ignored and assumed that it makes the unit harder to hit and that the speed is calculated into the attack and defenses against other units?

TODO: ...

• Question before ordering an attack will be
  • What is the desired order of the units attacking?
  • Which group of units shall be targeted for healing? (Healers must be close to this unit in the order) How shall be targeted for protection?
  • Which group of units shall be Blinked to the opponent to attack which group of units?
  • other spells casts?

• Should it be possible to order ranged attacks on your own units (when they are possibly in melee with the opponent?)

• Real time scheduling: player commands which unit and how many to send in, in real time. Would require a movement speed for units to cross the field, so that a stream of units can be created.

• all units that are not in combat can be used to accept an incoming flanking attack
2.4.4 Castle Battle

Q ⇒ dirt fillers for Moat?

Figure 12: Castle battle, troop order.
2.5 Infrastructure

2.5.1 In Game Mail

The mail system is known to be one of the vital communication mediums for this type of game. The mail system mustn’t necessarily function exact as classical POP3 Email. There are a number of features that can influence the game.

The following summarizes the features of the desired mail system.

- Every mail must have the standard features for mailing to maintain the consistency with what people expect from a mail client. Each mail should have a "To" address field, a "Bcc" address field, a subject and a body. (The BCC field allows for anonymous mailings, which might be only granted at later stages in the game.)

- The sender should be able to initiate a conversation one or multiple participants. There should however be a way for each participant remove himself/herself from the conversation. The list of multiple participants can be the entire DOMINION members, a manually entered list or a pre-made custom mailing list group.

- Each mail should be forwardable.

- Many new web-mail systems currently display mail as a continuing list of messages by all parties. This is a highly desirable form of communication in game. Each participants answer will be concatenated to the running list in the order they were received by the mail system.

- Each mail should be marked as unread initially by the system when it arrives at the destination and should allow the receiver to mark it as read/unread again as well.

- The sender of the mail will see a flag when the receiver has opened the mail. This plays a particular roll in the game, as you can ascertain a persons login times then. (As with anonymous sending, allowing the receiver to block the read notification can be granted at later stages in the game.)

- Another game feature that can be granted to the players at a specific time in the game is email scheduling; allowing the player to create an email, but have it delivered at a later time/date.

- Other mail aspects: An address book, ability to block a sender, different mailboxes for sorting and a markup language to allow for in message images, urls, etc...

Q ⇒ Possible TECHs: anon email, email scheduling, block read notification
2.5.2 The Event Record

The Event Record is a record of all events that are visible to a specific player. It will contain global events (possibly marked global on the record) and private events which are personal for each player. The system should keep track which events each player has read.

Global events are happenings that are reported to the entire game world. Prominent happenings will automatically generate a global event on each player’s Event Record. However, leaders of DOMINIONS can choose whether certain events will produce a global event. For example, if a leader would like to declare all out war on another tribe, they may do so by having the system announce the declaration of war as a global event.

Private events are those happenings which pertain only to a specific player.

Private Events:
- scoutings

Global Events:
- system events
- arms/trade pacts
- major war results: if a prominent politician is sent for talks, then it will be published; if the leader does something prominent, it should be announced automatically; some should be unstoppable game notifications; some, the leaders have an option to make certain events public.

The Event Record should contain a filter and a search facility so that players can easily wade through many events reports and have displayed only those that he/she finds relevant.

2.5.3 Game Clients

Downloadable Client

This client interface shall possess the greatest degree of implemented user interface features. The cost of this interface is that users must download and install this client to their personal computer. If all game features are implemented, this interface should be highly graphical, allow for a user-friendly interface that reduces the burden of issuing game commands and provide the greatest speed.
Browser Interface

Due to the limitations of what can be accomplished efficiently in a browser, the browser interface will most likely have a reduced feature set, especially when it comes to graphics. The compatibility, download size and execution speed must be considered as a priority with this interface.

Mobile Access

In order to cater to the growing telecommunications market, the game should be developed with mobile technologies in mind. Since the game runs over an extended period of time and a player is always available in game, mobile technologies will allow people periodically check on their game status or quickly issues some commands while they have spare moment in their daily schedule.
3 World Design Document

In order for the players to be able to control where structure and troop place is desired, the game world must have a way to designate a specific location in the world. The coordinate system this game shall adopt is that of the “hex map”, which is comprised of enough hexagonal cells to fill the game world.

Q ⇒ Make the coordinate system visible to the players? Try to avoid!

Since hexagon grids have not been standardized, there is no nomenclature for the different ways to assign coordinates to the grid pattern. We describe below what a few others have already defined as well [2].

Each hexagon cell is aligned with a vertex pointing due north. When all cells are tiled next to each other on the map, each cell will have a unique coordinate \([x_l, x_r]\). If all the hexagon cells on the top row of the map are numbered from left to right starting from zero to a certain \(X_{MAX}\), then the XL coordinate of a cell can be determined by drawing a symmetry axis that is perpendicular to the cell/hexagon edge that lies immediately to the left of the vertex lying due north and then following this line to a cell on the top most row. The cell number of this top most cell becomes the \(x_l\) coordinate. The \(x_r\) can be determined in a similar way, but with the symmetry axis drawn perpendicular through the edge to the right of the northern vertex. There is, however, a boundary condition to the given rule above. Cells will exist on the map where tracing the diagonal through the cell edges never reaches the top most row with the range zero to \(X_{MAX}\). In order to be able to obtain the coordinates for these cells, it must be assumed that the map wraps to the opposite side and the modulus \(^{14}\) can be used to obtain the coordinate. The topmost and bottommost row can be wrapped as well so that the world object becomes a torus.

Q ⇒ Because of the already existing Euclidean system, is it handier to use a similar representation? No, we shall say true to the best hex representation.

\(^{14}\)The modulus defined as the congruence relation \(\mathbb{Z}_n, n = X_{MAX}+1\), e.g. \(-1 \equiv 6 (\text{mod} \ 7)\)
Figure 13: $[xl, xr]$ computation.
Each CELL of the world map has the following characteristics:

- **TERRAIN TYPE**: Values: Grassland, Forest, Rocky, Water, Swamp or Desert. Terrain type determines unit movement speed on each spot, if at all possible, or which prerequisites are required. Some structures are forbidden on certain types of terrain.

- **MOVEMENT PENALTY**: Determines how much trouble a player has to move troops through the area. This allows for different gradients of Rocky TERRAIN for example or even completely impassable areas.

- **ELEVATION**: This value determines at what height the CELL lies according to sea level. Negative numbers indicate levels below sea level and could be filled by natural water. Steep gradients should indicate impassable cliffs or drop-offs.

- **RESOURCES**: Indicates what resources can be found at the location.

- **ROUTING INFO**: If we use multi-tiered A*, then this will contain needed routing info as the game progresses. If roads are incorporated, then travel times will decrease creating waypoints.

**Q ⇒** Elevation gives height advantage in combat?

**Q ⇒** Routing info for multi-tiered A*?

Before any players enter the game. The entire game world shall be generated with terrain. Once the world is generated, it will not change. Game elements will be placed on top of the game world, but the world itself, will not change (unless we later build in a way to modify terrain, e.g. redevelop swamp to more hospitable plains).

**Q ⇒** How much land area does one hex represent? Does it matter? For population determination maybe? How about each hex being 1 sq KM? Uppsala city is then 5 sq KMs. To determine the area of a hex, multiply its width by 0.9306049, and square the result. Thus, if your game-map has hexes 30 miles across (in diameter), each hex represents about 780 square miles (and it’s a convenient size for travel-times since 30 miles is a good rule of thumb for a day’s road travel on foot or horseback) [5].

**TODO:**

- Terrain resources that are bonus to standard ones? Civ style resources?

- Fog of War: won’t be useful after a while, because people can trade out maps on forums. Rule: No mapping tool available until entire world has been uncovered?

- Map not placed north on actual north, so land discovered can be any direction. Maps between players and tribes won’t match in the beginning. Scouting happens in blotches?
4 Designer’s Notes

4.1 Features

TODO: ...

• progressive resources. Resources needed at different stages of the game are different. In the beginning, coal, after a while iron, etc...

• Similar, but possibly better idea to ours: "Again, I’m going to steal a concept from Go that I’ll call capturing a territory.", "Space Crack: Using Planets to Build Territories" [http://lostgarden.com/2005/06/space-crack-using-planets-to-build.html],

• what is the advantage of a tribe deciding all to logon on at a certain time period?

• If leaders can gift villages of players that are inactive to other players, it gives two bonus’. First, players are not spending loads of time taking up dead weight in the form of friendly players and, second, it increases action and battle conflict since players are not spending loads of time battling players that are inactive from the opponent [1]. The end product would be active players battling active player opponents.

• [http://en.wikipedia.org/wiki/Chevauch%C3%A9e] Specific tactics were ”a quick cavalry raid through the countryside with the intention of pillaging unfortified villages and towns, destroying crops and houses, stealing livestock, and generally disrupting and terrorizing rural society. Most of troops used in a chevauche during the Hundred Years’ War were made up of light horse. ... it worth a pillagers time to take one hostage, instead of just killing them. Therefore it is not surprising that the peasant villages put up whatever meager resistance that they could.

• River and sea travel proved to be the easiest ways to transport supplies. During his invasion of the Levant, Richard I of England was forced to supply his army as it was marching through a barren desert. By marching his army along the shore, Richard was regularly resupplied by ships travelling along the coast.

• build in a sort of ”farming”[1] so that players that work at it can get ahead

• reforming the tribe takes times. People are not given the time to learn the game well before they get nobled.

• if the game is correct... you should have nothing but scattered villages in the game in the beginning and after the game progresses, you should see more and more blobs gradually being formed and gulping villages in the blobs. Civ4 style.

• Word: agronomy, noun, the science of soil management and crop production.
• STARTER_COLONY=Locale -¿ Commune -¿ Colony -¿ District -¿ Metropolitan -¿ Commonwealth -¿ Region -¿ County/Dominion -¿ Province/Kingdom -¿ State/Realm -¿ Country/Nation/Empire -¿ ...

• should high skilled players get a handicap? In exchange for what benefit? Rank bonus? pg.15

• The rich get richer... the poor stay poor. Go against it?

• MAJOR DESIGN FLAWS of TW [1]: Lack of scheduling for troop movements && No warning if tribes members are being attacked && No easy way to get rid of inactives. Tribes must continually spend energy cleaning up the mess of leaving players (noble inactives). Avoid this!

• You can pay for stuff in time or money (resources). Time can be construction time or the players time spent working to get something done.

• Scalability: Building, Attacking, ...

• Don’t make it so easy to knock out a player. Should be able to keep playing, just not maybe in such a good tribe or capacity. Let players live in their own little world, struggling maybe.

• "Recipes you create yourself are the best" (templates bad)

• Make sure and consider what happens in the odd event that villages get lost and there is too much POPULATION to support. Ex: building population needed is greater than the number of farm slots. Buildings become unoperable?

• Troops can have different speeds, but will build fatigue making them less efficient. Charge at short distances? Or don’t build that in? Just assume it is calculated in the battle resolution?

• You can cancel orders close to home within a certain amount of time without a problem. You can cancel orders far from home if you can send a scout to that army in time. Make the same interface? click army, cancel order, choose scout from which location or closest location to run. Sending a scout to your army allows you to change the command. The scout is also an "army", and so it should be possible to send a scout to the scout to cancel the canceling. Essentially. You can change the command of your army, but it will take time for the order to get there and time for the order to execute.

• You can start as many villages as you want in a commune. There should be a DISadvantage to starting them. It should be part of the game to know exactly how many villages to have under control? This means a diplomat can start as many villages as he/she likes as long as the resources are there?

• More action! (If the game is supposed to reward online activity. Why not collective online activity of the tribe, NOT each individual member’s activity. A tribe with a few very online players would be equal to a tribe with LOADS of very little online.
• Take over villages by creating uprisings and disturbances. Another villages influence could be enough to take over another village. Political warfare. The law branch should have just as much potential as the war branch? Maybe not.

• Does having an old village give a bonus?

• Transfer troop populations to other villages.

• Farmed villages are strategic (Dawn of War) strategic points like Dawn of War

• Troop experience? Heroes characters (Warcraft III)?

• Monetary system to be able to buy research and villages from others.

• How to solve the... I’m going to call my big brother on you aspect?!?! World persistence. Will limiting troop distance by food supply be enough?

• Resource overflow is placed in the tribe resource bank. Tribe community buildings and services built by the law/government villages.

• ? Leif idea: Different radius’ of troop support possibilities. At each different radius a different level of troops can be supported. If you stay within radius, then you can mass a certain number of troops without loosing grain. If you stretch too far, then you can sustain it, but your grain starts to deplete. Maybe even force it, but at the cost of loosing troops due to hunger and lower attack power?

• Fatigue: Builders: If you want to build day AND NIGHT, then you must have twice as many builders. Fatigue: Troops: Fatigue sets in beyond your domain of control/influence. Of course, there can be a tech to increase your distance. Actually, it shouldn’t be fatigue that limits troops, but food supply.

• You can mass an army at a certain rally point. Others will detect a large group of troops in the area. Possibly the closer they are too it, the more info they get. Also, the bigger the army, the easier it is to detect. The identity, doesn’t need to be given though. The army can be scouted to find out the identity and troop details. Of course, the army could have scouts for scouting defense as well to protect against this. Once troops have been assembled, they can be given group orders.

• A battle shouldn’t be over instantaneously. It takes time to kill each other. Proportionally to army sizes. Large defenses compared to very small attacking = very short. Large defense, large attacking = long battle.

• Land, sea, air and space different major tech branches.

• Military can take control of a resource point, but gathers can’t gather at a large distance. Keeps players from taking resources across the entire board. Forces a player to build infrastructure toward the resources required. That also means there must be some sort of infrastructure available that can be built to extend the realm toward a resource point.
• How about more tactical planning instead of calculating stupid attack times[1]?

• Make sure there is no advantage to playing two accounts on the same world? If they want to play two accounts, they should be able to? Is it possible to stop this?

• Should be a way to dedicate ALL Your production in ALL your villages to ONE product in ONE village. Why can’t I send all my builders to help out another village? Mythical Man Month limitation.

• No two units should have the same function. Heavy and Light of something, should NOT exist unless they have 2 different functions. Heavy and Light should be define in the tech levels. Light Cavalry should be level 2 perhaps, and heavy level 3.

• Dials that regulate automatic unit and building upgrading, should be able to stop a building mid build. Resources should gradually decrease instead of instant invest. (How does one steal resources then? Raiding of stockpiles might prove very lucrative or negligible profit)

• Possession of a farm village by the force of an army gives resources, but army fatalities through commoner rebellion, unless a nobleman is present. Greater the population outweighs the army present, the greater is the chance of troop loss because of rebellion.

• Actually openly declaring your tribes members is pretty stupid. Wouldn’t it make the game more dangerous if you didn’t know which friends and allies a player has? Allows for bluffing.

• Bonus on round numbered (5,10,15,etc) levels of buildings.

• Civilization style game play. Tech building. How about another dimension to the game. That someone can build a nation with strong diplomatic grounds, but less army. Maybe limit the support a player can get by the level of diplomacy? The higher the level the more support troops can be received. As game continues you should be less and less involved with the nitty gritty. Tech level should make it so that things start to be move to a more macro level of control.

4.2 Infrastructure

TODO: …

• if we want to build a game where people rotate being online to protect the DOMINION, then some kind of mechanism to allow players to schedule on a calendar when they will be online might be handy. The system could send emails when the number of players online is zero or next to zero.

• a list of vague legends (wins/defeats) is kept and is visible by all. Together with the public list of events.
• player’s should be able to define quite easily what they want a report/event notification on.

• the general forums are a way that people can have inter-tribal general discussions, but that is not specific enough since it doesn’t encompass only the leaders... there should be a forum SPECIFICALLY for those designated as Dukes(or the likes).

• each player should be able to make a list of personal allies (other players, NOT DOMINIONS) and enemies as well. Actually, this is handled in TW, by allowing the player to mark any other player on the map a certain color.

• need a way to mark a tribe/player as Do NOT attack and/or Target, even if there has not been a pact made. This would mean that "Enemy" is not pact, no?

• be able to sort data ... sort by village population for example.

• ability to move stuff up and down in the queues

• Event System(input), Web Server, Flash Interface, Notification System(output), Calculation Server(game Engine)

• terrain advantages. Cities built on top of a mountain have height advantage.

• leaders can turn on the option of communication channels. This means that a person must request a communication channel, which is to be accepted, before any communication can begin. This means someone can also terminate the communication channel and refuse to listen (accept requesters agents).

• Need something to keep track of war targets? Don’t know if it will be necessary in the new system tho.

• Leaders should be able to follow a members progress better. See building stats and troop totals at least?

• Tutorial on how to play. Certain games aspects should be reveal when the time is right.

• Forum or text medium should grant the power to have a check list

• READ OVERLOAD when reading forums written on by a large body of players. divisions?

• Time tag: [time gmt="-1"]1:00[/time]. Translates server time to local time for each user. If gmt attribute is not specified, the time is translated from the posters local time to server time. Then the time is translated again from the server time to each viewers local time.

• Voting.

• The game’s infrastructure should allow for different government types such as communism, democracy, dictatorship, etc...
• Scheduling window for attack proportional to distance attacking. You can schedule an attack 50 hours in advance if the target distance is 48 hours away. You can NOT schedule an attack 50 hours in advance if the target distance is 10 minutes away.

• Make it possible to assign different players to attacking than defense

• Stats: Until what time is the queue busy? What time will the warehouse be full?

• Game Master tools. Other players that can assist in keeping order in the game while playing. Which means you need a security hierarchy.

• Game should have 2 thingies instead of just reports: Reports AND Alerts. An alert the player can assign to get notification of something. A report is ALWAYS generated and available. A log duh.

• Total population, total stats.

• Certain events break a scheduled task. Ex: an ally grabs a villa that you have scheduled to launch an attack against.

• Village names should be quite resilient. Each additional name change exponential time longer to take effect? Villages should have an optional Banner that they can display? Or does having an old name give legendary bonus?

• Should be a limit to event scheduling... to promote online activity, a player must ok the event 12 hours prior to activation? *** One of the design goals should be that a player should be able to manage his/her domain by logging on every 12 hours.

• RSS event feed ... or perhaps Growl?

• User names and passwords http://en.wikipedia.org/wiki/Password_cracking#Salting

• Overall player rating can give access to ... experienced player worlds only

• Wiki? Good forum: Post ratings, user rights, etc... Chat. IM.

• Server time is a problem?!? Have one function to join all armies and launch?

• Much better group interfaces allowing HUGE tribes! Maybe have an automatic clustering system? Data mining :) Interface to break off a division of a tribe into an own tribe. Also an option to merge two tribes with verification checks by Dukes to ok the merge. "each player should control where is going”, true? Nah. They can quit after the merge. Majority merges. Minority are the exceptions.

• Village presets? Quick start(resources), offensive, defensive.... Make these player configurable? Make them available AFTER the first village? Be able to clone build one of your previously built villages?
• Better features for grouping of players making divisions within a tribe. Names: Armored division, etc...

• Scout information collection system. You scout a village. After that your info over that village gets updated with the latest info. It doesn’t change until the next scouting, but you can call up the info easily.

• If units can be there WAY in time, then you should be able to tell them to delay their arrival.

• The game engine processes a number of commands that do operations on a data set somewhere in the database. Isn’t this like a computer processor? Can we utilize the similarity somehow?

• set personal limits for soldiers. If the levels stay on the limits all systems in green. If they drop below levels, then systems in the red. And, the creation of units will commence to get back into the green?

• Only allied villages should be able to trade... this reduces violence in general because of mutual benefit.

• Allies should get notification of attacked villages in the alliance.

• Have a function to purchase with resources from another village. It transfers automatically and starts build.

• performance graphs for each player with comparisons.

• Some way to lay claim to a village. Show intent of invasion. So that they are aware you are nobling... Or just show the village icon burning so that people see it is under attack. Maybe that the other person can see a giant army on the map? Or maybe not... surprise. You are all of a sudden attacking a third party.

• Troop orders with a possible timestamp in the future. ex: I want to move 300 to that village at midnight.

• Should be able to permanently assign troops to a new village.

• Should be able to combine forces into one big army attack.

• Time should be a relative concept. All timestamps should be shiftable in the case of server failure or downtime or whatnot.

• Preorder SMS notification messages. You can then turn on SMS notification before leaving the game. notifications of attack or build chain completion.

• word: plebiscite

•
4.3 Interface

TODO: . . .

- Use google maps technique for mapping in web browser without flash.
- Make something like Warcraft III style army groups? Assign army and give it a macro or hotkey?
- Have document agreement facilities. One can make a document that is signed and keep in vault by both parties.
- Graphics versus Art!
- Word: Metropolis
- Reduce the amount of active buttons to the ones needed.
- Too many fucking numbers in the interface! Takes away from the game play.
- Lose the points on villages and make the picogram more descriptive of what the village is like. Let people guess or scout that the defenses are. But, how can you have rankings then?
- Have some kind of LOCK that locks a city as complete.
- "If you do write a BOT, please let us know. We are actually INTER-ESTED to know which bots have been made for the system and how well they perform."
- Always have an indicator on the map pointing to your home base.

4.4 Maintenance

TODO: . . .

- Financial purchased items a different server?
- Server/Software update/plugin capabilities
- Time is a relative thing... if you restore a backup then all of a sudden the server has started X hours later in the state of X hours previous.
References


