Dynamic Difficulty Adaptation for Heterogeneously Skilled Player Groups in Collaborative Multiplayer Games

Master Thesis
Miguel Cristian Greciano Raiskila
KOM-M-0539

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Dynamic Difficulty Adaptation for Heterogeneously Skilled Player Groups in Collaborative Multiplayer Games
Master Thesis
KOM-M-0539

Eingereicht von Miguel Cristian Greciano Raiskila
Tag der Einreichung: 31. Mai 2016

Gutachter: Prof. Dr.-Ing. Ralf Steinmetz
Betreuer: Christian Reuter

Department of Electrical Engineering and Information Technology
Department of Computer Science (Adjunct Professor)

Multimedia Communications Lab (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Master Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Die schriftliche Fassung stimmt mit der elektronischen Fassung überein.

Darmstadt, den 31. Mai 2016 Miguel Cristian Greciano Raiskila

Contents

1. Introduction **3**

2. State-of-the-art and Related Work **7**
2.1. Collaborative Gaming 7
2.1.1. Basic definitions 7
2.1.2. Characteristics and challenges 8
2.1.3. Additional research 9
2.2. Dynamic Difficulty Adjusting in Video games 9
2.2.1. Basic definitions 9
2.2.2. Challenges, advantages and disadvantages of DDA 10
2.2.3. Characteristics and Design Patterns of DDA 11

3. Concept of Dynamic Difficulty Adaptation in Collaborative Games **13**
3.1. Challenges of DDA in collaborative games 13
3.2. Distinct-role collaborative games 14
3.3. Classification of Dynamic Difficulty Adaptation Algorithms 16
3.3.1. DDA according to WHEN to detect 16
3.3.2. DDA according to WHAT to detect 16
3.3.3. DDA according to HOW to react 17
3.3.4. DDA classification summary table 18

4. Co-op Craft - the implemented game **19**
4.1. A brief background to StarCraft 19
4.2. The StarCraft II Galaxy Editor 21
4.3. Co-op Craft: Game concept 24
4.3.1. Game definition 24
4.3.2. Map layout and Gameplay 25
4.3.3. Description of Player characters (Heroes) 29
4.4. Dynamic Difficulty Adaptation in Co-op Craft 31
4.4.1. DDAA #1: Resources DDAA 31
4.4.2. DDAA #2: Respawning DDAA 33
4.4.3. DDAA #3: Milestone DDAA 34
4.4.4. Possible alternatives for DDA in Co-op Craft 37
4.5. Instructions to play Co-op Craft online 38
4.5.1. Installing StarCraft II in a computer 38
4.5.2. Playing Co-op Craft 39

5. ADappTowr - an alternative **41**
5.1. Game concept 41
5.2. DDA in ADappTowr 42

6. The Test **45**
6.1. Phase #1: Introductory Video 45
6.2. Phase #2: Playing the Co-op Craft game 45
6.3. Phase #3: Questionnaire 46

7. Results **49**
7.1. Numerical Data 49
7.1.1. Tables 49
7.1.2. Graphs 52
7.2. Analysis of Player and Team Performance 54
7.3. Questionnaire analysis 55
7.4. Encountered problems and possible solutions 56

8. Final Remarks **61**
8.1. Conclusion 61
8.2. Future Work 61

A. Glossary of terms **63**

Bibliography **65**

Abstract

This work focuses on the combination of two key concepts: Dynamic Difficulty Adjustment/Adaptation (video games adapting their difficulty according to the in-game performance of players, making themselves easier if the player performs poorly or more difficult if the player performs well) and Collaborative Multiplayer Games (video games where two or more human players work together to achieve a common goal). It considers and analyzes the challenges, potential and possibilities of Dynamic Difficulty Adaptation in Collaborative Multiplayer Games, which has to date been quite unexplored. In particular, it addresses the heterogeneously skilled player groups challenge: players with different skill levels play together in a video game, but how should the game adapt its difficulty if one player performs well when another performs badly?

We use previous research on Dynamic Difficulty Adaption in single player games to mold, define and classify general approaches to Dynamic Difficulty Adaptation in collaborative games. We then focus on a subgroup of collaborative games - distinct-role collaborative video games, where we believe there is a viable way to address the heterogeneously skilled player groups challenge.

To test our general approach to Dynamic Difficulty Adaptation in collaborative games we present a game that has been exclusively developed for this purpose: Co-op Craft. It is a collaborative game that uses the StarCraft II™engine and includes three Dynamic Difficulty Adaptation Algorithms. We analyze, justify and classify these algorithms and we also outline other valid alternatives. We had players from different gaming backgrounds test this game both with Dynamic Difficulty Adaptation active and with Dynamic Difficulty Adaptation disabled. We analyzed their performance in both cases and asked their opinion on the matter based on their experience. From the numerical data representing their performance and their impressions we then extract conclusions about our approach and also the potential of including Dynamic Difficulty Adaptation in popular collaborative games of today.

1 Introduction

Video games are a relatively recent phenomenon, having seen widespread use since only a few decades. The world of video games is rapidly and constantly evolving, and it is expected to be a very relevant area in the future. What started as a simple way of entertainment has come to even been cataloged as an art by some. Not only are people playing more and more varied video games due to the increasing use of mobile devices and the Internet, in our recent era video game development tools have become easily accessible and the average person can indeed design, create and publish a game. Some initially not so popular genres like serious games or cooperative games are receiving increasing attention, and there is more research being done on video games than ever before. Video games are also a great way to express creativity and imagination. Cutting-edge innovation has the potential to lead to commercial success, as the entertainment role of video games is still the predominant role.

When producing a video game, the designers hope for it to be successful, and while a video game's success can indeed be defined in many different ways, most video games are defined as "successful" if they sell well. Large amount of sales are the consequence of the game appealing to a large crowd, or in other words, the game is entertaining for a lot of people. With the advent of the Internet games have even begun to include micro-transactions: small additional payments that users of a game make in order to download and access additional content developed after the game's release. If a game's micro transactions keep on selling well after its release it is also a good sign of commercial success.

A key and tricky element in a game's entertainment is its difficulty and its learning curve: the game should neither be too easy nor too difficult, otherwise the interest is lost. However, the audience is heterogeneous: people are different, and different players will have contrasting skills and motivations when playing games. How can designers cater for this diversity in order for their video game to be entertaining to the widest crowd possible? The most popular approach is to introduce static levels of difficulty in their game (for example: easy, medium and hard). An alternative and not yet very extended approach is to dynamically adapt the difficulty of the game, a concept known as Dynamic Difficulty Adaptation and into which we will delve deeply in this work.

Another key concept in this work is collaboration in video games. Since the beginning there has been both single and multiplayer games, and in the vast majority of multiplayer games humans play and compete against each other. However, another genre of multiplayer games, commonly known as "co-op games", has seen significant growth in the past few years, to the point that some co-op games have become extremely popular. In co-op games two or more humans work and play together as a team to achieve a common goal. An example of popular co-op games is games belonging to the relatively new MOBA game genre. **MOBA** stands for "Multiplayer Online Battle Arena", is a subgenre of **RTS** (Real-Time Strategy) games, and defines games where teams of human players control different hero characters and push along weaker allied computer-controlled creatures to progress in a map and eventually destroy the enemy's main structure to win. Humans in the same team work together to defeat the opposing human team. Playable heroes have different abilities and roles, so teams aim to seek good synergy and combinations between them, but the team members also must use skill (casting spells, engaging battles...) to effectively perform their assigned role.

Let's look at the numbers of League of Legends™(commonly abbreviated to LoL), one of the most popular MOBAs. As of 2014, over 67 million players battle it out in LoL every month, more than 27 million people play at least one game of LoL every day and over 7.5 million players simultaneously play LoL during daily peak hours. [1] Even the 2012 numbers were astonishing: [2]. Tassi [28] analyzes the significance of these numbers in his article. Interestingly enough, LoL is free to play, i.e. anyone can download and play it. However it includes micro transactions that allow players to buy new "skins" or

[1] http://www.riotgames.com/articles/20140711/1322/league-players-reach-new-heights-2014

[2] http://majorleagueoflegends.s3.amazonaws.com/lol_infographic.png

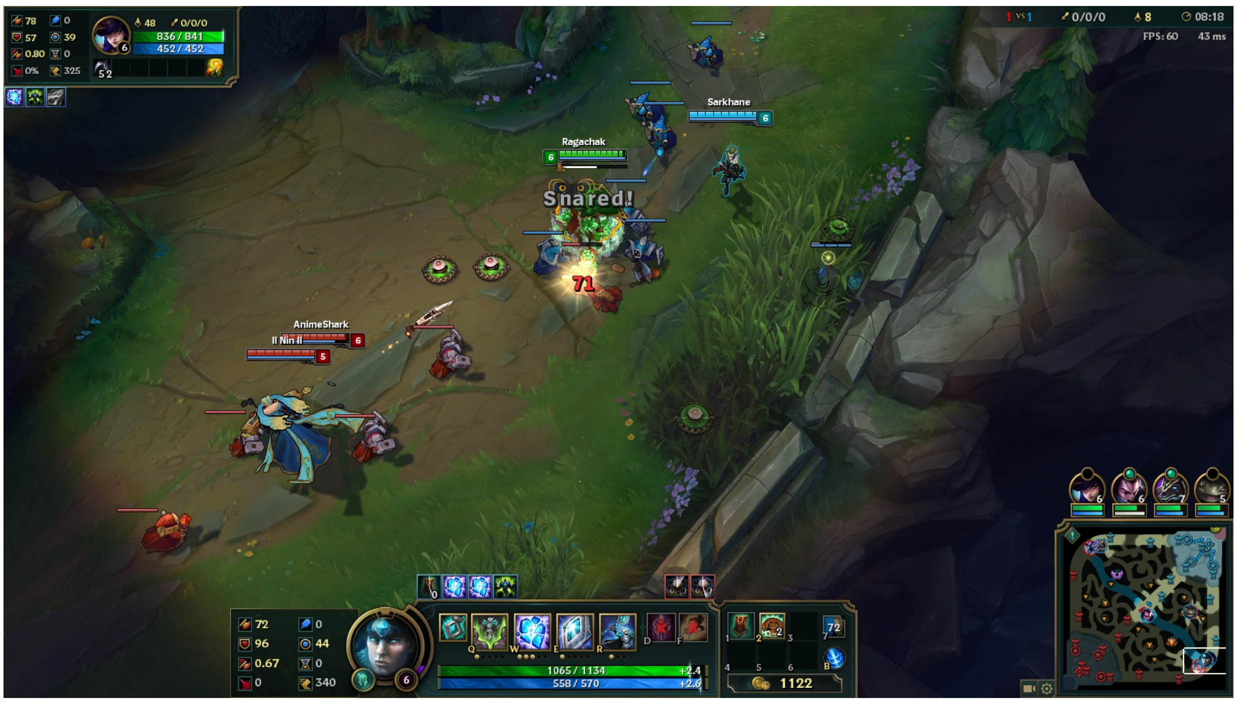


Figure 1.1.: Screenshot of a game of League of Legends™. The UI, heroes (champs) from opposing teams and computer-controlled minions can be clearly observed.

hero clothes/appearances. Such is the devotion of the player database that these micro transactions have more than paid off for the initially free game and the developers have successfully cashed in their product!

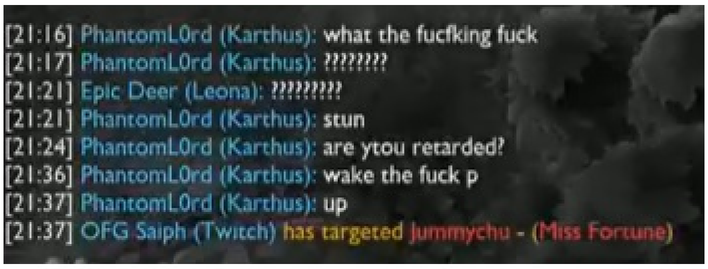
Both LoL and DotA 2 (another game of the same genre) top the eSports annually. **eSports** (also known as electronic sports) is a new form of sports where the primary aspects of the sport are facilitated by electronic systems (normally computers); and it has recently attracted large audiences that like to see professional video game players. Hamari and Sjöblom [15] delve deeper into the eSports appeal.

Looking at the data and the facts, we can conclude that collaborative play has come to stay. It is not just fun to tag along with a friend and play a video game together, it has even become a sport.

Returning to the difficulty factor in video games, how does it apply in co-op games? Certainly everything said and researched about difficulty in single player games can be extended to co-op games, if instead of the player we consider the whole team of players. But a new challenge appears: heterogeneously skilled players in the same team. Indeed, for a certain static level of difficulty a skilled player may find the game to be too easy, whereas a not-so-skilled player may find the game too difficult. There certainly would be a problem to choose a correct level of difficulty if both players belong to the same team. Due to this, the implementation of the traditional Dynamic Difficulty Adaptation for single players is also far from trivial. Dynamically adapting the difficulty would include observing all players in the team and adapting the difficulty for each of them accordingly, but also adapting the difficulty of the game for the team as a whole.

And what about the difficulty factor in the popular co-op games mentioned above? In general, LoL, DotA 2 and similar games have a steep learning curve, which means newcomers have a very hard time learning to play the game in the beginning. The frustration is hugely amplified in co-op games if newcomers play alongside experienced players, as their inability to play the game well enough does not only bother the newcomers, but also the experienced players who belong to the same team. In a toxic player community this leads to frequent rages, flaming and name-calling, making everyone's experience rather unpleasant. Tassi [28] mentions this problem and Allbrecht [3] delves into it. These games include a report system where players get banned if they misbehave, a system that has incited research too (See

Kou and Nardi [19]). Apart from better educating our teenagers (and not so teenagers) and the report system, what else could be done to mitigate this problem? What about addressing the steep learning curve and difficulty problems?



```
[21:16] PhantomL0rd (Karthus): what the fucfking fuck
[21:17] PhantomL0rd (Karthus): ????????
[21:21] Epic Deer (Leona): ?????????
[21:21] PhantomL0rd (Karthus): stun
[21:24] PhantomL0rd (Karthus): are ytou retarded?
[21:36] PhantomL0rd (Karthus): wake the fuck p
[21:37] PhantomL0rd (Karthus): up
[21:37] OFG Saiph (Twitch) has targeted Jummychu - (Miss Fortune)
```

Figure 1.2.: Screenshot of an unfortunate League of Legends chat. Sadly newcomers face this type of harassment frequently.

In this work we want to address Dynamic Difficulty Adaptation in Collaborative Games, which could potentially see an effective use in the popular co-op games of our day. Especially, we wish to successfully adapt the difficulty in co-op games where players in a same team have different skill levels. We will first delve into previous research on both Dynamic Difficulty Adaptation and Collaborative Games. Then we will proceed to establish, define and explain an abstract and generic approach to Dynamic Difficulty Adaptation in Collaborative Games. And afterwards we will of course want to see this tested in an actual game! For the purposes of this work we have implemented and programmed a new collaborative game called Co-op Craft that dynamically adapts its difficulty according to the players' performance. We will present this game and explain how it adapts its difficulty. We will also define the testing procedure and detail the obtained results. We will then analyze the results to extract significant conclusions and detect limitations and problems. Finally, we will evaluate our work and suggest possibilities to continue this work in the future.

2 State-of-the-art and Related Work

"Dynamic Difficulty Adaptation in Collaborative Video games" is quite unexplored as of yet and I have failed to find detailed scholar work on the subject. However, there is plenty of previous work in both "Collaborative Gaming" and "Dynamic Difficulty Adjusting in video games", separately. Therefore, here we will address relevant information and related work on these two subjects (also separately). We wish to present basic definitions, basic characteristics and challenges to these two subjects as portrayed by other authors, as well as interesting examples of implemented work. For the purpose of this work it would make sense to then apply established information and procedures from both subjects, as we treat "Dynamic Difficulty Adaptation" in combination with "Collaborative Gaming". We will do this in Chapter 3.

2.1 Collaborative Gaming

Whereas the intricacies of collaboration and team work have been researched and pondered upon for a long time, the concept of collaboration in video games is obviously not older than video games themselves. Most agree that video games became relevant about 50 years ago. Still, the vast majority of designed video games have been single player games or competitive multiplayer games, not collaborative multiplayer games. Though co-op games have been around for a while (two-player shooters in arcade machines are a prime example), it is only in recent years when scholars and video game developers have taken a keen interest in them. In recent years several co-op-only games have been developed (e.g. The Legend of Zelda: Tri Force Heroes™for Nintendo 3DS™) whereas other games and genres have included co-op mode as one of their attractive assets (e.g. Call of Duty™). The recently new game genre called MOBA (Multiplayer Online Battle Arena), which focuses on co-op gameplay, has become the star genre of the worldwide competitive eSports phenomenon. The MOBA game League of Legends™has become so popular that researchers have begun analyzing its features (e.g. Ferrari [10]) The advent of the Internet has definitely increased the interest in co-op gaming, as players can now find companions and partners to play with without the need of friends coming to their physical location. Sophisticated matchmaking systems are developed to pitch players of similar skill levels and make the game interesting for all players. We can expect co-op games and genres to continue to rise in popularity within the next few years.

2.1.1 Basic definitions

Before defining Collaborative Gaming, we should first look into "collaboration", which is more complex than one may think at first. According to Roschelle and Teasley [25]:

> "***Collaboration*** *can be defined as a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem*"

A similar concept is "cooperation", and many wonder if "collaboration" and "cooperation" are synonyms. Strictly speaking, they are not. Dillenbourg [9] asserts:

> "*In* ***Cooperation****, partners split the work, solve sub-tasks individually and then assemble the partial results into the final output*"

Therefore, collaboration is strictly speaking much more than cooperation, as it attempts to produce a better output than just the simple sum of individual and separate outputs of the different team mates.

In this work we shall adopt a lax position, however, as in the video game industry "Cooperative gaming" (or "Co-op gaming") is regarded as the feature that allows players to work together as teammates to achieve a common goal. [1] Hence, when a game includes a "co-op mode", it is strictly speaking including a "collaborative mode", not a "cooperative mode". "Collaboration" will however be addressed much more often in this work than "strict cooperation", so for simplicity purposes we will not be that strict and treat "collaboration" and "cooperation" as synonyms. If we do want to refer to the strict definition of cooperation given by Dillenbourg [9] above, we will then refer to it as "**strict(ly) cooperation**". Otherwise, "co-op games" will be a simpler and shorter way to mean "collaborative video games". Indeed, Dynamic Difficulty Adaptation in single-player games could be directly used in strictly cooperative games, as the individual sub-games can be treated as single-player games. This work would serve no purpose if it did not pursue Dynamic Difficulty Adaptation in collaborative games, which is a much more complicated task than applying Dynamic Difficulty Adaptation to several single-player games, as we will see in Chapter 3.

Returning to the beginning, we can now finally present the definition:

> "***Collaborative Games*** *are the set of video games that involve two or more human players playing and working together as a team to accomplish a common goal.*"

And as stated before, in this work "co-op games" will generally refer to this set of games too.

2.1.2 Characteristics and challenges

Collaborative games are obviously **multiplayer games**, so their design should take into consideration well-established elements from these games. Wendel et al. [30] outlines elements from traditional games:

- Fun
- Narration
- Immersion (how well does the player identify himself with his game character and actually feels the experience in the game world)
- Graphics
- Sound

and also elements/challenges exclusive to multiplayer games:

- Concurrent gaming
- Interaction between the players

Additionally, the **five essential elements for cooperative work** from Johnson and Johnson [18] are often valued when it comes to designing collaborative games:

- **Positive interdependence**: being aware that you are linked with other players in a way that you cannot succeed unless they do
- **Individual accountability**: individual assessment of each player's performance and reporting back the results to both the group and the individual
- **Face-to-face promotive interaction**: promoting each other's success by helping, encouraging, praising...
- **Social skills**: Interpersonal and small group skills are very useful for the success of a cooperative effort
- **Group processing**: Group members discuss their progress and working relationships

All of the above apply when conceiving and designing a collaborative game. Although relevant, it must be noted that these elements are not strictly essential *per se*. For instance, there are games that lack sound or outstanding graphics that are still interesting and/or popular.

[1] `https://en.wikipedia.org/wiki/Cooperative_gameplay`

2.1.3 Additional research

Many research articles focus on describing characteristics and implementations of co-op games. For example, Seif El-Nasr et al. [27] delves on understanding and evaluating cooperative games, and Rocha et al. [23] examines cooperative game mechanics. Their views can offer additional light to the outlined characteristics in the previous subsection.

Quite a lot of research has also gone into Collaborative Learning, Computer Supported Collaborative Learning (CSCL) and Game-based Learning (examples: Hämäläinen et al. [14]; Wendel et al. [29]; Admiraal et al. [2]) One of the interests in collaborative games is the possibility of making the players learn collaborative skills or simply learn things together as a group. Dynamic Difficulty Adaptation could potentially be very useful in Collaborative Game-Based Learning, adapting the difficulty of the game for a better learning experience. As explained by Wendel et al. [30], designing collaborative learning serious games is definitely possible, so Collaborative Learning may yet be another reason why co-op games see a rise in use in the future. In this work, however, Collaborative Learning is not the focus, so it will not receive further mention than this paragraph.

2.2 Dynamic Difficulty Adjusting in Video games

Traditionally the approach to set and adjust the difficulty of a video game has been **static**: before beginning the game the player is often asked to choose a difficulty setting (e.g. - easy, medium or hard) and levels and challenges become increasingly more difficult (normally linearly) as the player advances in the game. This approach does not take into account the player's input in the game to vary the game's difficulty and this is why it is called a **static** approach. An advantage to this approach is its simplicity, but a considerable disadvantage is frequently failing to achieve the entertainment sought in the first place (more advantages and disadvantages will be discussed in Section 2.2.2). As correctly pointed out by Hunicke [16], in contrast to films, books or televised media, video games are **interactive**, i.e. the player and his/her input is an essential factor in a video game. Gilleade and Dix [11] explain that the player as a human is an unpredictable dynamic entity who will also absorb information and adapt to the game experience as he plays, thus his skill improvement will normally not be linear. A lot of research has gone into "Dynamic Difficulty Adjustment", which does take into consideration the player's input to balance the game's difficulty - hence the term **dynamic**. Dynamic Difficulty Adjustment can be applied to any game genre.

2.2.1 Basic definitions

There are several definitions for Dynamic Difficulty Adaptation, although they are all intrinsically the same. First we should note that it has several synonyms: **Dynamic Difficulty Adjustment**, **Dynamic Game Difficulty Balancing** - or simply "**Dynamic Game Balancing**" **(DGB)**, and the commonly used acronym **DDA** all mean the same thing as **Dynamic Difficulty Adaptation**. Now some definitions. According to Kuang [20]:

> "*The main idea of DDA is to have facilities in the game available to gauge the player's performance and skill, and adjust the difficulty accordingly during gameplay to provide the most consistent (and hopefully most fun) experience for the player.*"

The definition by Glassner [12]:

> "*Games that apply DDA adapt themselves during gameplay to offer the player a consistent degree of challenge based on his changing abilities at different tasks.*"

I personally find the definition in the Wikipedia article very accurate:[2]

> "*DDA is the process of automatically changing parameters, scenarios, and behaviors in a video game in real-time, based on the player's ability, in order to avoid them becoming bored (if the game is too easy) or frustrated (if it is too hard).*"

The algorithm that operates DDA in a particular game is called the **Dynamic Difficulty Adjustment Algorithm**, and we will call it DDAA in this work. There can be several DDAAs in a game, or a big DDAA that can be subdivided in individual DDAAs, as we will see in Section 3.3.

2.2.2 Challenges, advantages and disadvantages of DDA

While DDA can definitely have benefits, it does not come cheaply. Hunicke [16] warns that DDA systems take control away from the designers and put it in the hands of code, which is not always reliable. DDA has been implemented commercially in few cases - a discussion of commercial DDA system designs and failures is presented in the work done by Arey and Wells [4]. It is also not easy to keep the player interested and challenged in interactive contexts. Hunicke et al. [17] gives proposals and tips to design DDA and address this problem, some of which we will see in the next subsection.

In the excellent article by Adams [1], dangers and disadvantages of implementing DDA instead of the traditional static difficulty levels are exposed. He mentions that:

- **Some players hate DDA.** For them, playing, and beating, a very difficult game is where the fun is, even if it means dying 500 times on the way to eventual success.
- **DDA can be exploitable.** Players may pretend to be worse than what they actually are so that the game becomes easier.
- **As opposed to what theory says, DDA may not work for all kinds of challenges in practice.** It is definitely not easy to modify the difficulty of a set puzzle which is already in-game. DDA could still however drop hints to the player to solve the puzzle, or choose a less complex puzzle for the next level.
- **DDA can create absurdities.** For example, in some car racing games, if you crash your car, the game slows down the other drivers so as to give you a chance to catch up. Then there is the notorious **rubber-band effect**: if you get too far ahead your opponents always catch up, and if you get too far behind, they always slow down.
- **DDA has the potential to ruin pacing and obviate good level design.** A hypothetically perfect DDA system that always kept all challenges at the same level of perceived difficulty would ruin the pacing of the game, which is intended to increase in difficulty with time.

Adams [1] is not totally opposed to DDA, however. He gives suggestions to the developer in case he wishes to implement it, notably that DDA should be as subtle and secret as possible so that players can not exploit it. He concludes that:

> "*DDA is difficult to implement, complicates tuning, and your player may not want it at all. Think long and hard before you commit yourself to it.*"

We will keep these warnings in mind throughout this work, but we are not discouraged of the possible disadvantages of DDA, since we believe it has significant advantages to offer. These advantages have probably been mentioned already or will be mentioned further ahead, but to contrast them with their disadvantages we will list them here. Here are some advantages of using DDA (the last three are given by Glassner [12]) over static levels of difficulty:

[2] `https://en.wikipedia.org/wiki/Dynamic_game_difficulty_balancing`

- **DDA eases the learning process in games with a steep learning curve or that are too difficult for beginners**. One example that has been discussed is League of Legends ™.
- **DDA can ease the process of balancing a game's difficulty**. Balancing is traditionally achieved with iterative testing feedback, which allows the developers to tweak behaviors and settings until the game is reasonably balanced. This is a difficult and time consuming process, as per Rollings and Adams [24]. However, while game balancing and tuning can not be totally automated, an effective DDA system can reveal deeper structures and relationships within a game system in less time and with greater accuracy, says Hunicke [16], thus greatly easing the balancing task.
- **In co-op games players in the same team can work effectively together regardless of their skill level.** This is actually one of the main motivations for this work!
- **Static difficulty levels are too broad.** What is "easy" for some may be "hard" for others. What level should a player choose in the beginning?
- **Static difficulty levels could be too coarse.** What if medium mode is too easy, but hard mode is too hard? Or what if the hardest difficulty mode is too easy? (something that usually happens to me when I become experienced in a certain game)
- **Static difficulty levels are too persistent.** A difficulty setting does not adjust to the player's rate of improving skill, especially if he is not allowed to change the setting later. The difficulty growth curve, at whatever setting, may prove to be too steep or too shallow for the player.

2.2.3 Characteristics and Design Patterns of DDA

Now we will turn our attention to DDA characteristics and design patterns and approaches suggested and/or implemented by researchers. They will be very useful when we consider DDA in co-op games in Chapter 3.

What is to be considered when designing a DDA system? According to Gilleade and Dix [11] there are three main factors to be considered:

1. **Physiological Motivators** - Why do players play? Is it for fun? Is it for the challenge? (e.g. if the players play for the challenge maybe they will not appreciate the game becoming easier - one of the disadvantages mentioned in the last subsection.)
2. **Experience/Skill** - How strong are the players? This factor definitely influences the level and nature of the changes to be performed.
3. **Detection** - When should a game adapt? Between levels? At certain periods of time? When a player expresses certain feelings? (the so-called "affective computing adaptation")

For Hunicke [16] DDA is a design problem that involves estimating WHEN and HOW to intervene, two key questions.

- **WHEN**: While we tend to think that DDA performs in real-time, this is not always necessarily the case. Bakkes et al. [5] suggest a case-based adaptive game AI, where the DDAA gathers information of previous games and starts taking decisions even before the game starts. In this work we will only consider DDAAs that only perform in real-time in-game though. Key question: "what should trigger the change in the game's difficulty?" "Affective computing adaptation" is interesting and has been researched quite a lot (e.g: Chanel et al. [6]). It consists of observing a player's feelings and triggering DDA according to them instead of the player's performance in-game. Gilleade and Dix [11] explore frustration as the trigger, and concludes that adaptive games based on a player's affective state must be taken with extreme care. In this work we will not use feelings as a trigger. We will use the traditional DDA approach of observing the player's performance in-game (See Section 3.3), because it is much less complex than "affective computing adaption".

- **HOW**: Performing a change/adjustment in the game difficulty is tricky. As stated by Gilleade and Dix [11], catering in excess for heterogeneity can lead to a bloated game, whereas tweaking the wrong features will disrupt the player's sense of disbelief (again, see disadvantages of DDA). It is important to support/challenge the player "just enough", to perform the changes smoothly and to avoid predictability. Hunicke [16] presents **adjustment policies**: the combination of **adjustment actions** and **cost estimations**. Actions can either be reactive or proactive. Reactive actions adjust elements already "in play", e.g. enemies already visible; proactive actions adjust elements "off stage", e.g. enemies spawning in the future. The designer must analyze the cost of these actions (benefits and drawbacks), and determine which one to apply. An example of a policy could be "Staying in the Comfort Zone", where ideally a player always has between 25% and 75% of his total health left. If a player drops below 25% enemies could be made to have lower precision whereas if a player hovers above 75% enemies could spawn in larger numbers. We will use some of these concepts in our own classification of DDAAs (Section 3.3).

M. Csikszentmihalyil [8] developed the so-called "flow model", which he explains in his book. This can be applied to DDA, where the aim is to keep the player in the flow channel (see Figure 2.1).

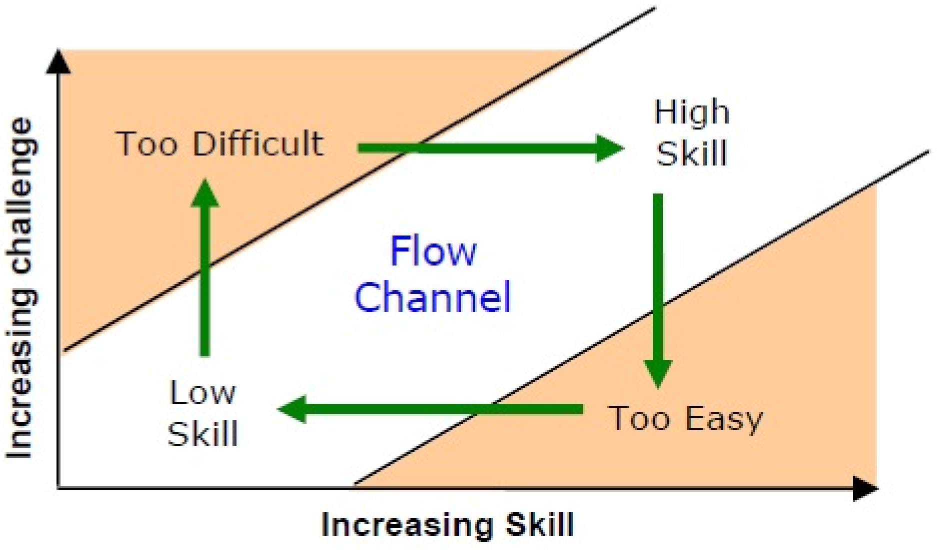


Figure 2.1.: Flow state transition over the course of a dynamic experience.

Despite needing a careful implementation and not being trivial, DDA does not need to be overly complicated. The Mario Kart™series is a prime example of simple yet effective DDA - in this racing game players get items that either tamper with opponents' driving or boost their own driving. Players at the end of the queue normally receive more powerful items than those leading the race. This produces the already mentioned "rubber-band" effect which allows for those that are behind to catch up with the rest of the drivers, while the leaders can not escape from the pack and create an unsurmountable advantage. Adams [1] is skeptical of this, as we have already seen, but in cases like Mario Kart™some claim it is part of the fun and charm of the game. In this work we strive for a similarly simple yet effective implementation of DDA, and because we will not be implementing a racing game, the "rubber-band" effect will not come into play and we will avoid its controversy.

To finish we will mention the work by Combs [7], where he discusses the AI for Massive Multiplayer Online Games (MMOG). He indirectly hints at DDA in MMOG, which could be very useful in this game genre. MMOGs can greatly benefit from "on-the-fly" adjustments for balancing their difficulty, as the always changing community of active players has a huge influence on the overall difficulty of these games. On the other hand the big complexity of MMOGs would make implementing DDA in them a challenging task.

3 Concept of Dynamic Difficulty Adaptation in Collaborative Games

As was already mentioned, very little or no research has been done specifically on DDA in co-op games, and that is why we gathered information about DDA separately from co-op games. With this background we will now present a general and abstract approach for DDA in co-op games, utilizing concepts seen in the state-of-the-art chapter as well as defining new concepts where needed.

Firstly we wish to establish our own definition of **Dynamic Difficult Adaptation** (or simply "**DDA**"), based on the ones from the previous chapter:

> "*DDA is the process of analyzing a player's ability and performance in a video game, and then accordingly adjust and change significant parameters, scenarios, and AI behaviors in order to adapt the game's difficulty and thus avoid the player from becoming bored (if the game is too easy) or frustrated (if it is too hard).*"

As mentioned before, a **DDAA** (Dynamic Difficulty Adaptation Algorithm) is an algorithm that is responsible for executing DDA in a video game. These algorithms can vary in size, complexity and number in a game.

3.1 Challenges of DDA in collaborative games

Apart from the challenges to DDA that have been already exposed in the previous chapter, what additional challenges does DDA carry when we intend to apply it in collaborative games?

First we have the **scalability challenge**, which applies not only to co-op games but also to multiplayer games in general. If a DDA consumes a lot of the machine's resources when only adapting the difficulty for a single player, this same machine may not be able to tolerate calculations for several players. Simple DDAAs are normally safe to be scaled from single to multiplayer, but resource-demanding DDAAs require attention, as scaling them can lead to game crashes or online lag, none of which are ever welcome. In the implemented game in this work the DDAAs did not have a major impact on computer performance, but this challenge remains as a warning.

The **heterogeneously skilled player groups challenge** is a major one for DDA in co-op games. It refers to the dilemma of modifying the general difficulty of a game where players of different skills play together in a certain team. A certain team may include a talented player and a not-so-talented player, a seasoned veteran and a fresh newcomer, an attentive player and a distracted one... If the game becomes generally easier, the skilled players will become even more bored; and if the game becomes generally more challenging, the less-skilled players will find it even more challenging and frustrating (probably impossible for them). Subdividing the game and adjusting certain parts to avoid this problem is not trivial and requires consideration and good design. In the next section we present a general approach to deal with this challenge, which is one of the main contributions of this work.

Finally we can also mention the **interference challenge**. It can take many forms, but it generally refers to the malfunctioning of DDA due to the unexpected influence of other players. Of course several DDAAs assigned to different players can interfere with each other in the same way concurrent threads can also interfere with each other (classical challenges of concurrent computing), but interference can also happen through the behavior of other players. For example, imagine an FPS co-op game where the difficulty increases as you deal damage (the more damage you deal means you aim better and you are considered a better player). A player throws high-damage grenades to deal with enemies, but the explosions also hurt nearby players from his team. The DDA detects that the player is dealing a lot of damage and substantially increases the difficulty of the game, when in reality the game should be made easier because the team is overall in worse shape! There are of course ways to deal with this problem:

turn friendly fire off, make the DDAA not count friendly fire as damage dealt or ease the game difficulty when players take damage. The underlying premise is: when implementing DDA in co-op games, be aware of the possible influence of ALL players.

3.2 Distinct-role collaborative games

Among all co-op games, we find one subgroup to be especially viable for DDA: **distinct-role co-op games**. In these games players in a same team have clearly distinct and complementary roles. Players depend on each other to achieve their common goal, and one player alone should not be able to "solo" the game by himself, i.e. complete the game with little to no help from the teammates. For example, in an FPS game one teammate may be able to heal his comrades while another may be able to snipe far away enemies. To emphasize the need for each role, maybe in one part of the stage a toxic gas forces players to depend on the healer to survive, whereas in another stage remote mines need to be sniped from far away to allow teammates to progress in a stage.

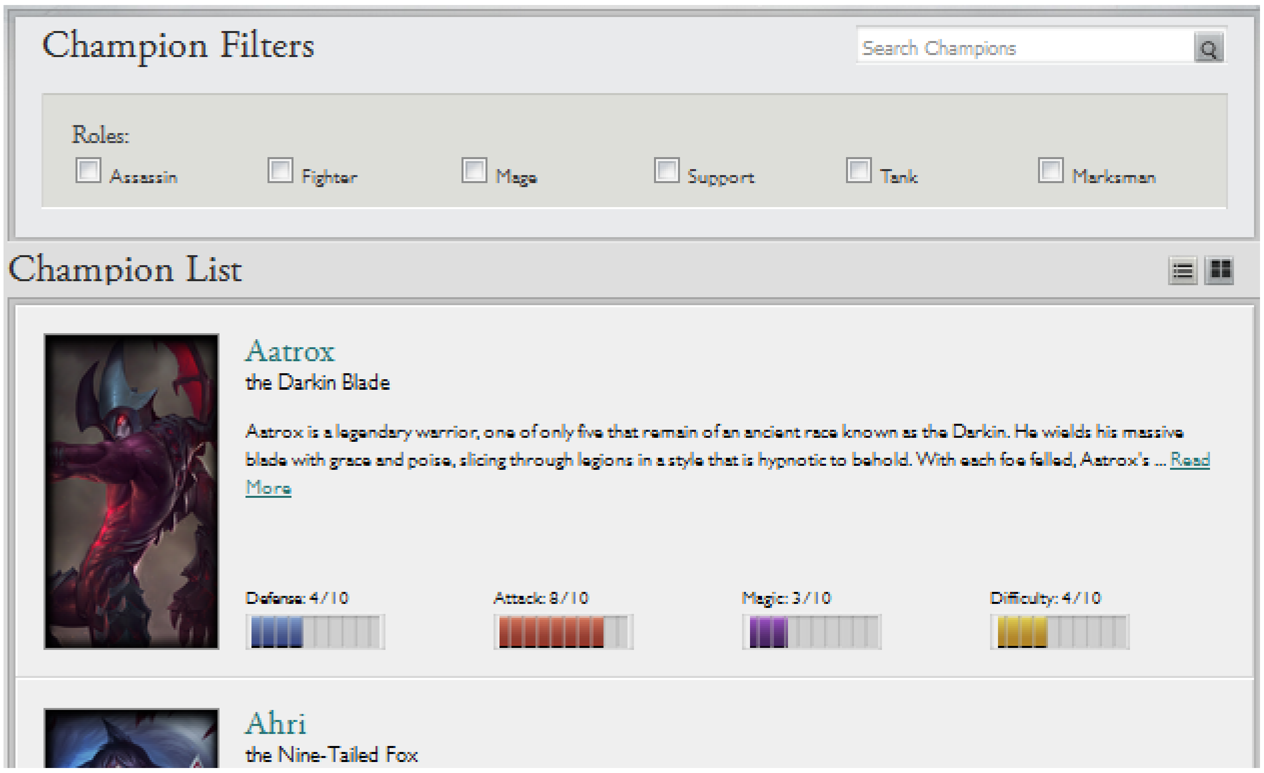
On the other hand these distinct roles should not be so exclusive and opposing that they make the collaborative game turn into an strictly cooperative game (recall the difference between collaboration and strictly cooperation as explained in Section **??**). Players are still supposed to be able to combine their input and work effectively side by side, hence the term "complementary" roles. Roles do not need to be permanent and can indeed be switched or changed for other roles in different phases of the game, as long as the roles remain complementary. In the previous example, the "healer" could become the "sniper" and the "sniper" could become the "ammo collector" in another level of the game (a level where hopefully there is no toxic gas and thus no need for a "healer").

Why would DDA be easier to implement in distinct-role co-op games? The reason is that lots of aspects from traditional single-player DDA can be then applicable, as they would only affect the one player with a determined role. This also means that the heterogeneously skilled teams challenge can be tackled more easily.

Let's look at an example of a co-op game that does NOT belong to this group: the classic arcade game House of the Dead™by Sega. This is a very straight forward "shoot the zombies" game, and it has a multiplayer co-op mode. Both players could be considered clones, because their roles are the same (shoot at the enemies that appear on the screen). Their health, ammunition and even their weapon is also the same (see Figure 3.1). Indeed, if one of the players excels at this game, he can single-handedly destroy all of the enemies without the help from his teammate. Changing the difficulty by spawning more or less monsters would not make a significant impact because the good player would still be the one who kills most or all of the zombies. It is a clear of example of the difficulty presented by the heterogeneously skilled teams challenge. In contrast, thinking back on our original example with a "healer" and a "sniper", modifying the amount of targets that have to be sniped would only affect the difficulty for the player who has the "sniper" role, which means traditional single-player DDA can be effectively applied here. DDA can still be applied to co-op games that are not role-distinct (in the House of the Dead™example, maybe the good player could be made to take a bit longer to reload and thus give the other player a chance to shine more). However, and as we have already discussed, it is definitely much more difficult to apply traditional single-player DDA in co-op games that are not role-distinct. For this reason, in this work we will focus solely on DDA in distinct-role co-op games.

The MOBAs presented in the introduction are examples of distinct-role co-op games, because champions have a role assigned to them. In LoL the roles are Assassin, Fighter, Mage, Support, Tank and Marksman. (See figure 3.2). Hero stats (attack, defense, health...) and abilities highly depend on the role. Normally one can see a variety of roles in the same team, but sometimes players do choose champions with overlapping roles. Thus it could be argued that LoL is not a pure distinct-role co-op game, or at least, not always.

Figure 3.1.: Screenshot of the arcade game House of the Dead™by Sega, (co-op mode)


Champion Filters

Search Champions

Roles:

☐ Assassin ☐ Fighter ☐ Mage ☐ Support ☐ Tank ☐ Marksman

Champion List

Aatrox

the Darkin Blade

Aatrox is a legendary warrior, one of only five that remain of an ancient race known as the Darkin. He wields his massive blade with grace and poise, slicing through legions in a style that is hypnotic to behold. With each foe felled, Aatrox's ... Read More

Defense: 4/10 Attack: 8/10 Magic: 3/10 Difficulty: 4/10

Ahri

the Nine-Tailed Fox

Figure 3.2.: League of Legends champions listing in the official webpage. Champions have distinct roles, so they logically can be filtered according to their role

3.3 Classification of Dynamic Difficulty Adaptation Algorithms

DDA can take a myriad of forms, even in distinct-role co-op games. Due to the many possibilities developers have when implementing DDA in their own particular games, we aim to now classify the possible DDA algorithms according to their characteristics. We will also try to present possible advantages and disadvantages of choosing a certain characteristic for a DDAA, as well as to illustrate these characteristics with examples.

Hunicke [16] already pointed out that two key elements in DDA is figuring out WHEN the system should intervene and also HOW the system should intervene. We are going to take this a step further and present different DDA classifications according to WHEN does the game detect that change is needed, WHAT does the game detect and HOW should the game adapt or react. These three elements are very important to be taken into account when designing a DDAA.

3.3.1 DDA according to WHEN to detect

1. **Timed-based triggering** - The DDAA activates at specified moments/times during the game. Within this classification we have to distinguish between periodic timed-based triggering and non-periodic timed-based triggering. In **periodic timed-based triggering** the DDAA activates every certain period of time. An example would be checking every 30 seconds the ammo of a player in an FPS game. In **non-periodic timed-based triggering** the DDA activates at points in time that are heterogeneously separated in time. In many kinds of games it is important to frequently adapt the game difficulty in the beginning, but as the game advances the DDAA will have gathered enough data to determine a level of difficulty for the rest of the game, and the DDAA does not need to activate that frequently. An example of this situation would be checking the average speed of a player in a racing game: in the beginning averages tend to fluctuate much more than later in the game, when they tend to be stable. There is also a **hybrid** option, where the DDAA activates every certain period of time, this period however varies in length in different phases of the game. An advantage of the pure periodic timed-based triggering is obviously its simplicity: with only a starting moment and a constant period one can determine every time the DDAA will activate.
2. **Event-based triggering** - The DDAA activates when certain events occur in the game. It could be argued that timed-based triggering is a subgroup of event-based triggering, the triggering event being "X amount of time has elapsed". We will exclude elapsed-time events here however, and refer exclusively to time-independent events (otherwise "Timed-based triggering" would only be a particularization of "Event-based triggering"). Within this classification we will distinguish **independent events** from **linked or iterative events**. An example of the former would be triggering the DDAA every time a player dies (generally past deaths do not influence future deaths). An example of the latter would be triggering the DDAA every time a player picks up a key object, when all key objects are linked with each other - for instance a DDAA triggers if the player picks up a lock, but only if the player has picked up a key before; otherwise the DDAA for the lock will fire once the player picks up the key.

3.3.2 DDA according to WHAT to detect

1. **Parameter probing** - The DDAA observes one or more significant parameters in order to decide how to adapt the game. A parameter is significant if it effectively measures one or more features of a player's performance, so it follows that the choice of significant parameters to be observed is key to the DDAA's effectiveness. Examples of possibly significant parameters: total amount of collected resources, damage dealt, number of enemy units killed... According to what the parameter's value is being compared to, we can distinguish comparing-to-standard parameter probing and player-progression-analysis parameter probing. **Comparing to standard** involves, as its name suggests,

comparing the parameter's value to a standard value. If the parameter value falls behind the standard, the DDAA should attempt to make the game easier, whereas if the parameter is ahead of the standard, the game's difficulty should be increased. Deciding what the standard value of a parameter should be is also a design decision that must be justified. **Analyzing the player's progression**, on the other hand, compares the parameter's value to previous values of the same parameter in the game and determines how to react based on the progression. For example, the DDAA probes the player's score. If the score starts to increase at a slower rate than before, maybe the DDAA will take action so that the player picks up the pace again. The score in itself is however not compared to a given standard.

2. **No probing** - Sometimes probing parameters is not necessary. For example, if a game objective is taking too long to complete the developer may decide to have a DDAA ease the difficulty. One could argue that in this case the parameter "elapsed time" is being observed, but on the other hand it could also be argued that "elapsed time" is not a significant parameter to player's performance (taking long does not always translate to performing poorly). Personally I do think "time elapsed" is significant in many types of games, but in this work we will still consider a DDAA to be non-probing if it does not probe a significant parameter other than the elapsed time. "Time" is normally a special parameter, common to all players in the game, which can normally be measured and accessed in its own particular way in many game development tools, so it is worthy to be treated specially.

3.3.3 DDA according to HOW to react

In this section there are many characteristics to be considered, and we will classify the DDAs by pairing excluding or opposing actions. Pairs of opposing actions are independent from each other, so a DDAA can have, for example, both a reactive and explicit action (but not a reactive and proactive action - if a DDAA is reactive and proactive it is because it has reactive actions and proactive actions).

1. **Reactive/Proactive** - Here we will take the definition presented by Hunicke [16]. The DDAA is reactive if it adjusts elements that are already "in play" or "on stage" (e.g. enemies the player is fighting right now), thus these existing elements "react". The DDAA is proactive if it adjusts elements that are "off stage" (e.g. enemies that will spawn in the future), thus the future elements "proact" (act before they exist in the game). As explained by Hunicke [16], there is a trade off: reactive DDA has an immediate effect, but can disrupt the player's sense of disbelief; whereas proactive DDA is more powerful but is affected by the distance factor - if the effects are too far away in the future, it is uncertain whether the DDAA will achieve the desired effects, it may even produce undesired effects!
2. **Discrete/Continuous** - Many DDAAs will involve calculations to produce their output. According to the output we can classify DDAAs as discrete - the output can only take a fixed set of values (e.g. "On/Off", "{-1,0,1}"...) or continuous - the output can take any value in an infinite set of values (e.g. "any real number between 0 and 1", "any positive number"...). Please note that discrete/continuous are not necessarily related to integer/real values as in Math.
3. **Implicit/Explicit** - A DDAA is explicit when its changes are intentionally visible and perceptible for the player. For example, the DDAA may choose to highlight the weak spot of enemies so that the player can destroy them more easily. On the other hand the DDAA is implicit when the changes are intended to go unnoticed. For example, the DDAA may adjust an enemy's precision so that it hits the player 10% less of the time. Intuitive players may still perceive differences and changes, especially if they compare the game when they play to the game when others play, but with implicit DDAAs the developer aims to make difficulty adjustments as less obvious and as smooth as possible, and tries to avoid disrupting the player's sense of disbelief.
4. **Suggestive/Auto-corrective** - A DDAA is suggestive when it drops hints and tips to the player in order to improve his/her performance, but does not modify the player's input, i.e. all actions

performed are still traced solely to the player. An auto-corrective DDAA, in contrast, can modify significant parameters and/or the player's actions with the intent to improve performance. For example, in many RTS games a key factor is to gather resources with worker units, but sometimes workers are left idle. A suggestive DDAA might highlight idle workers so that the player is aware of them and orders them to get to work, whereas an auto-corrective DDAA might automatically order workers to harvest resources if they have stood idle for too long. Developers must be careful with auto-corrective DDAAs, because players may perform "less than ideal" actions on purpose and get displeased when they find out that the game is not obeying their actions. In the mentioned example, maybe the player wants to leave a worker standing idle outside the base to spot incoming enemies, but if the DDAA sends him back to work, the player's intention is brought to naught.

5. **Standardizer/Progressive/Predictive** - In the previous subsection we discussed how a parameter probing DDAA could either observe the progression of values of said parameter or compare the parameter to a given standard. Apart from observing, a DDAA can also react according to a given standard or according to the progression of one or more parameters. A standardizer DDAA attempts to bring the player's performance closer to the standard (if the player's performance is below standard the game will tune down the difficulty, and if above standard, the DDAA will increase the game's difficulty.) A standardizer DDAA's effectiveness can be measured relatively easily - how closer does the player's performance get to the given standard with the DDAA active? A progressive DDAA attempts to maintain the player's performance progression according to his previous performance in the game. It does not attempt to bring the performance closer to a given standard, but rather deduces a standard according to the player's performance so far. We must also consider predictive DDAAs, which analyze the player's performance so far and predict how (s)he will fare in their future tasks to know how to react. For example, if a player performed not so well in mini-bosses (lost too much health, took too many hits, missed several attacks...) the DDAA might simulate the outcome against the boss and predict that the performance so far will not be enough to defeat the final boss, and thus decide to tune down the boss' difficulty.
6. **Single-player/Multi-player** - This characteristic comes naturally when applying DDA to collaborative games. The DDAA's actions may directly affect either one of the players (Single-player DDAA) or several players (Multi-player DDAA). If there were different teams of players in the game we could even add the term Team DDAA to refer to a DDAA that affects all players in the same team.

3.3.4 DDA classification summary table

WHEN	WHAT	HOW
- Timed-based triggering - Event-based triggering	- Parameter probing - No probing	- Reactive/Proactive - Discrete/Continuous - Implicit/Explicit - Suggestive/Auto-corrective - Standardizer/Progressive/Predictive - Single-player/Multi-player

4 Co-op Craft - the implemented game

Now that we have defined and explained DDA in co-op games and our general approach, it is time to actually see an example of implemented DDA in a co-op game and test its effectiveness. The following paragraphs and sections are here to provide the background that lead to the implemented game. The background is not necessary to understand the game concept, so the reader may choose to jump directly to Section 4.3.

To test our general conception of DDA in co-op games, we could ideally take an existing open source co-op game and implement some DDAAs in it. However there is very few multiplayer open source games to start off with [1]. Out of these, even fewer are co-op games, and none are clearly distinct-role co-op games. The mentioned League of Legends™, though a distinct-role co-op game, is not open source and is also quite complex, so even if we had access to the code, developing significant DDA for this game would be tedious work more fit for a video game development team.

In this situation I was left to program a distinct-role co-op game all by myself. To keep things as simple as possible, the game would be a two player human team against the computer AI. But even if we attempt to make it as simple as possible while still testing out DDA in co-op games, programming a game from scratch is still a strenuous task that must not be underestimated, especially with the time constraints this master thesis had.

To develop the game one could opt for a general video game development tool like the Unity cross-platform game engine, which students have had the chance to use in university lectures and labs. From my own experience I did know that programming a game all by myself in a few months would mean an extremely basic gameplay, with very lackluster graphics and possibly no sound. The idea for such a game (named "ADappTowr") did hit the paper, and will be explained here as an alternative to the actual implemented game (See Chapter 5).

Instead of the broad Unity game engine, I decided to develop my co-op game based on the RTS-specific StarCraft II engine. StarCraft II™, a 2010 RTS developed by Blizzard Entertainment, is not totally open source, but provides some powerful modding tools for the fan community to develop and create their own games based on the StarCraft II engine. A great advantage in using this engine is that graphics, sound and a lot of the gameplay is already set and has a professional feel. The end result would not be spheres jumping on rectangular platforms, which was the best I could manage to program in Unity in a few months. The biggest disadvantage to using the StarCraft II engine is that it is of course not as flexible as a broad engine. As we will see, though, an RTS-game engine still offers a great variety of possible games to develop.

4.1 A brief background to StarCraft

In 1998 Blizzard Entertainment released a military science-fiction real-time strategy (RTS) game known as StarCraft™. It is a highly acclaimed and popular game that has sold millions of copies worldwide, and critics consider it one of the most influential games of all time [2]. Its praise comes mainly for being the first RTS game that introduced unique and distinct factions (Terran, Zerg and Protoss) that play very differently from each other, but still keep a balanced gameplay, i.e. no race/faction is overpowered against the other. Its multiplayer is excellent and a big number of gamers have played this game in LAN parties and in the online Blizzard server Battle.net for many years. Over the years much research on the RTS genre has included the StarCraft game - see for example Ontanón et al. [22], Sánchez-Ruiz [26] and Lewis et al. [21].

[1] `https://en.wikipedia.org/wiki/List_of_open-source_video_games`

[2] `http://web.archive.org/web/20080913085022/http://www.gamepro.com/article/features/110068/the-52-most-important-video-games-of-all-time-page-4-of-8/`

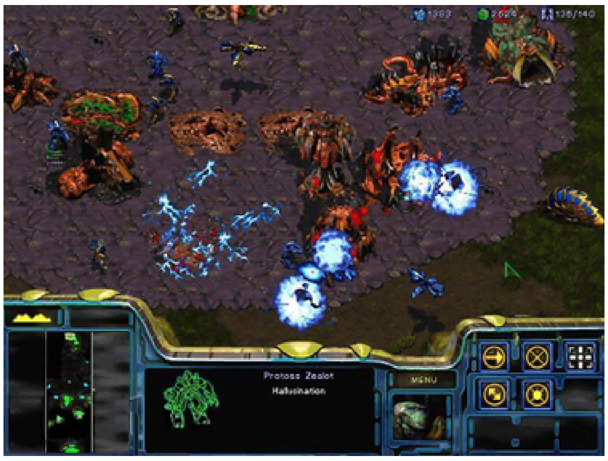


Figure 4.1.: Original StarCraft: a Protoss force attacks a Zerg colony

StarCraft spawned an expansion later in the year 1998 called StarCraft: Brood War™, but 12 more years would need to pass until the sequel StarCraft II: Wings of Liberty™was released. StarCraft II included improved graphics and a much more user-friendly User Interface, while preserving the lauded gameplay and balance from the original. StarCraft II has made it to the eSports too, although it is less popular than the aforementioned LoL and DotA 2. Two expansion packs have been launched: Heart of the Swarm™in 2013 and Legacy of the Void™in 2015, but Blizzard keeps patching and updating the game up to this day as well as releasing more content through micro-transactions. StarCraft II clearly shines as one of the best RTS ever and clearly as the best RTS game of today, in part because the pure RTS-genre has hugely decreased in popularity in recent years and thus StarCraft II stands as the last survivor game of its genre.

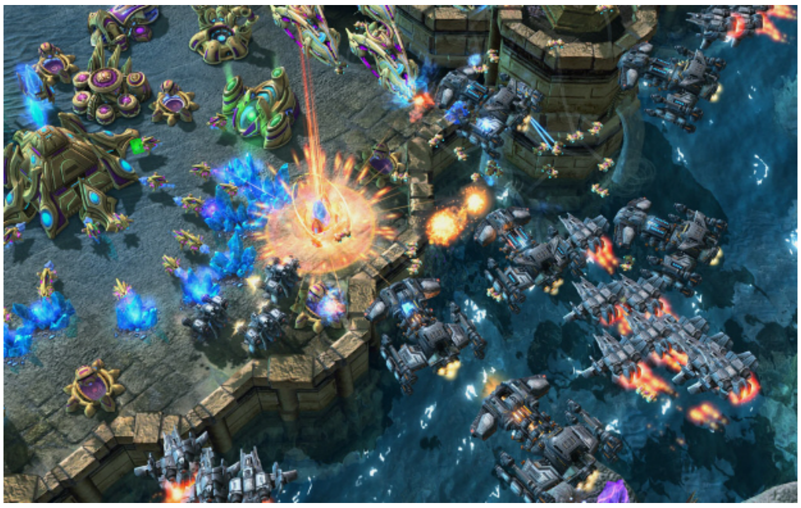


Figure 4.2.: StarCraft II: a Terran air fleet attacks a Protoss base. The improvement in graphics is notable.

I have personally played countless hours both the original StarCraft and the sequel StarCraft II, it is one of my favorite games. Its learning curve is steep though, even higher than the MOBA games already mentioned. Even if my skills are far from professional in the game, my in-depth knowledge of the franchise and the games is a big advantage when designing a game with this engine - I know the game units, spells/abilities, the core mechanics...

4.2 The StarCraft II Galaxy Editor

Apart from its addictive online multiplayer and amazing campaigns for single player, another factor for StarCraft's success were the custom games (or "custom maps") that anyone could create with Blizzard's development tool, the StarCraft Map Editor. Blizzard has the tradition of releasing development tools that greatly extend the lifetime of its games, as new games or maps were constantly published during the years following StarCraft's release. The original editor allowed to modify the terrain in the map, basic unit stats and provided a bunch of hard-coded functions called "triggers". Triggers would fire on events, check conditions and then perform actions. Even though the original editor had some serious limits as to what developers could create, still many different maps and ideas where implemented and a fan community of developers evolved. The community would eventually publish enhanced Editors that transcended the limits imposed by the original, which led to StarCraft's modding to take off. I personally enjoyed creating maps back then in the original Editor and those were my first steps into programming, long before I even learned a programming language!

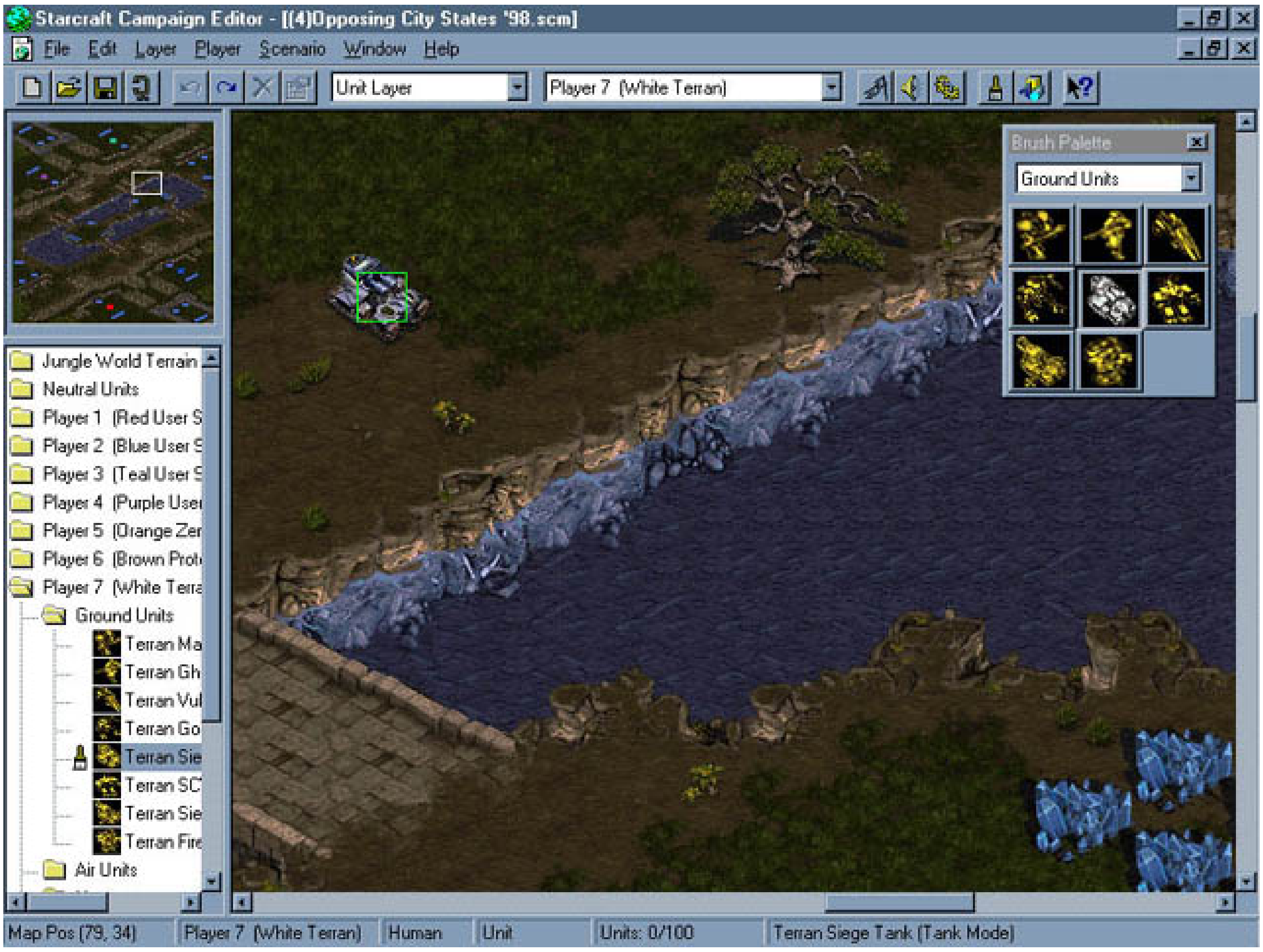


Figure 4.3.: Screenshot of StarEdit - the original StarCraft Map Editor

To emphasize the potential the StarCraft Editor had, a map called "Aeon of Strife" was created with the Editor, a precursor to the current popular MOBA games. In there, stronger StarCraft hero units battled in an arena where other weaker computer-controlled StarCraft units fought in a tug-of-war. Later the same idea was implemented and refined with the WarCraft III™Editor, and the resulting map was called "Defense of the Ancients" (original DotA). The map became extremely popular, to the point that tournaments began to be held - tournaments to play a fan-developed custom map! [3] The developers were afterwards hired by Valve and they developed the aforementioned DotA 2, this time outside the WarCraft III engine. Blizzard tried to gain the rights for this custom map/game which had spawned from their own Editors, but lost the lawsuit (See Haas [13]). This is the summarized history on how the MOBA genre spawned from the RTS engine. All from a user-friendly and deceptively simple RTS Map Editor!

With the release of StarCraft II in 2010 Blizzard made available the new StarCraft Galaxy Map Editor. Quoting the Blizzard team [4]:

> *"All our StarCraft II development tools are in your hands. When you install the free StarCraft II Starter Edition, you also get the StarCraft II Editor. Fire it up, make your own maps and mods, and share them with the world via Battle.net."*

Including modules enabling the editing of Data fields, Triggers and Terrain, the StarCraft II Map Editor improves upon its predecessor, the World Editor from Warcraft III, in every way. The Editor even features a proprietary scripting language called Galaxy based on C. Every mission within the single player Campaign for StarCraft II was created by using the Galaxy Editor. One can open every Campaign Map and check out every aspect (Terrain, Triggers, Regions and so on) of it. [5] The Editor allows for even replays of played games to be analyzed and debugged.

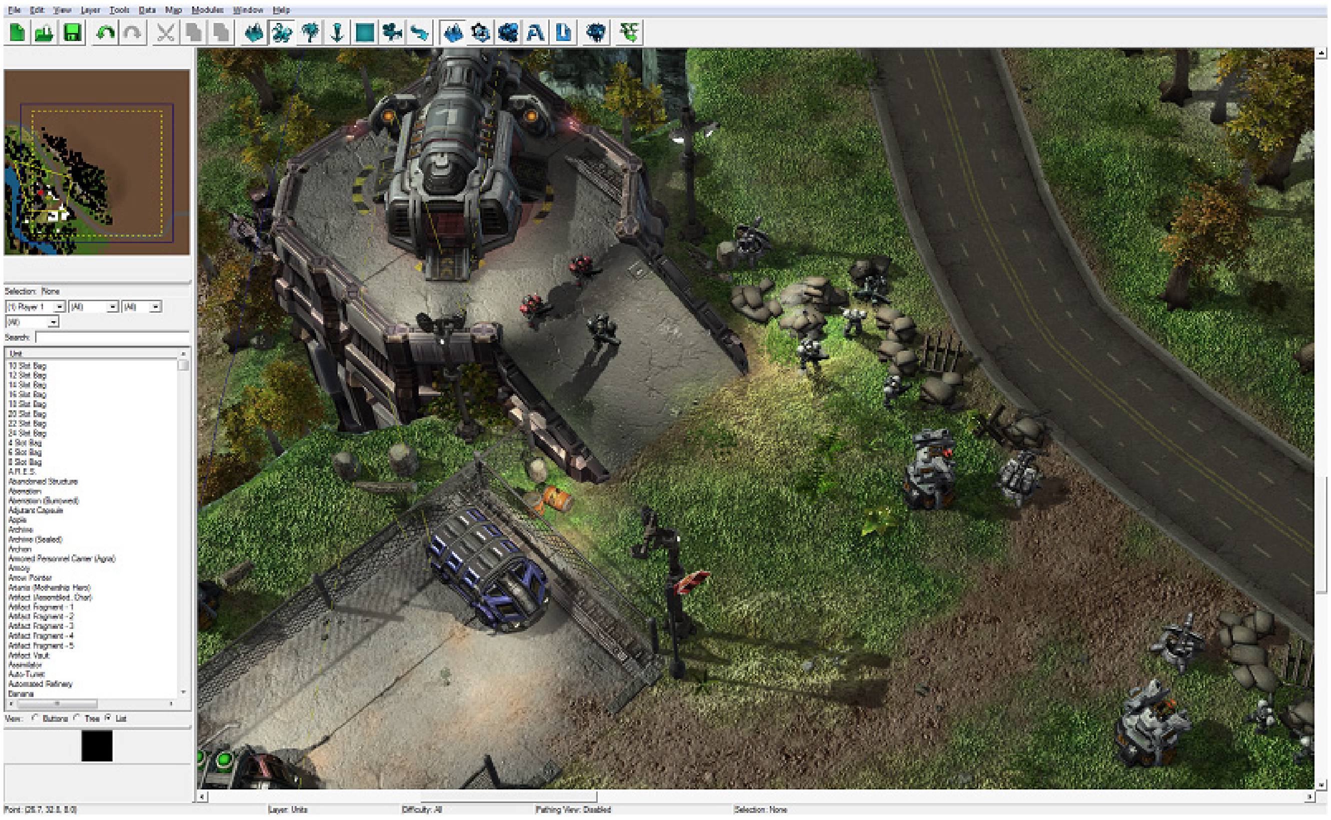
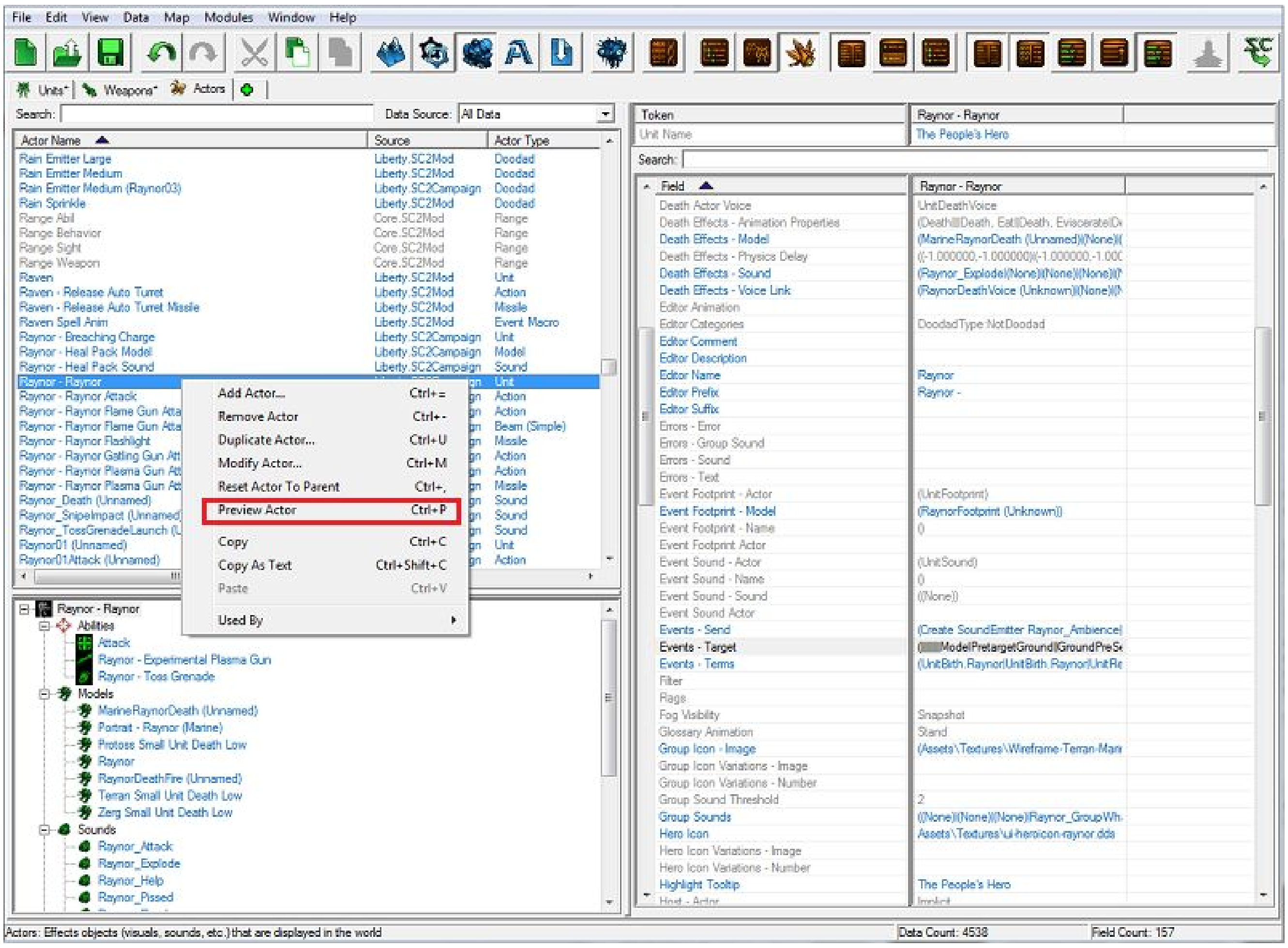


Figure 4.4.: Screenshot of the Galaxy Editor.

While the Galaxy Editor improves over its predecessors in Terrain, Data and Triggers and allows to mod almost every single aspect of the game, it is also much more complex. Complexity means it is quite difficult to start off and learn how to use the Editor. Even with the official tutorials in `http:`

[3] `https://www.facebook.com/notes/yaphets-pis/a-history-of-dota-part-1/377203832338260/`
[4] `http://us.battle.net/sc2/en/game/maps-and-mods`
[5] `http://wiki.teamliquid.net/starcraft2/StarCraft_II_Editor`

//us.battle.net/sc2/en/game/maps-and-mods, most of the community agrees that it is a difficult tool to get started with. The Data Editor is quite imposing the first time. For example, imagine you just want to modify the damage a unit inflicts when attacking. You go to the Units section, but find out you need to go to the Weapons section and find the weapon associated to the unit. There you are redirected again to the Effects the weapon causes. And finally, once you find the "Damage Effect" of the weapon can you find a numerical field where you can modify the inflicted damage. It is quite tedious for a beginner.



Actor Name	Source	Actor Type
Rain Emitter Large	Liberty.SC2Mod	Doodad
Rain Emitter Medium	Liberty.SC2Mod	Doodad
Rain Emitter Medium (Raynor03)	Liberty.SC2Campaign	Doodad
Rain Sprinkle	Liberty.SC2Mod	Doodad
Range Abil	Core.SC2Mod	Range
Range Behavior	Core.SC2Mod	Range
Range Sight	Core.SC2Mod	Range
Range Weapon	Core.SC2Mod	Range
Raven	Liberty.SC2Mod	Unit
Raven - Release Auto Turret	Liberty.SC2Mod	Action
Raven - Release Auto Turret Missile	Liberty.SC2Mod	Missile
Raven Spell Anim	Liberty.SC2Mod	Event Macro
Raynor - Breaching Charge	Liberty.SC2Campaign	Unit
Raynor - Heal Pack Model	Liberty.SC2Campaign	Model
Raynor - Heal Pack Sound	Liberty.SC2Campaign	Sound
Raynor - Raynor	gn	Unit
Raynor - Raynor Attack	gn	Action
Raynor - Raynor Flame Gun Atta	gn	Action
Raynor - Raynor Flame Gun Atta	gn	Beam (Simple)
Raynor - Raynor Flashlight	gn	Missile
Raynor - Raynor Gatling Gun Att	gn	Action
Raynor - Raynor Plasma Gun Att	gn	Action
Raynor - Raynor Plasma Gun Att	gn	Missile
Raynor_Death (Unnamed)	gn	Sound
Raynor_SnipeImpact (Unnamed)	gn	Sound
Raynor_TossGrenadeLaunch (U	gn	Sound
Raynor01 (Unnamed)	gn	Unit
Raynor01Attack (Unnamed)	gn	Action

Raynor - Raynor
- Abilities
  - Attack
  - Raynor - Experimental Plasma Gun
  - Raynor - Toss Grenade
- Models
  - MarineRaynorDeath (Unnamed)
  - Portrait - Raynor (Marine)
  - Protoss Small Unit Death Low
  - Raynor
  - RaynorDeathFire (Unnamed)
  - Terran Small Unit Death Low
  - Zerg Small Unit Death Low
- Sounds
  - Raynor_Attack
  - Raynor_Explode
  - Raynor_Help
  - Raynor_Pissed

Token	Raynor - Raynor
Unit Name	The People's Hero

Search:

Field	Raynor - Raynor
Death Actor Voice	UnitDeathVoice
Death Effects - Animation Properties	(Death||Death, Eat||Death, Eviscerate|D
Death Effects - Model	(MarineRaynorDeath (Unnamed)|(None)|(
Death Effects - Physics Delay	((-1.000000,-1.000000)|(-1.000000,-1.000
Death Effects - Sound	(Raynor_Explode|(None)|(None)|(None)|(
Death Effects - Voice Link	(RaynorDeathVoice (Unknown)|(None)|(
Editor Animation	
Editor Categories	DoodadType:NotDoodad
Editor Comment	
Editor Description	
Editor Name	Raynor
Editor Prefix	Raynor -
Editor Suffix	
Errors - Error	
Errors - Group Sound	
Errors - Sound	
Errors - Text	
Event Footprint - Actor	(UnitFootprint)
Event Footprint - Model	(RaynorFootprint (Unknown))
Event Footprint - Name	()
Event Footprint Actor	
Event Sound - Actor	(UnitSound)
Event Sound - Name	()
Event Sound - Sound	((None))
Event Sound Actor	
Events - Send	(Create SoundEmitter Raynor_Ambience|
Events - Target	(|||ModelPretargetGround||GroundPreS
Events - Terms	(UnitBirth.Raynor|UnitBirth.Raynor|UnitRe
Filter	
Flags	
Fog Visibility	Snapshot
Glossary Animation	Stand
Group Icon - Image	(Assets\Textures\Wireframe-Terran-Mar
Group Icon Variations - Image	
Group Icon Variations - Number	
Group Sound Threshold	2
Group Sounds	((None)|(None)|(None)|Raynor_GroupWh
Hero Icon	Assets\Textures\ui-heroicon-raynor.dds
Hero Icon Variations - Image	
Hero Icon Variations - Number	
Highlight Tooltip	The People's Hero
Host - Actor	Implicit

Actors: Effects objects (visuals, sounds, etc.) that are displayed in the world

Data Count: 4538

Field Count: 157

Figure 4.5.: Screenshot of the Data Module in the Galaxy Editor. Possibilities for modification are endless, but one can be overwhelmed in the beginning if one attempts to understand what all the fields are for.

Despite its initially overwhelming complexity, the Galaxy Editor is considered to be the best RTS Development Tool there currently is, and one only needs to know just a bit of all that the Editor can do in order to create interesting and excellent maps. I personally had not had any experience with the Galaxy Editor before this work, but with the tutorials and with a lot of patience and effort, I was able in just over two months to create the Co-op Craft game that was used to test DDA in co-op games in this work.

4.3 Co-op Craft: Game concept

With the StarCraft game and editor background explained, we can now delve into the Co-op Craft map/game itself, which was developed for the sole purpose of this work. First of all it is important to not confuse this custom map with the official Co-op mode in StarCraft II. In the Co-op Mode (formerly known as "Allied Commanders") players take on the role of various commanders from the StarCraft universe, each of them possessing unique abilities and upgrades and bestowing special bonuses on their armies. Players are able to battle through a series of special scenarios together, leveling up their commanders' capabilities as they progress. [6] This could have been an interesting option for testing DDA in co-op games, but even though different commanders have their own combinations and upgrades, their roles are not totally distinct as explained in Section 3.2. Good players are able to "solo" (beat by themselves) the missions even on the hardest mode without help from their ally. Besides, the co-op maps, unlike the single player Campaign, are not open to the public.

All in all, despite the traditional pains of programming, I have really enjoyed programming this game! And it really provides a distinct-role co-op experience where we will be able to apply DDA, as we will see in Section 4.4.

4.3.1 Game definition

Co-op Craft (the name is a cross between "Co-op" and "StarCraft") is a **real-time battle arena distinct-role collaborative two-player** game. Both players control a hero unit with high combat stats and abilities. Both heroes complement each other, as we will see when we describe each one individually in Section 4.3.3. Waves of weaker computer-controlled units spawn periodically and stream through a main lane. There are waves of both allies and enemies, each pushing in the opposite direction in a tug-of-war. The objective for the players is to destroy the map boss unit (a Xel'Naga Construct) while protecting a friendly structure (the Holy Temple). The game resembles mostly a MOBA, with the difference that the two human players do not face equivalent enemy heroes. Co-op Craft also includes some RTS aspects that MOBAs do not have, mainly being able to recruit extra troops for the allied minion waves.

[6] `http://starcraft.wikia.com/wiki/Co-op_Missions`

4.3.2 Map layout and Gameplay

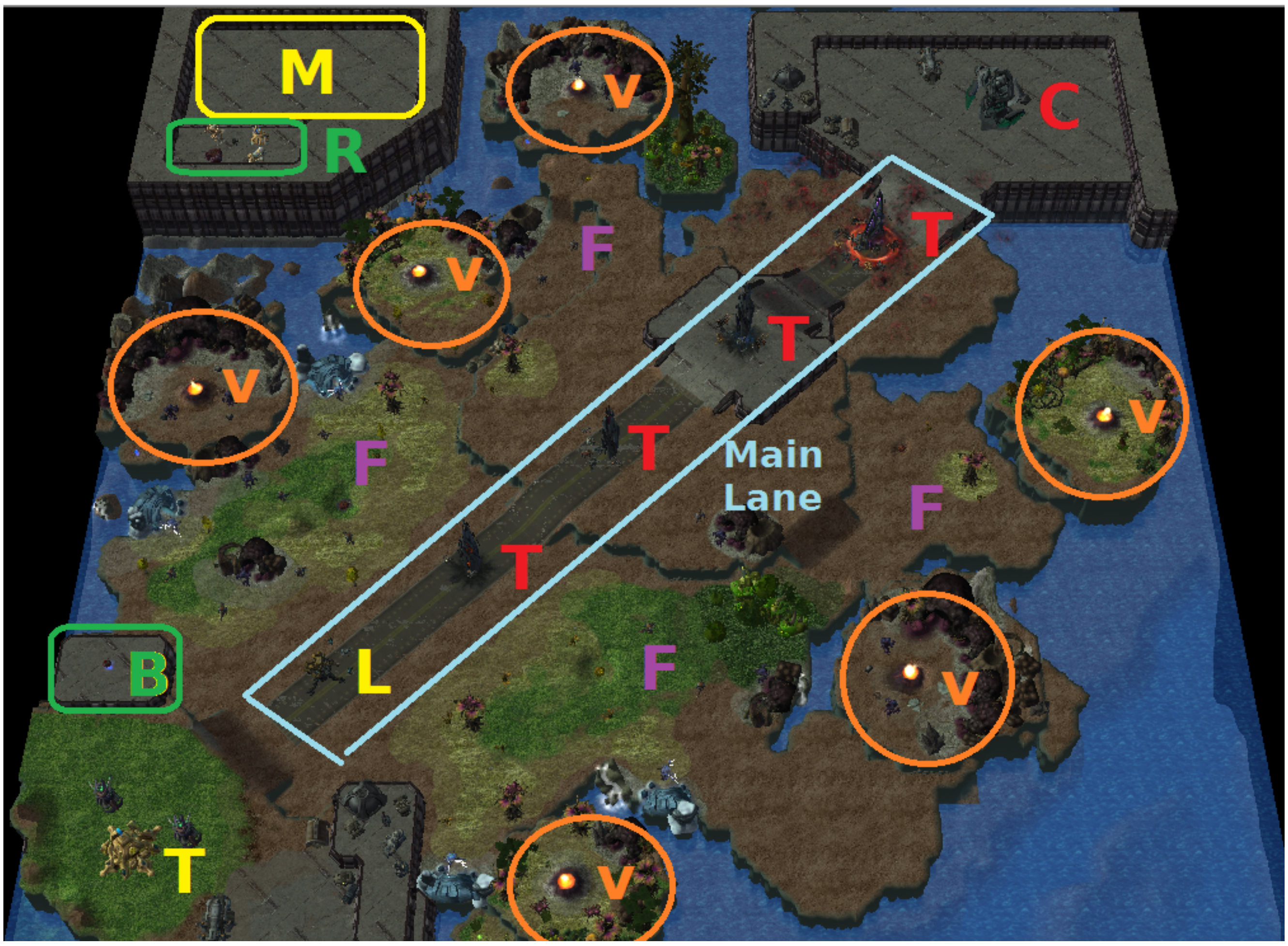


Figure 4.6.: Co-op Craft Map Layout.
LEGEND: C=Construct (Map Boss); T(red)=Enemy Tower;
L=Laser Drill (allied Tower); T(yellow)=Temple; B=Base;
M=Minion Sample Units; R=Recruiting and Researching structures;
V=Volcano; F=Feral Monsters

A map layout of Co-op Craft can be seen in Figure 4.6. Unlike other popular MOBA maps, Co-op Craft only includes one main lane where **minions** (the weaker computer-controlled units) will be marching through. Allied minion waves spawn periodically in the Temple grounds and push toward the Construct, whereas the enemy minion waves spawn in the Construct platform and push toward the Temple. In the way there is one allied Tower (a **Laser Drill** that can hit any enemy in the map as long as it is visible for the players) and **four enemy Void Towers** that increase in health and have better defensive abilities the closer they are to the final Construct.

Players obtain **resources** as they kill enemy minions, but they can also "farm" more abundant resources if they kill **Feral Monsters** outside the Main Lane. These Feral Monsters do not push in the Main Lane, but they are still hostile to the players and have much higher health and attack than regular minions. Feral Monsters and minion waves are divided into air and ground units to emphasize the distinct-role nature of the game. There are also **six volcanoes** on the map that periodically erupt resources, which can then be collected by simply touching them with the hero unit. There is a "catch" though: the volcanoes are situated on high ground plateaus, erupted resources have an expiry time (they disappear if not collected on time), and there is also a high concentration of feral monsters on the volcano plateaus. If players can collect those resources it definitely pays off, as they provide much more income than killing regular minions.

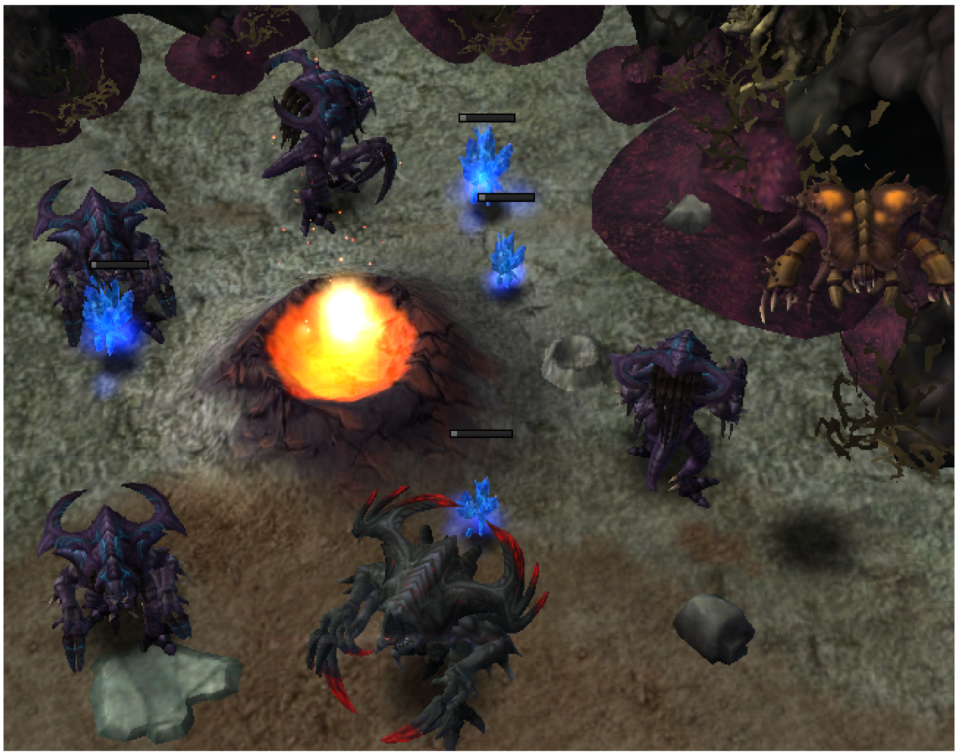


Figure 4.7.: A volcano guarded by feral monsters. Minerals have erupted.

Gathered resources can be spent to either recruit more minions for the allied waves or to upgrade the hero's stats, spells and abilities. In the top left corner of the map players can see what units are currently spawning in the allied waves (M=Minion Sample Units) and they can also select the **Recruiting and Researching Structures**. These structures can also be accessed through the normal UI and players can buy new upgrades or units in any moment of the game. There are different types of recruitable units, but also a limit to each unit type (which keeps players from "spamming" only one type of unit). The upgrades in the Research Structure are divided into: basic upgrades (Increase Attack, Armor and Health of a hero); spells (improve the spells the hero already has or unlock new ones) and army improvements (the minion wave units and the Temple can also be upgraded). Players are always given two options to improve an existing spell or research a new spell, in order to give the game some replay value. Spells include damaging spells, defensive spells, AoE (Area of Effect) spells and powerful ultimate spells. Spending gathered resources in both recruiting new minions and upgrading the hero is crucial for success in the game.

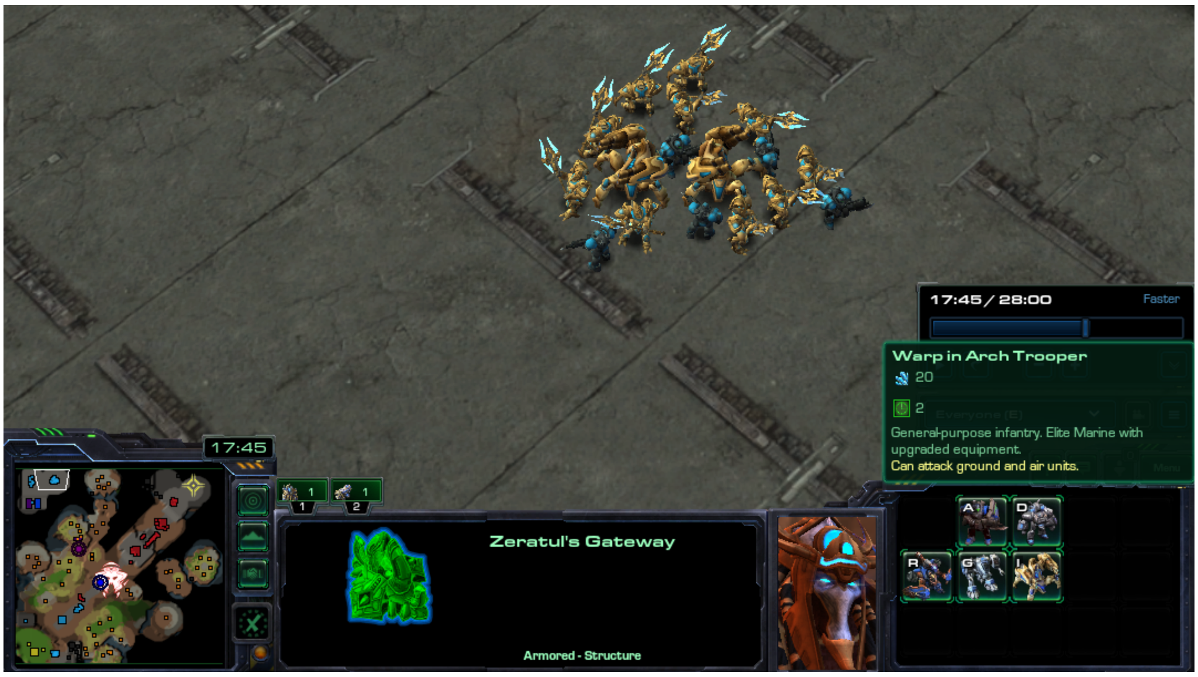

17:45 / 28:00
Faster
Warp in Arch Trooper
20
2
General-purpose infantry. Elite Marine with upgraded equipment.
Can attack ground and air units.
17:45
Zeratul's Gateway
Armored - Structure

Figure 4.8.: Sample minions at the top left screen of the map. A Recruiting Structure (Zeratul's Forge) is selected and the player can recruit more minions for the allied waves.

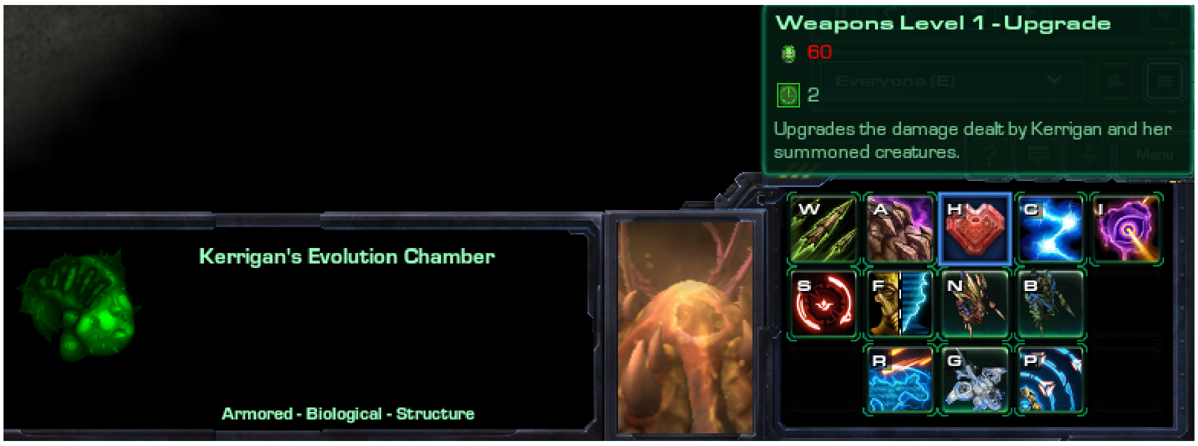

Weapons Level 1 - Upgrade
60
2
Upgrades the damage dealt by Kerrigan and her summoned creatures.
Kerrigan's Evolution Chamber
Armored - Biological - Structure

Figure 4.9.: A Research Structure (Kerrigan's Evolution Chamber) where a player can upgrade the hero and the minions.

The game is intended to see **players perform in three major ways**: (1) pushing and fighting along the allied minion waves in the main lane (main objective); (2) harvesting resources by "farming" on feral monsters; (3) sneakily collecting the volcano resources. Both the player's ability to meaningfully move around the map as well as their ability to cast spells and fight effectively are put to the test and reflect the overall player's performance. All along both players must also communicate, develop strategies and work together to fulfill their goals.

As time elapses in the game the enemy waves acquire more and more minions. The players must recruit more allied units and improve their heroes in order to progress and avoid losing the Temple. Every several minutes (the equivalent to twenty minion waves or so) the enemy spawns a gigantic boss creature called the **Titan** which will push along the enemy troops towards the Temple. This is intended to keep the players on guard and to spice up the game. The Titan is powerful, has a lot of health and can attack both ground and air units simultaneously. But on the other hand it can be attacked by both air-attacker units and by ground-attacker units. As well as the Titan, a big feral monster called the Lizdarak and the enemy and allied towers can be attacked by both ground-attackers and air-attackers. This overlapping aims to have both players and heroes work together and collaborate in achieving their main goals, while still keeping distinct roles (One hero can only directly attack ground units and the other hero can only directly attack air units, with the exceptions that have already been mentioned).

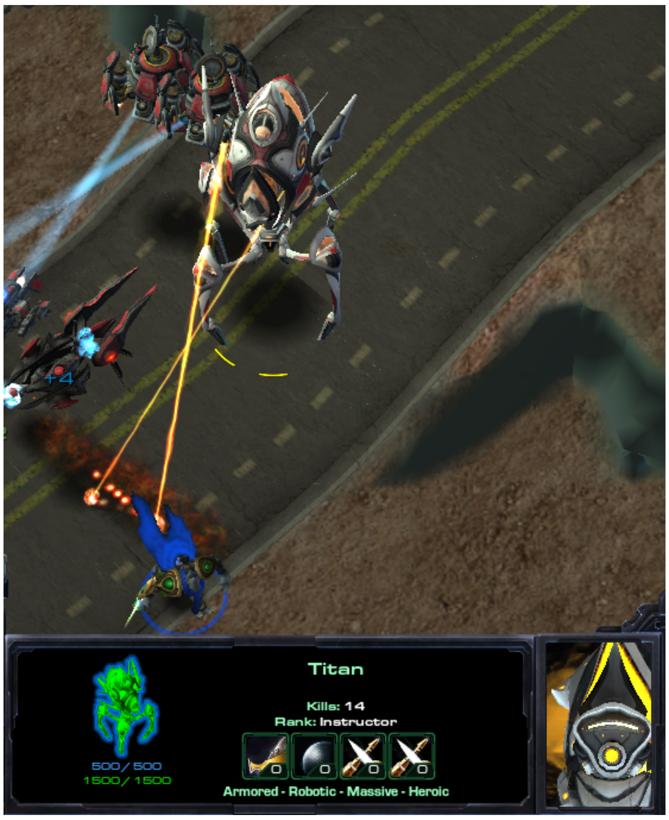


Figure 4.10.: One of the map bosses (the **Titan**) attacks the melee hero (Zeratul). Titans have a lot of health and are deadly, so both heroes must combine their efforts to stop them.

Due to Co-op Craft using an RTS engine, it is a fast-paced game. It has however quite a lot of information for people who play the game for the first time (even if it is similar to a very simplified version of the popular MOBAs!), and for this reason I made an effort to include all details and all information as hints and text in-game. Before the actual game of Co-op Craft there is a skippable **tutorial** and a non-skippable **training phase** where both players work together in a preliminary stage. The aim is to have them get used to the controls before the actual game starts. All objectives and gameplay are also clearly explained in text for those who are new to the game.

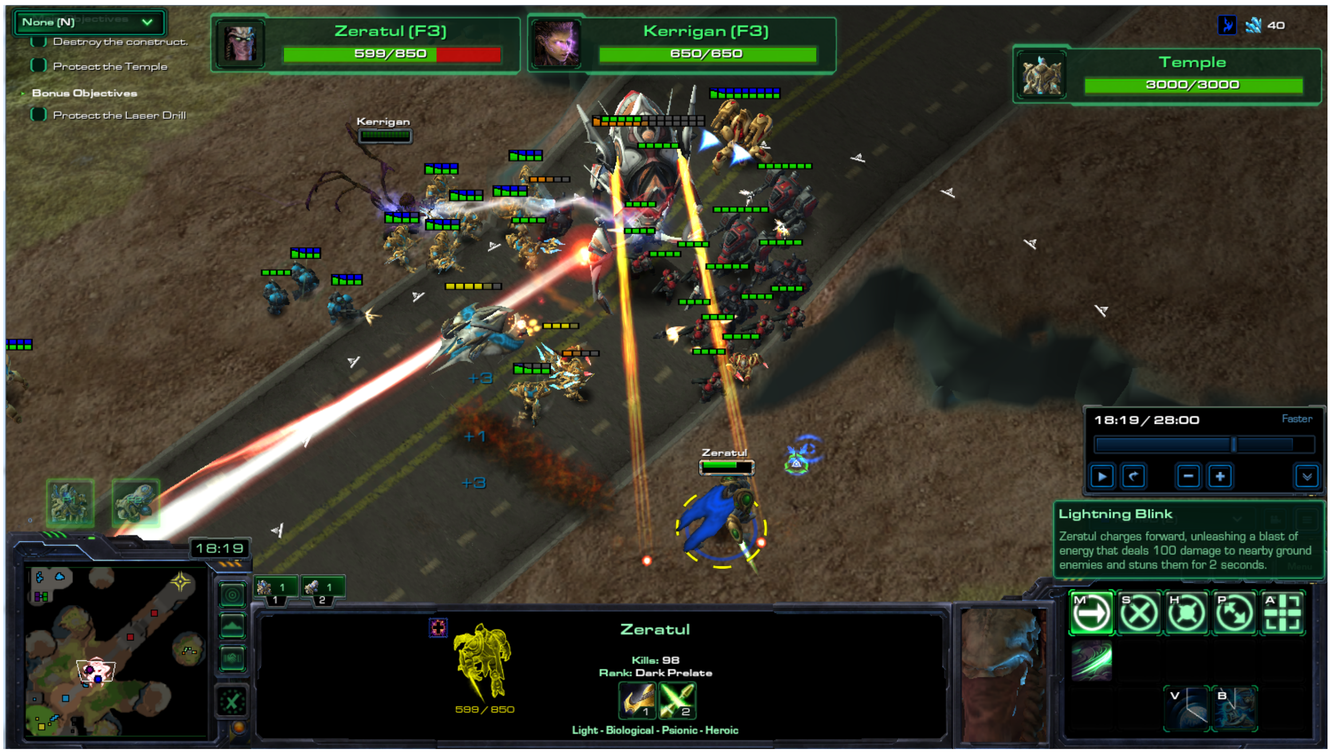


Figure 4.11.: A screenshot that depicts the action in Co-op Craft. The melee hero (Zeratul) and the ranged hero (Kerrigan) work together to destroy a Titan. Blue-colored units are allied minions, whereas red-colored units are enemy minions.

4.3.3 Description of Player characters (Heroes)

There are two heroes in the game, which means one player controls one and the other player controls the other. Both heroes start off with two basic spells. Heroes do not have mana or energy, but their spells do have a **cooldown** (a minimum time that has to elapse between two consecutive uses of a spell). This prevents heroes from spamming their spells and forces the players to use them wisely. The first spell allows the hero to **teleport/jump** to a nearby location, ignoring platform level differences. This allows heroes to teleport to higher or lower ground, something which they always need to do to enter or leave the base or volcano plateaus, for example. Heroes can restore health by returning to the base platform. If heroes die, they respawn at the base platform after waiting for some time. Teleporting also allows heroes to move faster throughout the map. The second spell is the **signature spell** of the hero, and is thus different depending on the hero. Both heroes have distinct roles because they can only attack one type of enemy units (air or ground), recruit one type of allied units (air or ground) and gather one type of resources (minerals or vespene gas). Both heroes complement each other and must work together to achieve their common goals.

The first hero is called **Zeratul**. He is a **melee** warrior that **can attack ground units**. He is responsible for **recruiting ground allied units** and collecting **minerals**. He has kind of a "**tank**" **role**, i.e. he is good at taking hits and prolonged damage. The role is most notable after defense upgrades. His spells are more oriented to assist allied units or cripple enemy units. His signature spell is "**Void Shield**" which grants him and nearby ground allied units a shield that lasts for a few seconds, greatly increasing the survivability of him and his allies.

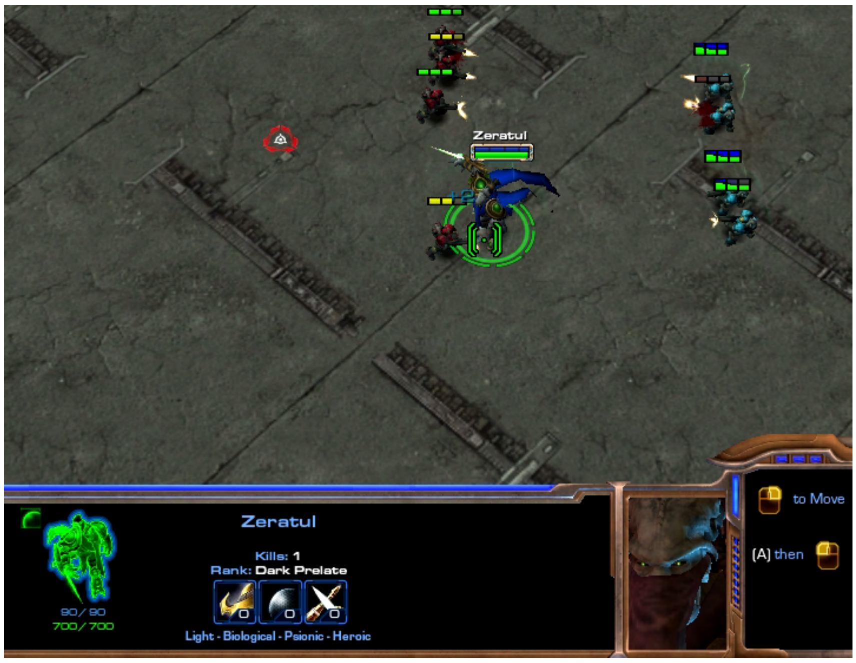


Figure 4.12.: **Zeratul**, the melee ground-attacking hero. He has used Void Shield to protect himself and his allies.

The second hero is called **Kerrigan**. She is a **ranged** warrior that **can attack airborne units**. She is responsible for **recruiting air allied units** and collecting **vespene gas**. She has kind of an "**assassin**" **role**, i.e. she is good at inflicting damage from far away but is more vulnerable to damage. The role is most notable after attack upgrades. Her spells are more oriented to increase her damage output. Her signature spell is "**Spawn Hydralisks**", which invokes a few extra allied monsters that last for a few seconds and attack enemies, thus increasing Kerrigan's offensive prowess.

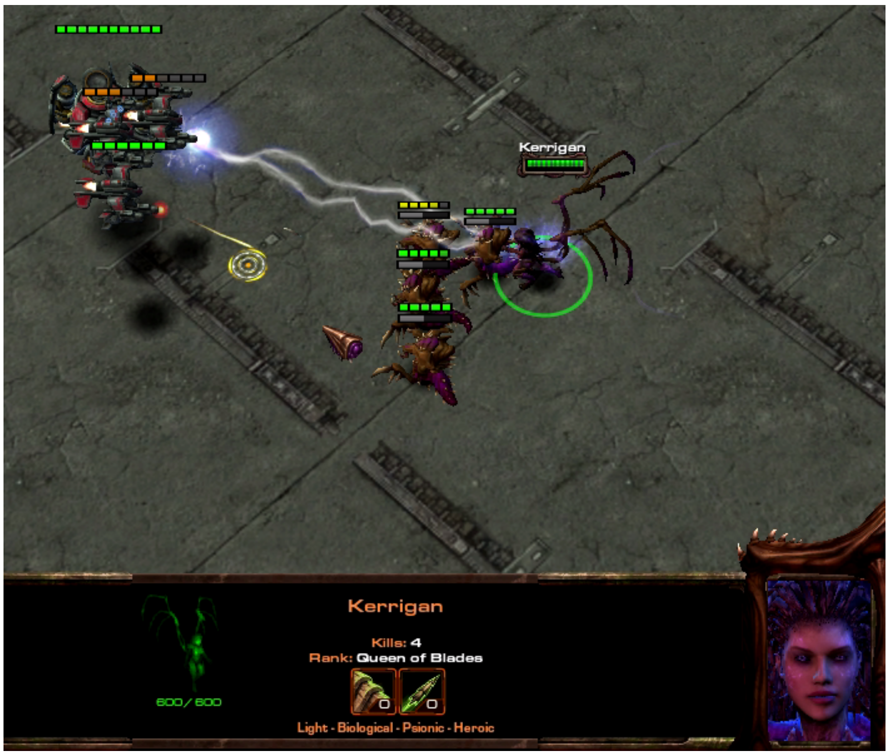


Figure 4.13.: **Kerrigan**, the ranged air-attacking hero. She has spawned Hydralisks (allied monsters) to help her attack the airborne enemies.

Here is a summary table of the distinct roles of both heroes:

Hero	Zeratul	Kerrigan
Warrior	Melee	Ranged
Role	Tank	Assassin
Attacks	Ground	Air
Recruits	Ground	Air
Collects	Minerals	Vespene Gas

4.4 Dynamic Difficulty Adaptation in Co-op Craft

We will now discuss in detail the implemented DDAAs in Co-op Craft. There is of course many other possible DDAAs that could have been developed. The first three subsections will focus on the ones that were actually implemented, and give reasons to why they were included. In the last subsection we will present some alternatives. In Co-op Craft three DDAAs were implemented, and were even given names: **Resources DDAA**, **Respawn DDAA** and **Milestone DDAA**.

An important comment to be made is that DDA in Co-op Craft is only active during the first 30 minutes of the game. The reason to this is that few games from beta-testing made it later than the 30 minute mark, and thus data for the standard values could not be effectively collected after the 30 minute mark. This is not a big problem, however, as Co-op Craft is intended to last an average of 30 minutes. If a game lasts longer, the last DDA changes made at the 30 minute mark will be permanent until the end of the game, which is reasonable as by that time we can assume that the Co-op Craft DDA will have figured out a sensible level of difficulty for the game.

4.4.1 DDAA #1: Resources DDAA

Resources allow players to improve the allied minion's waves and the heroes, so they are a good target for DDA. Game difficulty can be increased or decreased by providing more or less resources, respectively. Additionally, gathering resources reflects the skill of a player in killing enemy units and also collecting volcano resources, two pivotal features in the gameplay.

We decided to create a significant parameter related to resources that would reflect this mentioned skill: "**collected_resources**" (or simply "C_R"). In Zeratul's case it would be "collected_minerals" ("C_M") and in Kerrigan's case it would be "collected_vespene" ("C_V"). It is defined as the total accumulated amount of resources collected so far in the game. In contrast to the parameter "available resources", which decreases every time a player buys or invests in an upgrade, C_R only increases throughout the course of a game, and would be equivalent to "available resources" if the player did not buy anything throughout the game. This new parameter C_R reflects both the skill of the player killing enemy units and gathering the available resources in a volcano, and the higher it is the better the player's performance is.

The Resources DDAA probes the C_R parameter for both heroes every two minutes, compares it to a standard S and tweaks other parameters accordingly. First it defines yet a new parameter, "K", which is defined as:

$$K = \frac{C_R + 1}{S + 1} \tag{4.1}$$

The "+1" is there to avoid possible divisions by zero and also to provide a cushion-effect with smaller numbers. If $K < 1$ the player is under-performing and the DDAA will make the game easier, whereas if

$K > 1$ the player is over-performing and the game will be made harder. With this new K the following parameters are tweaked: time between volcano eruption times, time erupted resources last before disappearing and attack speed of feral monsters. If, for example, the player is under-performing, volcanoes will erupt resources more often, the resources will last longer so the player has more time to pick them up and the feral monsters will attack slower. The aim is for the player to have an easier time collecting volcano resources and thus increasing the C_R parameter and bringing it closer to the standard S. If the player is over-performing, the effects would all be the opposite, also intending to bring the player closer to the standard S.

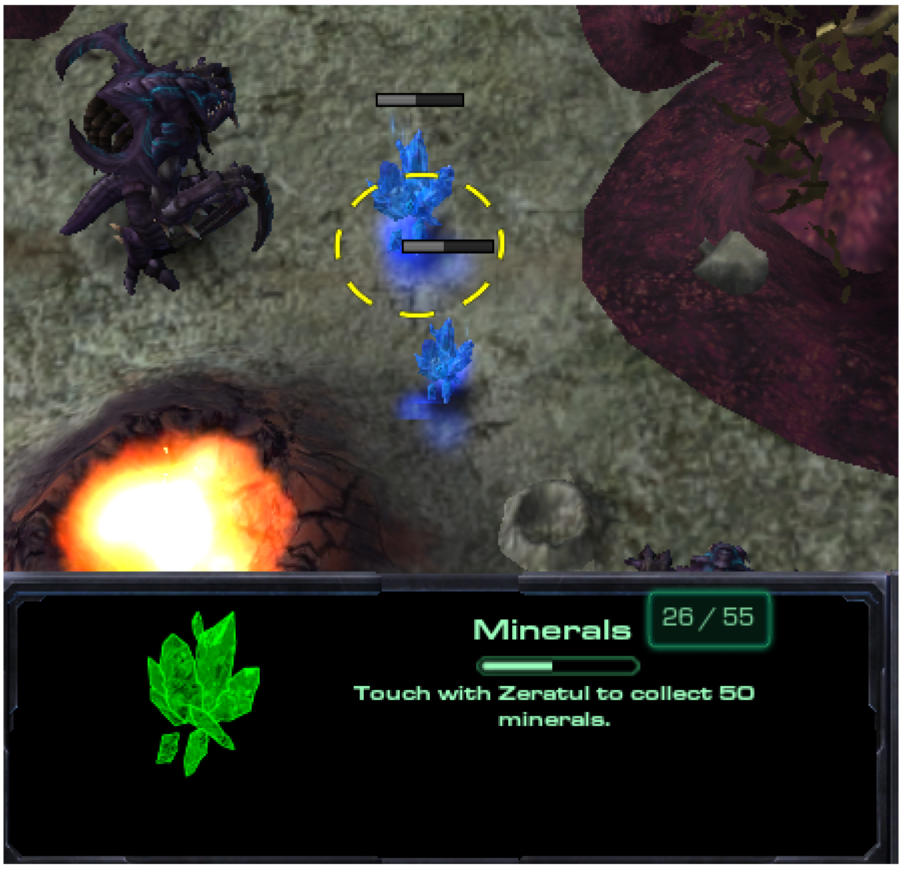


Figure 4.14.: Among other parameters, the Resources DDAA tweaks the expiry time of erupted resources.

The period was chosen to be two minutes because a smaller period would be oversensitive to spikes in C_R whereas a longer period would fail to provide smooth changes to the changing parameters (the tweaked parameters' values grow old quickly). Two minutes also happens to be the standard time between volcano eruptions, so the player is expected to run to the volcano and collect resources at some point in those two minutes, so significant changes in C_R are also expected to happen once every two minutes.

The DDAA acts separately for the two players: on the one hand it compares "collected_minerals" C_M to "standard_minerals" S_M, and tweaks the respective parameters for the Zeratul player: time between mineral eruptions, time minerals last and attack speed of ground feral monsters; on the other hand it compares "collected_vespene" C_V to "standard_vespene" S_V and tweaks the respective parameters for the Kerrigan player. The standard was taken as a mean average from several play-throughs performed by the beta testers: every two minutes both C_M and C_V were registered and with the data from all these games, the average was taken. Thus we got an average value for C_M at the 2-minute mark, another at the 4-minute mark, and so on; and the same with C_V. These averages were set to be the standard, so when a player is under-performing, (s)he is performing below average.

Following the classification made in Section 3.3, the Resources DDAA can be classified as a **periodic time-based triggering**, **parameter probing**, both **reactive** (it adjusts the attack speed of existing feral monster units) **AND proactive** (existing erupted resources do not have their remaining expiry time modified, but rather future erupted resources will last shorter or longer), **continuous** (the tweaked variables can take any value in an interval of allowed values), **implicit** (the player is not directly notified of these changes every two minutes), **auto-corrective** (significant parameters like feral monsters' attack speed are tweaked), **standardizer**, **single-player** DDAA.

4.4.2 DDAA #2: Respawning DDAA

When a player dies (s)he is respawned back in the base, but must wait a while until (s)he can return to action. The longer the game has been in progress, the longer a player has to wait to respawn after dying. It is however feasible to tweak this respawn waiting time in order to prevent a weaker player (who presumably dies often) from becoming frustrated.

The standard equation to calculate the **respawn waiting time** R_t when a player dies is:

$$R_t[secs] = \frac{t[secs]}{20} \tag{4.2}$$

where t is the elapsed game time. Additionally, R_t needs to be in the interval [20 secs, 4 minutes], otherwise the gameplay is flawed (waiting forever or dying without barely a penalty). If the above equation brings a value outside of the interval, R_t is clamped to the minimum or maximum value accordingly.

The Respawning DDAA modifies the previous equation to:

$$R_t[secs] = \frac{t[secs]}{10} - 15D \tag{4.3}$$

where D is the hero **death count**, i.e. the number of times the hero has been killed during the game.

As can be seen from this last equation, every time a player dies (s)he has to wait 15 seconds less to respawn. This means that if a player dies often (s)he will not be always out of the action due to respawn waiting, thus easing the game difficulty. However if a player dies less often, (s)he will have to wait a longer time than with the standard equation, especially in the late game where the minuend term is dominant (and is twice as big in the DDAA equation as opposed to the standard equation), thus increasing the game difficulty.

The number values in the equations were taken intuitively and following other games of a similar genre. The equations work quite well to achieve desired gameplay.

Following the classification made in Section 3.3, the Respawning DDAA can be classified as an **independent-event time-based triggering** (it only activates when a player dies, an event not set at a specific moment in time), **parameter probing**, **proactive** (it adjusts the respawning time of a player just after dying, which happens just an instant ahead in the future), **continuous** (the respawning time can take any entire value between 20 seconds and 4 minutes), **implicit** (the player is not directly notified of his respawning time), **auto-corrective** (respawning time is a significant parameter being modified), **progressive** (the number of times previously killed is taken into account), **single-player** DDAA.

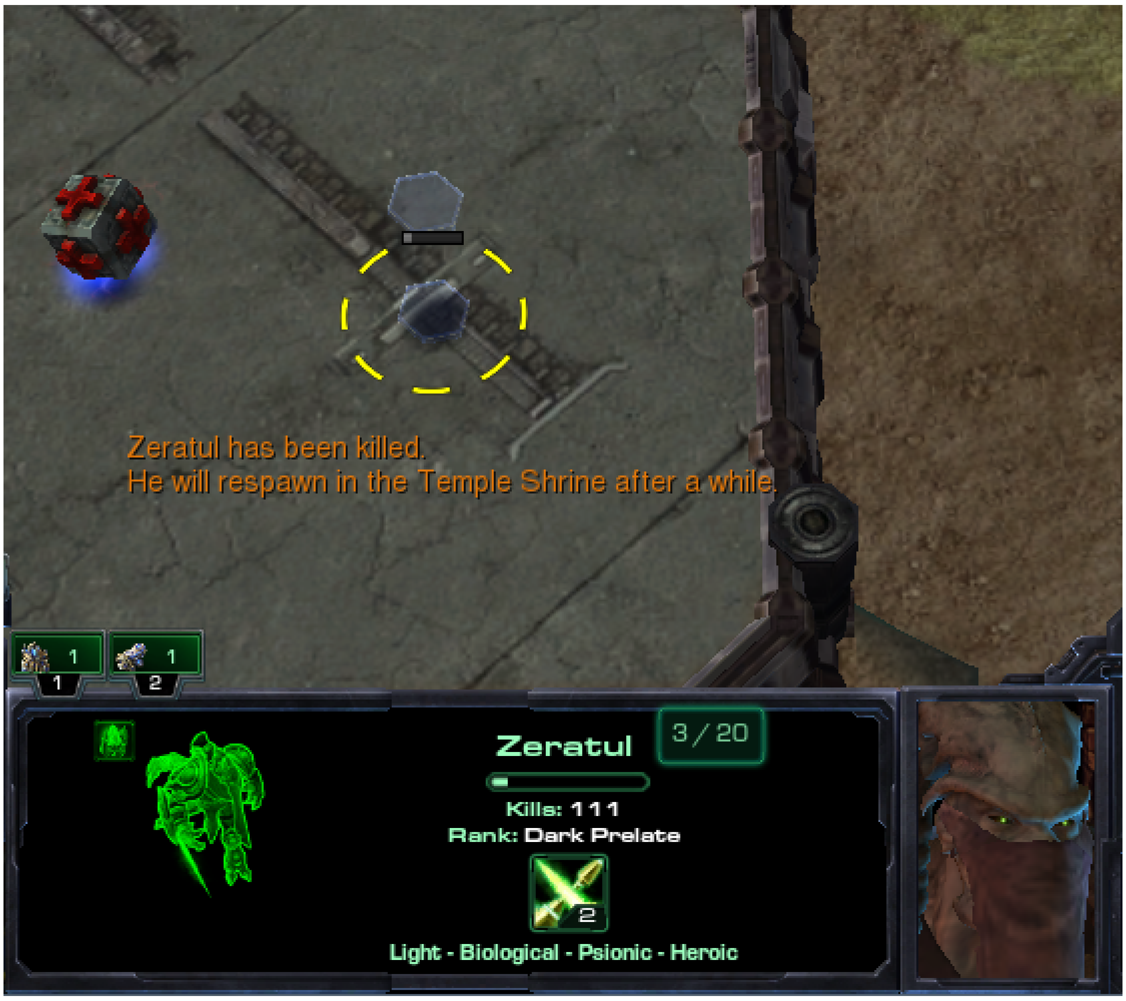

Zeratul has been killed.
He will respawn in the Temple Shrine after a while.

Figure 4.15.: The Respawning DDAA takes into account the number of times a hero has already died in order to determine the time (s)he has to wait to respawn.

4.4.3 DDAA #3: Milestone DDAA

Both previous DDAAs have been single-player, so there is still a multi-player DDAA to add if we want to emphasize the collaborative feature of the game over strictly cooperation in separate tasks. Besides, the main lane with minion waves and allied/enemy towers, which are after all directly related to the main objectives, have been left untouched until now. It comes as no surprise then that this last DDAA involves everything mentioned in this paragraph and is possibly the dominant DDAA when addressing game difficulty. It must be noted, however, that the Milestone DDAA is closely linked to the first DDAA (Resources DDAA), as we will see later.

First we define a significant parameter that reflects the team's main progress in the game: the **milestone parameter** M. M is defined as the sum of the health of all enemy structures minus the sum of the health of all allied structures:

$$M = \sum_i H_{Ei} - \sum_i H_{Ai} \tag{4.4}$$

Enemy structures are the four enemy towers, whereas allied structures are the Laser Drill, the Temple Towers and the Temple itself. Although M is not totally informative of the state of the game and the main objectives (e.g. its value could be the same whether one enemy tower is destroyed or whether two enemy towers and the Laser Drill are destroyed), M is an easy to calculate parameter which decreases when enemy towers are damaged (and the players' team is presumably making progress) and which increases when allied structures take damage (and the players' team is presumably struggling against the enemy waves). In contrast to C_R and R_t, M is **common to both players**, as it represents the progress or the struggle of the team.

As with the Resources DDAA and for similar reasons, every two minutes we will compare M to a standard S_M and adjust the enemy minion waves accordingly. Every two minutes players are expected to gather resources from volcanoes, but also to push in the main lane, so it also makes sense to observe M every two minutes. Besides, we wish to adjust the enemy waves differently for air and ground minions (see next paragraph), and for this we will make use of the K parameter used in the Resources DDAA (which is already being updated every two minutes).

Why adjust enemy waves according to individual player? In a scenario where a stronger player teams up with a weaker player and dominates the progression in the main lane, we wish to make the game more difficult for the stronger player without making it impossible for the weaker player. Indeed, we wish to adjust the difficulty while overcoming the heterogeneously skilled teams challenge (See Section 3.1) A reasonable way to do this is to increase the amount of enemies that the stronger player is responsible for, but not so much the amount of enemies the weaker player is responsible for. Thus, changes in enemy waves depend on both M (common) and the player's performance in collecting resources, represented with K (individual).

Whereas the standard obtained for C_R was the average from games played by beta testers, the standard for M is a little bit trickier. Even though the average was measured in these games played by beta-testers, the standard deviation from the average was much higher and thus the average could not be trusted. Instead, I decided to observe these games and set some logical or expected milestones at certain points in time, and adjusted S_M accordingly:

- By the 8 minute mark the first enemy tower should have been destroyed and the Laser Drill taken 15 % damage: $S_M(t = 8min) = 3000$
- By the 12 minute mark the second enemy tower should have been destroyed and the Laser Drill taken 30 % damage: $S_M(t = 12min) = 1400$
- ...
- By the 28 minute mark all enemy towers and the Laser Drill should have been destroyed: $S_M(t = 28min) = -4600$

S_M in intermediate points in time was taken to be the median between the previous and the next S_M. For example:

$$S_M(t = 10min) = \frac{S_M(t = 8min) + S_M(t = 12min)}{2} = 2200$$

This is how all the remaining values for S_M until the 30 minute mark were obtained.

How do we modify the enemy's waves with M? The first criterion is to add/subtract a whole enemy wave for every 2000 health difference between the actual M and the standard S_M. A whole enemy wave is one unit of every unit type, and amounts to about 250 resources. By a simple rule of thumb ($250x8 = 2000$) we can calculate the "available resources" A_R the DDAA has to either add or subtract minions in enemy waves:

$$A_R = \frac{S_M - M}{8} \tag{4.5}$$

This A_R factor is still common to both players. But now we modify it according to each player's K factor. The second criterion is to subtract a whole enemy wave if the player is under-performing 50 % below average ($K = 0.5$) or add a whole enemy wave if the player is performing two times better than the standard ($K = 2$). Using some math we get the equation for the final A_R^* factor, different for each player:

$$A_R^* = A_R + 250log_2K \tag{4.6}$$

If $A_R^* > 0$, the DDAA will add units to the standard enemy waves, whereas if $A_R^* < 0$ the DDAA will remove units from the standard enemy waves.

To calculate how many units of each unit type should be added or removed, we thought of a function that would "buy" one unit at a time, iterating through the different unit types until it is left without enough A_R^* to buy any more units. Examples:

- $A_R^* = 40$ for the Zeratul player. The DDAA adds 2 Marines and 1 Zealot to the standard enemy ground waves ($12 + 15 + 12 \leq 40$).
- $A_R^* = -105$ for the Kerrigan player. The DDAA removes 2 Corsairs, 1 Wyvern and 1 Eagle from the standard enemy air waves ($15 + 25 + 50 + 15 \leq -105$)

NOTE: Cost of 1 Marine = 12 minerals; cost of 1 Zealot = 15 minerals; cost of 1 Corsair = 15 vespene; cost of 1 Wyvern = 25 vespene; cost of 1 Eagle = 50 vespene.

It is noteworthy to mention that these changes do not stack, i.e. every two minutes the new M is calculated and new changes to be made to the standard enemy waves are calculated. This keeps things simpler than having to observe the current amount of units for each unit type at a certain point in time before making changes.

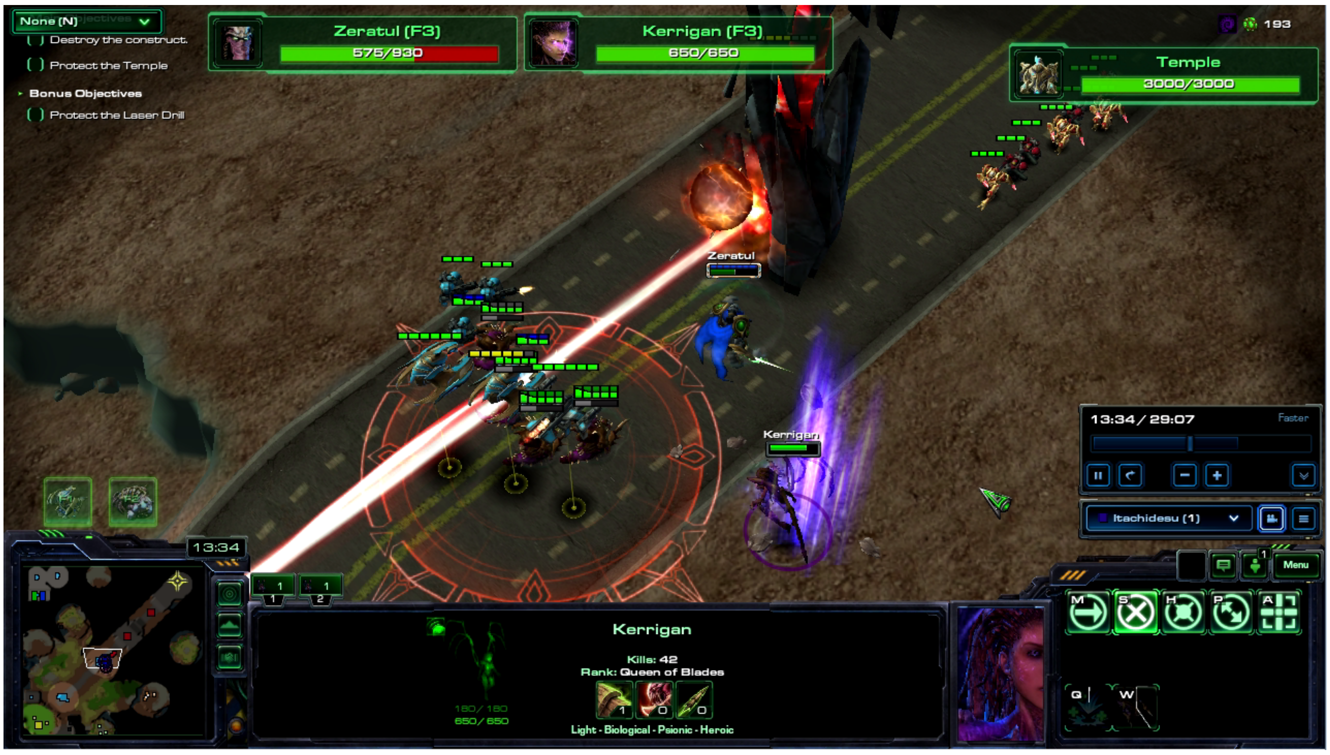


Figure 4.16.: The player heroes (Zeratul and Kerrigan) attack along with allied minions to destroy one of the Enemy Towers. The remaining health of both allied and enemy structures are used to calculate the milestone parameter M.

Following the classification made in Section 3.3, the Milestone DDAA can be classified as a **periodic time-based triggering**, **parameter probing**, **proactive** (it only modifies the amount of enemy units in minion waves that will spawn after the DDAA has acted, not waves that had already spawned before),

continuous (it can modify any number of units in the enemy waves), **implicit** (the player is not directly notified of these changes every two minutes), **auto-corrective** (the amount of units in enemy waves is a significant parameter which is being modified), **standardizer**, **multi-player** DDAA.

4.4.4 Possible alternatives for DDA in Co-op Craft

To illustrate the many possibilities DDA has, here we will mention some alternative DDAAs (or ideas for alternative DDAAs) that were not implemented in the final version of Co-op Craft for this work.

In the Resources DDAA we are measuring the player's performance through the parameter collected_resources C_R. However we could use other parameters alongside or instead of C_R. For example, we could observe a player's **APM** ("Actions Per Minute" - the number of actions (such as selecting units or issuing an order) completed within a minute of gameplay in RTS games. High APM is associated with high skill, see Lewis et al. [21]); his **kill count** (the amount of units killed by the player); or his **death count** (total times the player dies in the game). These parameters are easy to see in the StarCraft II replays and can reflect a player's performance, but we also believe that they are not too trustworthy. For instance, a player can greatly increase his APM by repeatedly and frantically clicking a unit to move to the same spot (this will count several orders when the player is actually only issuing one order - move to a certain spot). In the case of "death count", we would assume that the lower a player's "death count" is, the better he is performing, but maybe he is just staying a lot of the time in the base and out of the action and that is why he dies less often! Similarly, "kill count" only counts units that receive the fatal blow from the hero: a hero might have inflicted a lot of damage on a certain unit, but if another unit deals the killing blow, the kill count of that unit is modified and not the hero's! Not to mention that kill count counts all type of units equally: the weaker ones that die in one blow and the stronger ones that may take time to die. We have observed these parameters when testing Co-op Craft (see Section 6.2), but only to support our assertions on a player's performance rather than as the ultimate measuring yardstick. These parameters could however be part of alternative DDAAs. For example, if "Death Count" is high the game could modify something so that this player does not die too often.

A more trustworthy parameter than the above, albeit a bit more difficult to observe in the StarCraft II replays, is the total **Damage Dealt** by a player. In contrast with "Kill Count", this parameter is not conditioned by dealing the killing blow, and is also representative of the effectiveness of a player in battle. We could compare it to a standard in time like we do in the Resources DDAA. Depending on how above or below the player is from the standard, we could then modify the player's attack speed accordingly - increase it to make the game easier and decrease it to make it more difficult. In both cases "Damage Dealt" should then get closer to the standard. This DDAA would have the same characteristics as the Resources DDAA, except that it would be strictly reactive (only modifies the attack speed of the existing hero unit).

The difficulty of the minion waves is only adjusted in the implemented version of Co-op Craft with the Milestone DDAA, which only modifies the amount of minions in the waves. Alternatively, the game could also modify the minions' attack speeds, the damage received by them, or the damage the enemy minions receive from the players. One must be careful when tweaking these parameters, though, as if a player notices that a unit is clearly better or worse at different moments of the game he is more likely to have his sense of disbelief disrupted. This would be even more clear if the health or armor of enemy units is tweaked - some human players click on enemy units to see how much health they have left, and it could be obvious that the same type of unit has different total health at different moments of the game. This is why we personally decided to only modify the amount of minions in the waves and not the units themselves.

The milestone DDAA could alternatively be triggered when an allied or an enemy structure is destroyed instead of probing the health of all structures every two minutes, turning it from a periodic time-based triggering DDAA to an event-triggered DDAA. This would simplify the frequent calculations of the milestone parameter M but on the other hand the reactions of the milestone DDAA would be too coarse:

there could be several minutes of difference between the destruction of two structures and adjustment would probably have been useful in this period of time.

The ability of a player to use his hero's spells could also be observed. For example, a parameter like "Absorbed Damage" could measure how much total damage the player controlling Zeratul has absorbed with his "Void Shield" ability (see 4.3.3).

A player's performance is usually high if he uses his hero's spells often. Thus we could also implement explicit and suggestive DDAAs to remind a player to use his spells. For example, we could observe how much time elapses after a player casts a certain spell. If the player is not killed during this time and the elapsed time reaches three times the spell's cooldown time, then this DDAA could highlight the spell in the player's command card and send him a text message reminding him to use this spell. The DDAA would only do this twice for each spell so it does not become too annoying for a player who refuses to use spells by his own will. This would be a **linked-event triggering** (the DDAA triggers every time a spell has been cast and resets if the spell is cast again before three times its cooldown time has elapsed), **non-probing** (the cooldown time of the spell is needed, but this is a constant value, not a variable parameter), **NEITHER proactive nor reactive** (the DDAA does not modify anything), **discrete** (the DDAA either throws hints at the player or it does not), **explicit** (the player is directly and purposefully notified of these hints), **suggestive** (the DDAA does not modify anything, it only suggests a player to use a spell more often), **single-player** DDAA.

Another suggestive DDAA could observe the amount of feral monsters killed by a player and his collected_resources C_R. If C_R is below the standard and the player has not killed many feral monsters, the DDAA could hint the player to "farm" and kill feral monsters because he can gather more resources this way.

When implementing discrete and suggestive DDAAs like the ones mentioned above one must realize that their effectiveness is difficult to measure. They are mostly intended to help players who struggle by dropping hints, but these hints are either ON or OFF (depending on the player's performance). The difference in performance of a struggling player with or without hints can either be small or big, it is not easy to predict what impact the hints will have. And they do not have a standard in time to which significant parameters can be compared to.

4.5 Instructions to play Co-op Craft online

Co-op Craft is open to the public and free to play! Here we will detail the instructions step by step to play. If the reader does not wish to play, he may skip this section entirely.

All that is needed to play is two computers that can install the StarCraft II game (Computer requirements: `https://us.battle.net/support/en/article/starcraft-ii-system-requirements`) and an Internet connection. Here are the instructions (as of May 2016) in order to play Co-op Craft:

4.5.1 Installing StarCraft II in a computer

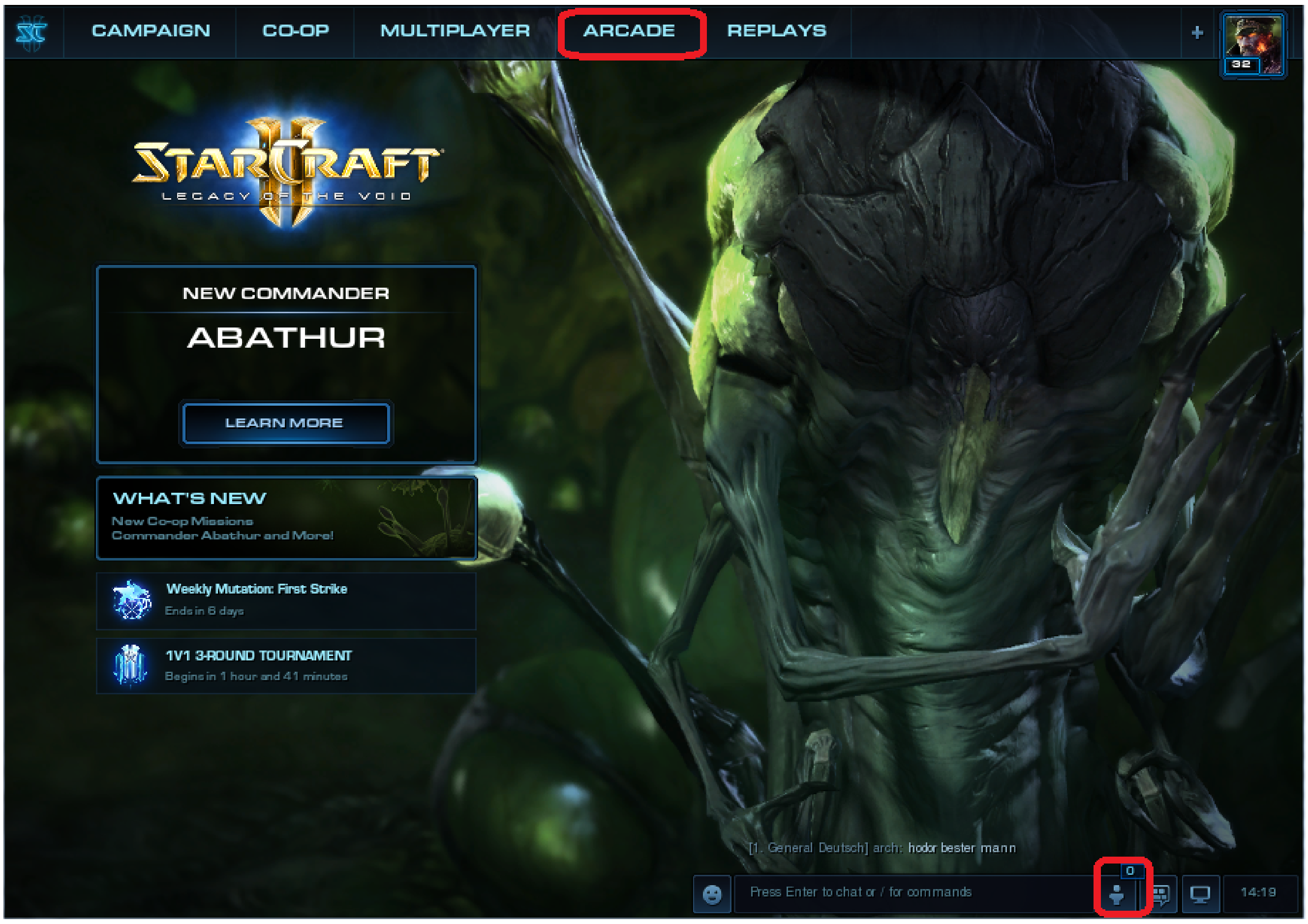
This section must be done separately with both computers. If StarCraft II is already installed in the computer, this section can be skipped.

1. If you already have a Battle.net account, skip this step. Otherwise, go to `https://eu.battle.net/account/en-us/creation/tos.html` and create a new Battle.net account (it is free of charge). Confirm the new account by clicking the link sent to your email address.
2. Log in with your Battle.net account in http://eu.battle.net/en/
3. Select StarCraft II from the available games and then click on "Try Free Now". You should get a message similar to "StarCraft II: Starter Edition has been added to your games".
4. Download the desktop App for Battle.net: `http://eu.battle.net/en/app` (insert your Battle.net account when asked)

5. Once downloaded and installed, open the Battle.net App.
6. Once opened, log in with your Battle.net account and select StarCraft II from the games on the left. It should start downloading. Total download amount is ∼24GB as of May 2016, so it will take a while.

4.5.2 Playing Co-op Craft

1. Both players open the Battle.net App and select StarCraft II from the games on the left. Make sure both of you have the latest version, patch and upgrade. Otherwise it will start downloading the new patch (be patient until it finishes)
2. Select "PLAY" and the game will start loading.
3. Once the game has loaded, if it is the first time you open StarCraft II you will probably be redirected to a cinematic and a tutorial. Skip them as soon as you can and you should be redirected to the main game screen, which should look similar to the one in Figure 4.17.



[1. General Deutsch] arch: hodor bester mann

Press Enter to chat or / for commands

Figure 4.17.: Screenshot of the main screen in StarCraft II. Highlighted in red are the "Arcade" Tab and the "Friends" button.

4. If you are not already friends with your partner in Battle.net, find your partner and add him as a friend. Click the button with the icon of a person in the bottom of the screen (see Figure 4.17). If you are playing close to your partner, you will find him in "Players near you". Right-click on your partner and select "Add Friend". Your partner must accept the friend request. If your partner is not near you you will need to find him through his BattleTag ID. Select "Add Friend" and then the option "Real ID & BattleTag Friend". Insert your partner's BattleTag ID. One can see his or her own

unique BattleTag ID in the Battle.net App when you click your user name. A BattleTag ID usually consists of a name followed by a # and some numbers.

5. Now that you are friends with your partner, the player who will be playing Kerrigan will be the host and will invite the other player to a party (DO IT THIS WAY! The host is always Kerrigan). The host right-clicks on the other player and selects "Invite to Party". The other player accepts the party invitation.
6. The host clicks the ARCADE tab in the top of the screen, and then types "Co-op Craft" in the Search Box and presses "Enter". The first option(s) as of May 2016 is/are the Co-op Craft game(s). Be sure the author's name is "KameD" (which is my Battle.net username, by the way). See the highlighted red areas in Figure 4.18.

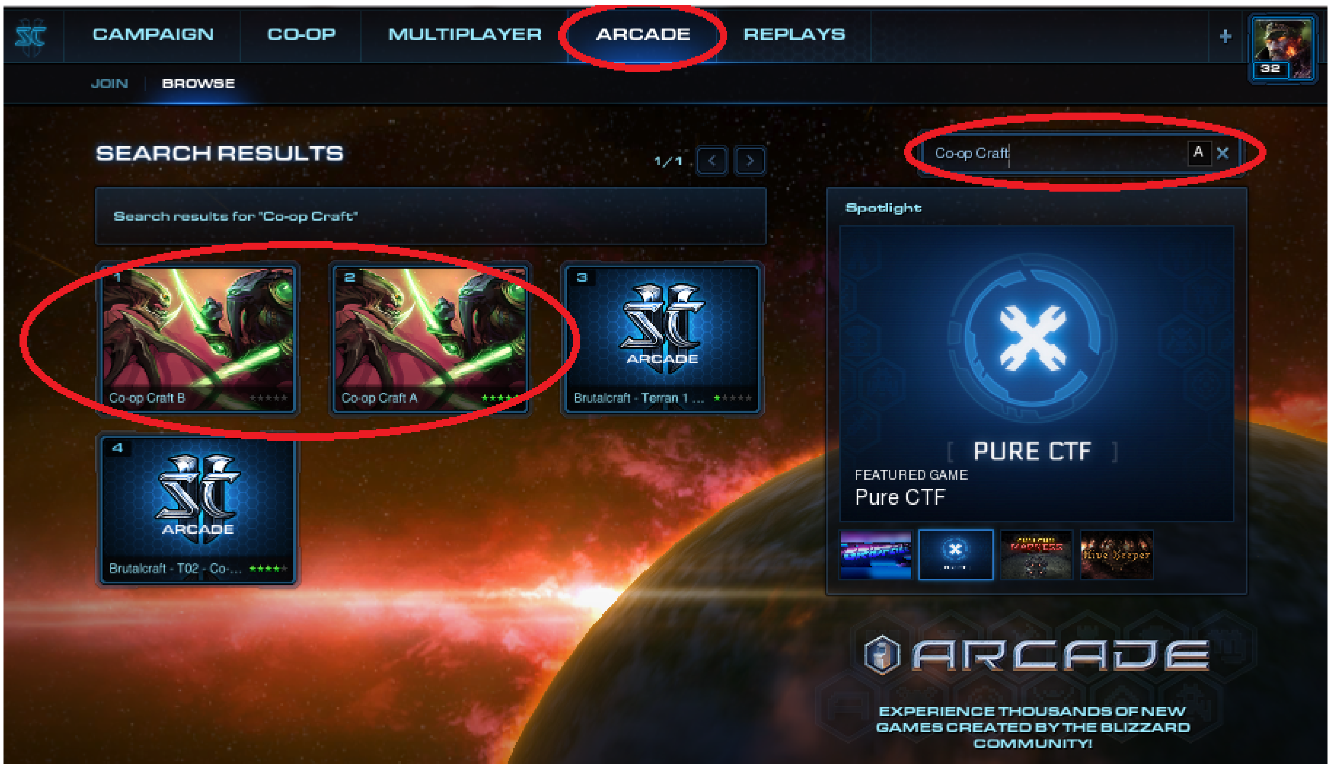

CAMPAIGN
CO-OP
MULTIPLAYER
ARCADE
REPLAYS
JOIN
BROWSE
SEARCH RESULTS
1/1
Co-op Craft
Search results for "Co-op Craft"
Spotlight
Co-op Craft B
Co-op Craft A
ARCADE
Brutalcraft - Terran 1 ...
Brutalcraft - T02 - Co-...
PURE CTF
FEATURED GAME
Pure CTF
ARCADE
EXPERIENCE THOUSANDS OF NEW GAMES CREATED BY THE BLIZZARD COMMUNITY!

Figure 4.18.: Screenshot of the Arcade main screen in StarCraft II. Highlighted in red are the "Arcade" Tab, the search box where one must enter "Co-op Craft" and the actual Co-op Craft game(s). For testing purposes (see Chapter 6) there used to be Game A and Game B, like in this screenshot, but after submitting this report I might have changed it so that there is only one "Co-op Craft" game map.

7. The host clicks on the game and selects "Play as Party". The other player will click "Download". Both players will be taken to the Lobby and the game will start loading automatically. Wait for the game to load and then you will be playing Co-op Craft.
8. Phases of the game: i) Tutorial, ii) Training iii) Intro iv) Actual game. i) and iii) can be skipped, but I do not recommend you to do so the first time. The tutorial, training and info are there to help you get used to controls and gameplay. Read all the text in the game carefully and attentively, maybe it is a lot, but it is there to help you!
9. Good luck!

NOTE: it is recommended to reset the hotkeys to the standard or default set in order for all hotkeys to properly work. This applies to those who have played StarCraft II before and changed their hotkeys according to preferences. It is also recommended that the language you are playing is English, if StarCraft II is installed in another language you should change the language to English (see `https://eu.battle.net/support/en/article/6372`)

5 ADappTowr - an alternative

In this short chapter we wish to present an alternative game to the implemented Co-op Craft, along with DDA possibilities associated to it. This game is called **ADappTowr**. Our objective here is to show how the general conceptions outlined in Chapter 3 can be applied to other game genres and other style of video games. Indeed, ADappTowr is a completely different game from Co-op Craft. The game was intended to be developed with the cross-platform Unity game engine. [1] It was actually my initial idea to test DDA in this alternative game, but later I decided to work on Co-op Craft instead, leaving ADappTowr only in a conceived state. I figured that I would be able to produce a much better game within my time constraints with the StarCraft II Galaxy Editor than with Unity, given my previous experience and taking into account that I was not developing a game with a development team but rather by myself.

5.1 Game concept

ADappTowr (the name is a cross between "Adaptation", "Application" and "Tower") is a **puzzle platformer shooter distinct-role collaborative two-player** game. Both players control a character that can move, jump and shoot. These characters are pitted inside a tower with different platform levels. Their objective is to go up the tower by completing levels, which increase in difficulty as the players ascend the tower. In order to complete a level, heroes must work together and perform different tasks that allow their partner to progress.

Levels must be completed within a certain amount of time. Levels contain moving platforms, doors and switches. A player may press a switch to open a certain door, but the moment he leaves the switch the door closes again. Thus, players take turns in holding a switch and traversing open doors. The puzzle complexity would increase alongside the game difficulty. Holding the switch activated may also imply having to hold position from incoming enemies while the partner goes through the door.

The other important feature of the game is the moving platforms. When one of the players attempts to cross a chasm, he must be careful and aim correctly when jumping (like in other platformer games). Enemy droids hang from the ceiling and walls and shoot projectiles at the player who is jumping. The other player attempts to protect his partner by shooting at the droids from the edge of the chasm in FPS-sniper style. When one of the players gets through he will be the one shooting at the droids to allow his partner to reach him in the other side, so the roles of platforming and FPS are interchanged but never shared simultaneously.

Game elements like ammo, health packages, bonus items could be added. Enemies could be droids in the ceiling/walls and/or zombies that pursue the players. The advantage of using fixed droids is that they require no path-finding algorithms to implement them - they simply shoot toward the location of the human character!

Ideally the levels are not just hard coded, but generated on-the-fly with a random-level generator that uses parameters like "game difficulty", "number of levels completed", "platform types to be used"... Groups of platforms with determined movement behaviors could be stored as prefabs in Unity, thus simplifying the complexity of this generator (it would only need to calculate the placement for groups of platforms and not the placement of every single platform in the level).

[1] http://unity3d.com/unity

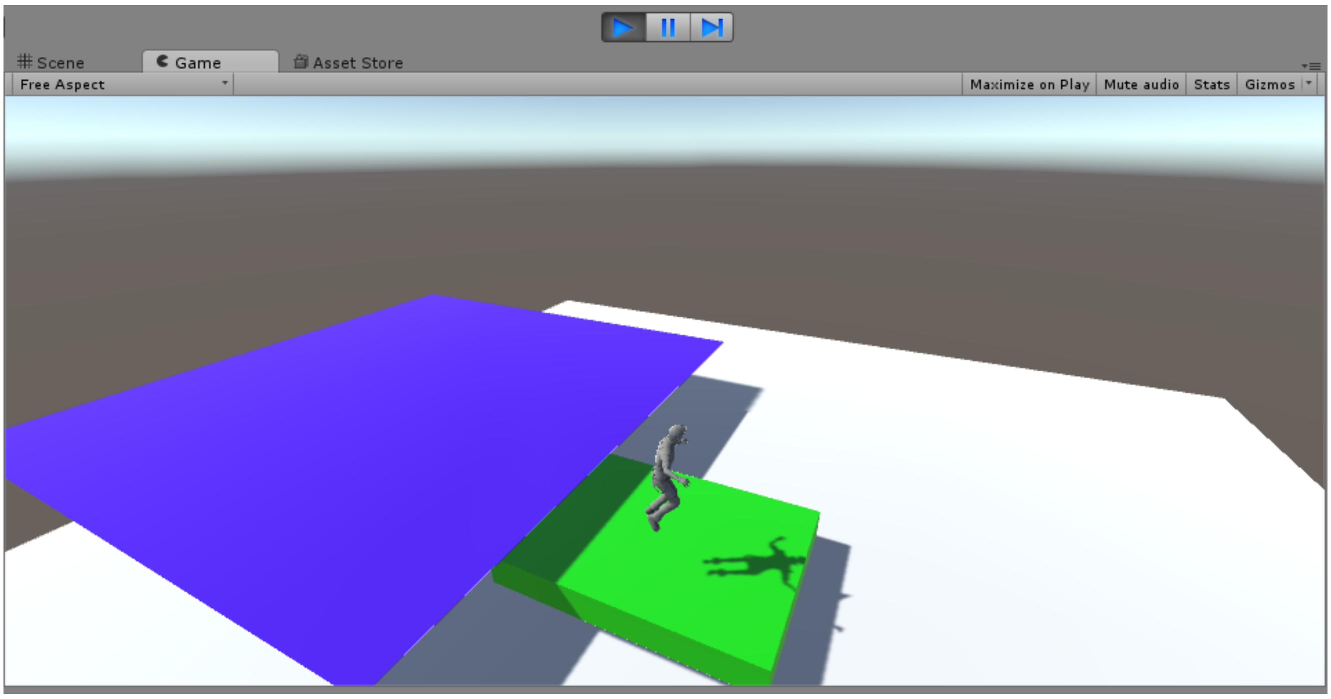


Figure 5.1.: Screenshot of ADappTowr in a very early stage of development. A game character jumping on platforms can be appreciated. This is as far as the author went on implementing ADappTowr before it was decided that Co-op Craft would be the implemented game instead.

5.2 DDA in ADappTowr

Although ADappTowr was not (yet) implemented, several ideas for possible DDAAs were conceived. We will outline some of them. Due to the gameplay, a player will probably perform differently in different tasks (a player may be good at shooting, but maybe not so good at jumping or deciphering puzzles). In Co-op Craft we decided to associate player's performance exclusively with the collected_resources C_R parameter, but here we do consider the different performances for different tasks. Like in Co-op Craft, there are many other alternatives and possibilities apart from the ones conceived here, but here are the ones we had in mind:

- Depending on how well a player shoots, we will adjust the size of the enemies. If the player is not very good at shooting, the size of the enemies will be bigger so the player has a higher chance to hit them. If the player is good then the size would be reduced to make them harder to hit. To determine how well a player shoots, we would analyze the percentage of enemies shot down within the time it takes for the partner to cross the chasm. The size of the enemies will be adjusted the next time this player has the FPS role. This DDAA would thus be classified as an **event-based triggering** (the DDAA activates when the partner starts jumping across the chasm and deactivates when he finishes), **parameter probing** ("percentage of enemies killed" is the significant parameter being probed), **proactive** (it modifies the size of future enemies), **continuous** (the size of an enemy could take any value within a minimum to maximum value range), **auto-corrective** (it aims to modify the significant parameter "percentage of enemies killed"), **standardizer**, **single-player** DDAA.
- The amount of droids that spawn, the speed of their projectiles and the movement patterns of the platforms would be determined by the skills of the player as he jumps the chasm. If the player is hit, killed or falls to his death more often than determined by a standard, then the game tunes down the difficulty by spawning less enemies, slowing down the projectiles and simplifying the platform movements. Every time the player is hit or dies the DDA would be triggered and analyze

how often this is happening. This DDAA would thus be classified as an **event-based triggering** (the DDAA activates when the player is hit or dies), **parameter probing** ("number of times this DDAA has been triggered" is one the significant parameters being probed), **reactive AND proactive** (it modifies the speed of existing and future projectiles, the movement of existing platforms and the spawning of future droids), **continuous** (the speed, number of enemies spawned and movement values are taken within a valid range of values), **auto-corrective** (it aims to modify the significant parameter "number of times this DDAA has been triggered", so that this DDAA is triggered less or more often), **standardizer**, **single-player** DDAA.

- When bonus items like ammo or health packages spawn, the amount that they carry will be modified according to the current health or ammo of both players. This DDAA would thus be classified as an **event-based triggering** (the DDAA activates when a bonus item spawns), **parameter probing** ("health" and "ammo" are the significant parameters being probed), **proactive** (it modifies the amount the bonus item provides an instant before it spawns, just like the Respawning DDAA in Co-op Craft), **continuous** (the bonus amount takes a value in a valid range), **auto-corrective** (it aims to modify the significant parameter "number of times this DDAA has been triggered", so that this DDAA is triggered less or more often), **standardizer**, **multi-player** (it takes into account the health and ammo of BOTH players) DDAA.
- At certain points in time (Countdown timer signals 3 minutes left, 1 minute left, 30 seconds left, etc.) the game observes the progress of the team in the level. If they are ahead, it increases the game difficulty; if they are behind, it decreases it. Tweaked parameters could be movement of platforms, spawning of enemies... The progress can be measured as the percentage of doors traversed to total doors in the level. This DDAA would thus be classified as a **non-periodic timed-based triggering** (the DDAA activates when the countdown timer signals a certain amount of time left), **parameter probing** ("percentage of traversed doors" is the significant parameter being probed), **reactive AND proactive** (it modifies the movement of existing platforms, the spawning of future droids...), **continuous** (platform speed, enemies spawned... take values in a valid range), **auto-corrective** (it aims to modify the significant parameter "percentage of traversed doors"), **standardizer**, **multi-player** (it affects the whole level, i.e. it affects both players) DDAA.

All of the DDAAs above are also classified as **implicit** DDAAs. As with Co-op Craft, the idea was to test teams playing the game both with and without DDA. The disruption of disbelief due to difficulty adjustment was to be avoided. See the comments on page 47 about players not noticing DDA.

6 The Test

With the Co-op Craft game (and the DDA it includes presented), we will now proceed to explain in detail how we tested the DDA's effectiveness. We gathered **twelve players** (six teams) and had them follow the exact same procedure, which we will call "**The Test**". Firstly it should be noted that these twelve players had never played Co-op Craft before the test, they were all different from the beta testers who helped me develop, refine and balance Co-op Craft. These new testers were not to test the gameplay, but the effectiveness of the DDA in Co-op Craft. I aimed to get players from diverse backgrounds, especially gaming backgrounds (regular MOBA gamers, casual gamers, almost no gaming experience...). The testers were from diverse countries too - Spain, Greece and Germany; although the partners did share their country of origin and their mother tongue in order to communicate better. The whole test procedure lasts two hours maximum, which I found to be a reasonable time to spare, taking into account that getting two people to play together is much harder than getting just one person.

6.1 Phase #1: Introductory Video

Due to the relative complexity of the Co-op Craft game for first timers, I decided to make a video in order to prepare and help the players perform better in the game. Although everything is detailed with text in-game, in an RTS game sometimes there is just not enough time to read everything, and of course images can also be more useful than words in many occasions. This 20-minute video can be found in: `https://www.youtube.com/watch?v=_58XZZ7LHkw`. The purpose of the video is to explain basic gameplay and showcase all possible spells for each hero, so that players have a basic idea of what they are buying when spending resources. The purpose was NOT to present the important concepts of this work, but to HELP the players perform better, hence why DDA is not even mentioned. Before playing the game, the two players in the team watched this video and asked me any questions they came up with. All in all it made for a good introduction for what the players would be up to.

6.2 Phase #2: Playing the Co-op Craft game

After seeing the introductory video together, the team played the game, following the steps described in Section 4.5.2. In order to test the effectiveness of the implemented DDA, all teams played the game twice: once with the DDA active and once with the DDA disabled. For this purpose we made two types of game: **Co-op Craft A (DDA active)** and **Co-op Craft B (DDA disabled)**. Players were not notified beforehand of the difference between the game types.

Logic dictates that the second time people play the game their performance is quite higher than the first time (where they have to get to used to the controls and the gameplay). This factor must be taken into account when comparing the two games played by a same team. In order to palliate this in the global results, we had half of the teams play first Game A and then Game B and the other half first Game B and then Game A. Players were also asked to switch heroes in the second game. Although heroes share similar controls, their spells and roles are distinct enough to guarantee a first time experience controlling heroes in both games, helping to keep the comparison between both games as rigorous as possible.

Even though I was present in some of the games played, I interfered as less as possible. The only help I provided was to repeat information given in the introductory video and the tutorials in-game, as well as hinting on basic controls to select the hero, move it and cast a spell for those that needed it (a few players were stuck for a while in the base, not knowing how to jump off the platform, despite this being explained in both the video and the tutorial). At no point did I give suggestions on how to proceed, on what to spend resources, on how to fight better, nor did I warn them of potential dangers or traps in the map. And of course I never even feigned on taking control of their hero, not for a single moment.

Once the games were played, replays were saved and sent for me to analyze. Three major factors were observed in the games: **the evolution of the K factor in time** for both players and for the first 30 minutes: $K_M(t)$ and $K_V(t)$, with $t <= 30min$; the **outcome** (victory/defeat) and the **total elapsed time in the game**: $T[secs]$. These major factors were used to measure the effectiveness of the Resources DDAA and the Milestone DDAA.

$K(t)$ measures the effectiveness of the Resources DDAA (Section 4.4.1). Without DDA $K(t)$ should deviate from the standard performance $K(t) = 1$, reflecting the performance of the different players. It is expected that with active DDA $K(t)$ should get closer to the standard for all players, especially as more time elapses and players have opportunities to "catch up" (if they are below average) or players are forced to "retrogress" (if they are above average). The outcome and T measure the effectiveness of the Milestone DDAA (Section 4.4.3). If there is a change in the outcome, it must make sense: teams winning in B may lose in A because the game is more difficult; whereas teams losing in B may win in A because the game is easier. Whether there is a change or not in the outcome, all games should tend to last a similar amount of time irrelevant of the team's overall skills in Game A. So the standard deviation of T from its average should be smaller in Game A than in Game B. For example, if a team loses quickly in Game B, they might also lose in Game A, but they should last longer there because the game is easier. Similarly, if a team beats Game B quickly, they should take longer to beat Game A because the game will be harder. But if a team beats game B after a long time, they probably will have still found Game B to be difficult (despite the final outcome), so Game A will be easier and they will be able to beat it faster.

Measuring the effectiveness of the Respawning DDAA (Section 4.4.2) proves to be trickier. An idea would be to measure the total time a player spends respawning in a whole game, and it should tend to a standard value in Game A. However we actually did not consider this parameter because the Respawning DDAA is not as relevant as the other two DDAAs in modifying player's performance - its main purpose was to avoid frustration in weaker players who die often and have to wait forever to respawn.

Apart from the three major factors taken into account, we also observed some minor factors: the player's **death count** (total times the player dies in the game), the player's **APM** ("Actions Per Minute") and the player's **kill count** (amount of units killed by the player). As explained in Section 4.4.4, these minor factors are easy to see in StarCraft II replays and are used as support for our assertions on a player's performance rather than as the ultimate measuring yardstick. Refer to the aforementioned section for the reasons as to why we do not find them completely trustworthy of a player's performance.

6.3 Phase #3: Questionnaire

Once a team had played Game A and Game B they still had a final task in "The Test": fill out a **questionnaire**. The purpose of the questionnaire is to get an insight in the **psychological impact of the DDA** in the players, as well as asking an honest opinion about DDA on several people with different gaming backgrounds. We will now outline all questions asked (text in italics) and comment the motives of asking such questions.

- *Co-op Craft is supposed to be a collaborative and cooperative game. Would you agree that it is? Why (not)?*
- *Did you feel part of a team with your partner? Were you able to work well together? Why (not)?*

These first two questions are to reassure the good design of Co-op Craft as a distinct-role co-op game. We expect players to answer positively to both questions and give good reasons for their answer. If most or many of the players answer negatively with reasonable arguments, the game is probably not well designed as a distinct-role co-op game, thus we would not be able to take the results concerning DDA in this game as representative for DDA in co-op games.

- *As you know there were two game types: A and B. Could you notice a difference between the two game types? If so, what do you think was the difference?*

This is an interesting question. In this work we have not delved deeply into the psychological impact of DDA in the implemented game, yet here we make an important exception. As the reader might recall, all of the implemented DDAAs in Co-op Craft are classified as implicit DDAAs (see Section 4.4). As explained in Section 3.3, implicit DDAAs try to make changes in the game without making them too obvious to the player and disrupting his sense of disbelief. This is normally desirable, as pointed out by Adams [1] (see also Section 2.2.2). Hence when questioned about the difference between Game A and Game B, we expect the players to NOT notice that there was DDA going on in Game A. Then the assertion that DDA is implicit in Co-op Craft would be correct. Note that the players may still notice an apparent difference in the difficulty of the games (for some Game A might be easier, for some it might be more difficult), but even a sharp observer should not notice that the game is adapting according to player's performance. The real difference between Games A and B is explained in the next question (players were asked to fill the previous questions before reading and answering the final ones).

This is the actual difference between the two types of games: Type A included a Dynamic Difficulty Algorithm (DDA), whereas Type B didn't. In other words, in Type A the game analyzed both your performance and your partner's performance and adapted the game difficulty accordingly. For example, if you were behind in collected resources, the volcano eruptions would happen more often and the volcano resources would last longer, in order for you to catch up. On the other hand if you were ahead, volcano eruptions would happen less often and the resources would last less. This is one example among others that Game Type A implemented. Game Type B had the standard difficulty and did not adapt at all according to player performance.

- *What are your impressions on this? Do you think it was a good idea to adapt the difficulty of the game according to you and your partner's performance? Was it done well in this game in your opinion? Would you like to see this kind of adjustments in other video games? Why (not)?*

These are the most important questions. Positive answers and opinions here, along with good reasons from the player's perspective, would help us conclude that DDA has great potential in co-op games and that there are good motives to believe that it would be appreciated and well-received if implemented commercially.

- *Do you have any other comments or suggestions? For example: how could Co-op Craft be improved, what did you miss in this game, what did you like about the game, why would you (not) play it again, was it too easy/frustrating, why will you not consider me your friend after this... And anything else miscellaneous you would like to comment about. ;)*

This is obviously the least important question of the questionnaire for the purposes of this work (the focus of this work is not on Co-op Craft as a game). But here I can get some nice feedback in case I want to improve the game in the future.

7 Results

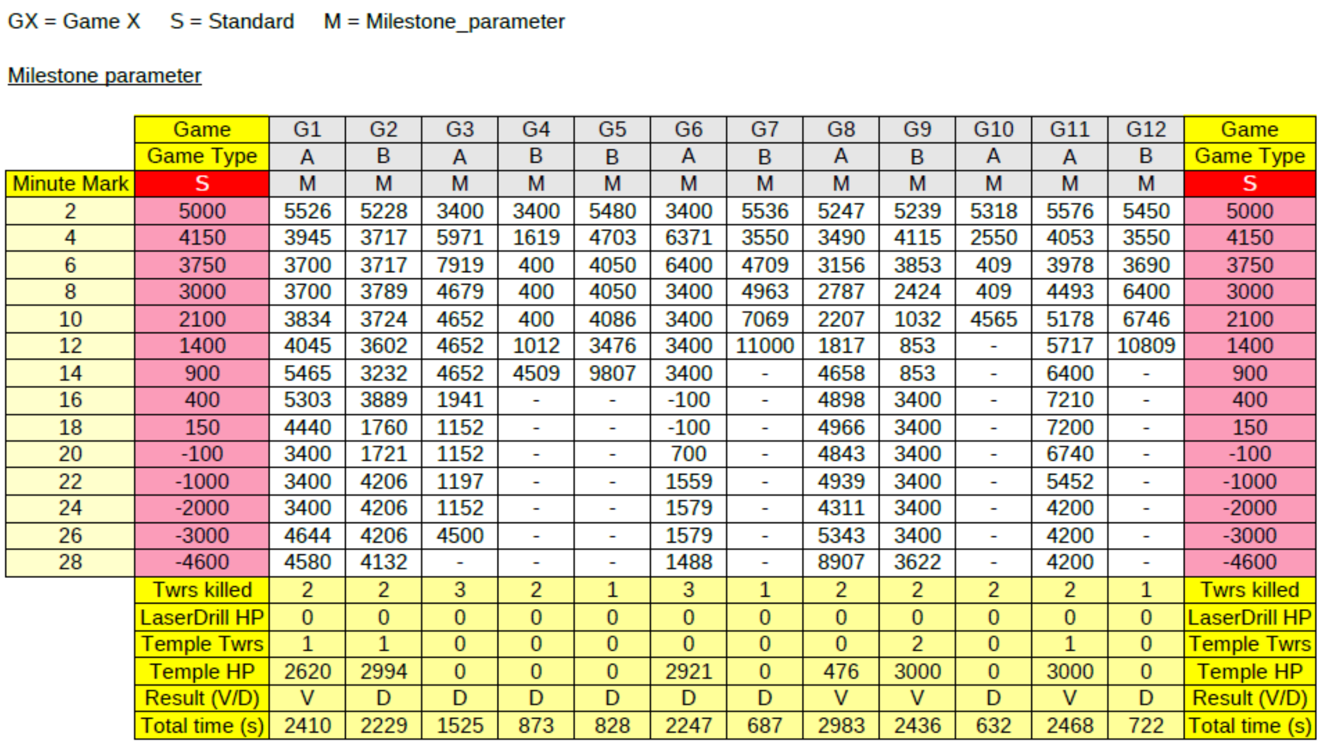
Here we will present the results of "The Test", both the raw data and the analysis to this data. We will also evaluate the performance of all players and teams with support from the numerical data and my personal appreciation from viewing the replays. Following this we will analyze the answers to the questionnaire. Finally we will expose the encountered problems and suggest possible solutions to them.

7.1 Numerical Data

The data in the tables should be self-explanatory. The graphs however convey the relevant data more clearly and allow to extract conclusions more easily. Games 1 and 2 were played by Team 1, Games 3 and 4 by Team 2, etc. Team 1 consisted of players 1 and 2, Team 2 of players 3 and 4, etc. In mathematical terms:

$$T_i \text{ plays } G_{2i-1} \text{ and } G_{2i};\ T_i \equiv (P_{2i-1}, P_{2i}); i = 1, ..., 6$$

7.1.1 Tables

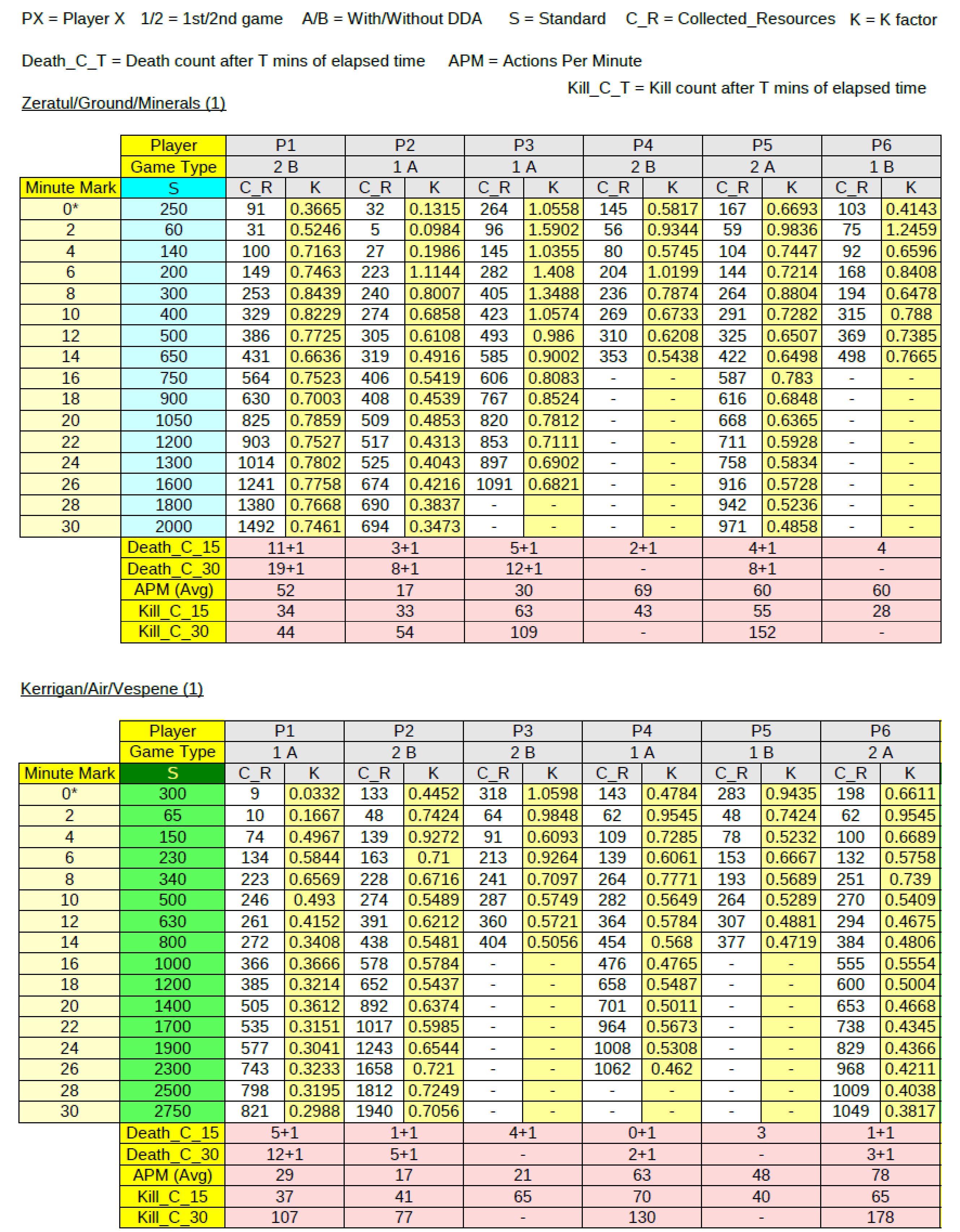


GX = Game X S = Standard M = Milestone_parameter

Milestone parameter

	Game	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	Game
	Game Type	A	B	A	B	B	A	B	A	B	A	A	B	Game Type
Minute Mark	S	M	M	M	M	M	M	M	M	M	M	M	M	S
2	5000	5526	5228	3400	3400	5480	3400	5536	5247	5239	5318	5576	5450	5000
4	4150	3945	3717	5971	1619	4703	6371	3550	3490	4115	2550	4053	3550	4150
6	3750	3700	3717	7919	400	4050	6400	4709	3156	3853	409	3978	3690	3750
8	3000	3700	3789	4679	400	4050	3400	4963	2787	2424	409	4493	6400	3000
10	2100	3834	3724	4652	400	4086	3400	7069	2207	1032	4565	5178	6746	2100
12	1400	4045	3602	4652	1012	3476	3400	11000	1817	853	-	5717	10809	1400
14	900	5465	3232	4652	4509	9807	3400	-	4658	853	-	6400	-	900
16	400	5303	3889	1941	-	-	-100	-	4898	3400	-	7210	-	400
18	150	4440	1760	1152	-	-	-100	-	4966	3400	-	7200	-	150
20	-100	3400	1721	1152	-	-	700	-	4843	3400	-	6740	-	-100
22	-1000	3400	4206	1197	-	-	1559	-	4939	3400	-	5452	-	-1000
24	-2000	3400	4206	1152	-	-	1579	-	4311	3400	-	4200	-	-2000
26	-3000	4644	4206	4500	-	-	1579	-	5343	3400	-	4200	-	-3000
28	-4600	4580	4132	-	-	-	1488	-	8907	3622	-	4200	-	-4600
	Twrs killed	2	2	3	2	1	3	1	2	2	2	2	1	Twrs killed
	LaserDrill HP	0	0	0	0	0	0	0	0	0	0	0	0	LaserDrill HP
	Temple Twrs	1	1	0	0	0	0	0	0	2	0	1	0	Temple Twrs
	Temple HP	2620	2994	0	0	0	2921	0	476	3000	0	3000	0	Temple HP
	Result (V/D)	V	D	D	D	D	D	D	V	V	D	V	D	Result (V/D)
	Total time (s)	2410	2229	1525	873	828	2247	687	2983	2436	632	2468	722	Total time (s)

Figure 7.1.: Milestone parameter with time $M(t)$ for $t \leq 30min$, all 12 games included. Additional parameters common for the team like Enemy Towers killed and the game outcome (V=Victory/D=Defeat) can also be seen. These additional parameters were registered at $t = 30min$ (except the Game Outcome and the total elapsed time T, registered obviously at the end of the game).



PX = Player X   1/2 = 1st/2nd game   A/B = With/Without DDA   S = Standard   C_R = Collected_Resources   K = K factor

Death_C_T = Death count after T mins of elapsed time   APM = Actions Per Minute   Kill_C_T = Kill count after T mins of elapsed time

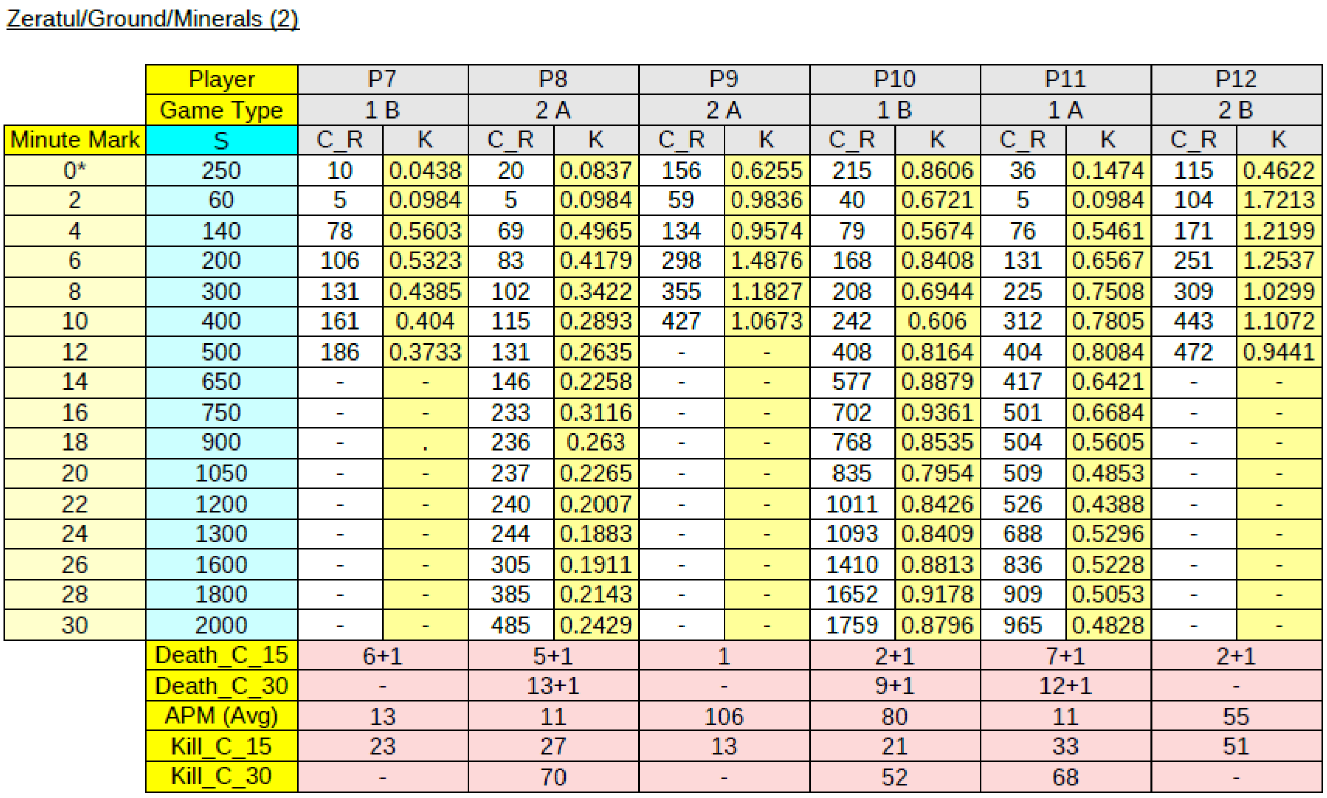
Zeratul/Ground/Minerals (1)

	Player	P1		P2		P3		P4		P5		P6	
	Game Type	2 B		1 A		1 A		2 B		2 A		1 B	
Minute Mark	S	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K
0*	250	91	0.3665	32	0.1315	264	1.0558	145	0.5817	167	0.6693	103	0.4143
2	60	31	0.5246	5	0.0984	96	1.5902	56	0.9344	59	0.9836	75	1.2459
4	140	100	0.7163	27	0.1986	145	1.0355	80	0.5745	104	0.7447	92	0.6596
6	200	149	0.7463	223	1.1144	282	1.408	204	1.0199	144	0.7214	168	0.8408
8	300	253	0.8439	240	0.8007	405	1.3488	236	0.7874	264	0.8804	194	0.6478
10	400	329	0.8229	274	0.6858	423	1.0574	269	0.6733	291	0.7282	315	0.788
12	500	386	0.7725	305	0.6108	493	0.986	310	0.6208	325	0.6507	369	0.7385
14	650	431	0.6636	319	0.4916	585	0.9002	353	0.5438	422	0.6498	498	0.7665
16	750	564	0.7523	406	0.5419	606	0.8083	-	-	587	0.783	-	-
18	900	630	0.7003	408	0.4539	767	0.8524	-	-	616	0.6848	-	-
20	1050	825	0.7859	509	0.4853	820	0.7812	-	-	668	0.6365	-	-
22	1200	903	0.7527	517	0.4313	853	0.7111	-	-	711	0.5928	-	-
24	1300	1014	0.7802	525	0.4043	897	0.6902	-	-	758	0.5834	-	-
26	1600	1241	0.7758	674	0.4216	1091	0.6821	-	-	916	0.5728	-	-
28	1800	1380	0.7668	690	0.3837	-	-	-	-	942	0.5236	-	-
30	2000	1492	0.7461	694	0.3473	-	-	-	-	971	0.4858	-	-
	Death_C_15	11+1		3+1		5+1		2+1		4+1		4	
	Death_C_30	19+1		8+1		12+1		-		8+1		-	
	APM (Avg)	52		17		30		69		60		60	
	Kill_C_15	34		33		63		43		55		28	
	Kill_C_30	44		54		109		-		152		-	

Kerrigan/Air/Vespene (1)

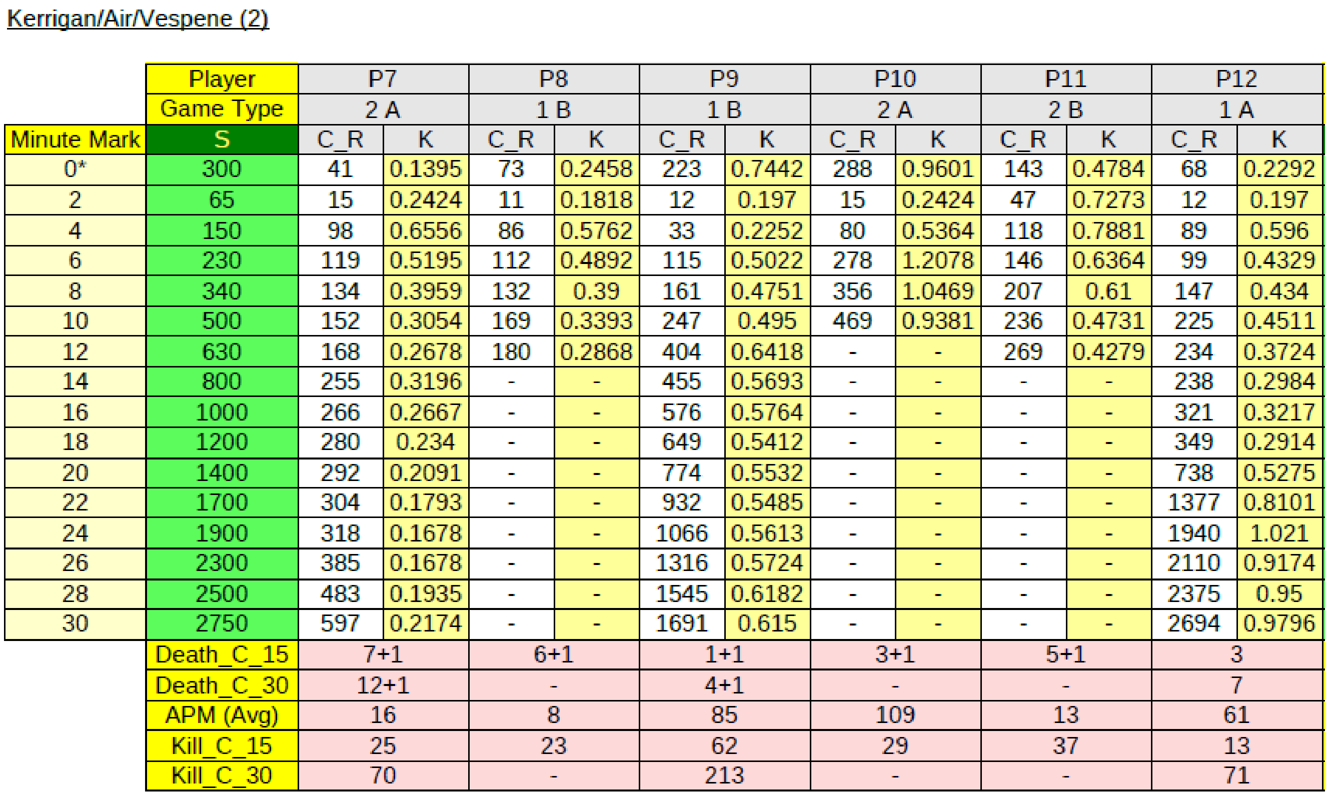
	Player	P1		P2		P3		P4		P5		P6	
	Game Type	1 A		2 B		2 B		1 A		1 B		2 A	
Minute Mark	S	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K
0*	300	9	0.0332	133	0.4452	318	1.0598	143	0.4784	283	0.9435	198	0.6611
2	65	10	0.1667	48	0.7424	64	0.9848	62	0.9545	48	0.7424	62	0.9545
4	150	74	0.4967	139	0.9272	91	0.6093	109	0.7285	78	0.5232	100	0.6689
6	230	134	0.5844	163	0.71	213	0.9264	139	0.6061	153	0.6667	132	0.5758
8	340	223	0.6569	228	0.6716	241	0.7097	264	0.7771	193	0.5689	251	0.739
10	500	246	0.493	274	0.5489	287	0.5749	282	0.5649	264	0.5289	270	0.5409
12	630	261	0.4152	391	0.6212	360	0.5721	364	0.5784	307	0.4881	294	0.4675
14	800	272	0.3408	438	0.5481	404	0.5056	454	0.568	377	0.4719	384	0.4806
16	1000	366	0.3666	578	0.5784	-	-	476	0.4765	-	-	555	0.5554
18	1200	385	0.3214	652	0.5437	-	-	658	0.5487	-	-	600	0.5004
20	1400	505	0.3612	892	0.6374	-	-	701	0.5011	-	-	653	0.4668
22	1700	535	0.3151	1017	0.5985	-	-	964	0.5673	-	-	738	0.4345
24	1900	577	0.3041	1243	0.6544	-	-	1008	0.5308	-	-	829	0.4366
26	2300	743	0.3233	1658	0.721	-	-	1062	0.462	-	-	968	0.4211
28	2500	798	0.3195	1812	0.7249	-	-	-	-	-	-	1009	0.4038
30	2750	821	0.2988	1940	0.7056	-	-	-	-	-	-	1049	0.3817
	Death_C_15	5+1		1+1		4+1		0+1		3		1+1	
	Death_C_30	12+1		5+1		-		2+1		-		3+1	
	APM (Avg)	29		17		21		63		48		78	
	Kill_C_15	37		41		65		70		40		65	
	Kill_C_30	107		77		-		130		-		178	

Figure 7.2.: Collected Resources C_R and K factor for players 1-6. Additional minor parameters Death_Count, APM and Kill_Count also are included. NOTE: "X+1" in Death_Count means that the player died X times in the actual game and once in the tutorial. Absence of "+1" means he did not die in the tutorial.



Zeratul/Ground/Minerals (2)

	Player	P7		P8		P9		P10		P11		P12	
	Game Type	1 B		2 A		2 A		1 B		1 A		2 B	
Minute Mark	S	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K
0*	250	10	0.0438	20	0.0837	156	0.6255	215	0.8606	36	0.1474	115	0.4622
2	60	5	0.0984	5	0.0984	59	0.9836	40	0.6721	5	0.0984	104	1.7213
4	140	78	0.5603	69	0.4965	134	0.9574	79	0.5674	76	0.5461	171	1.2199
6	200	106	0.5323	83	0.4179	298	1.4876	168	0.8408	131	0.6567	251	1.2537
8	300	131	0.4385	102	0.3422	355	1.1827	208	0.6944	225	0.7508	309	1.0299
10	400	161	0.404	115	0.2893	427	1.0673	242	0.606	312	0.7805	443	1.1072
12	500	186	0.3733	131	0.2635	-	-	408	0.8164	404	0.8084	472	0.9441
14	650	-	-	146	0.2258	-	-	577	0.8879	417	0.6421	-	-
16	750	-	-	233	0.3116	-	-	702	0.9361	501	0.6684	-	-
18	900	-	.	236	0.263	-	-	768	0.8535	504	0.5605	-	-
20	1050	-	-	237	0.2265	-	-	835	0.7954	509	0.4853	-	-
22	1200	-	-	240	0.2007	-	-	1011	0.8426	526	0.4388	-	-
24	1300	-	-	244	0.1883	-	-	1093	0.8409	688	0.5296	-	-
26	1600	-	-	305	0.1911	-	-	1410	0.8813	836	0.5228	-	-
28	1800	-	-	385	0.2143	-	-	1652	0.9178	909	0.5053	-	-
30	2000	-	-	485	0.2429	-	-	1759	0.8796	965	0.4828	-	-
	Death_C_15	6+1		5+1		1		2+1		7+1		2+1	
	Death_C_30	-		13+1		-		9+1		12+1		-	
	APM (Avg)	13		11		106		80		11		55	
	Kill_C_15	23		27		13		21		33		51	
	Kill_C_30	-		70		-		52		68		-	



Kerrigan/Air/Vespene (2)

	Player	P7		P8		P9		P10		P11		P12	
	Game Type	2 A		1 B		1 B		2 A		2 B		1 A	
Minute Mark	S	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K	C_R	K
0*	300	41	0.1395	73	0.2458	223	0.7442	288	0.9601	143	0.4784	68	0.2292
2	65	15	0.2424	11	0.1818	12	0.197	15	0.2424	47	0.7273	12	0.197
4	150	98	0.6556	86	0.5762	33	0.2252	80	0.5364	118	0.7881	89	0.596
6	230	119	0.5195	112	0.4892	115	0.5022	278	1.2078	146	0.6364	99	0.4329
8	340	134	0.3959	132	0.39	161	0.4751	356	1.0469	207	0.61	147	0.434
10	500	152	0.3054	169	0.3393	247	0.495	469	0.9381	236	0.4731	225	0.4511
12	630	168	0.2678	180	0.2868	404	0.6418	-	-	269	0.4279	234	0.3724
14	800	255	0.3196	-	-	455	0.5693	-	-	-	-	238	0.2984
16	1000	266	0.2667	-	-	576	0.5764	-	-	-	-	321	0.3217
18	1200	280	0.234	-	-	649	0.5412	-	-	-	-	349	0.2914
20	1400	292	0.2091	-	-	774	0.5532	-	-	-	-	738	0.5275
22	1700	304	0.1793	-	-	932	0.5485	-	-	-	-	1377	0.8101
24	1900	318	0.1678	-	-	1066	0.5613	-	-	-	-	1940	1.021
26	2300	385	0.1678	-	-	1316	0.5724	-	-	-	-	2110	0.9174
28	2500	483	0.1935	-	-	1545	0.6182	-	-	-	-	2375	0.95
30	2750	597	0.2174	-	-	1691	0.615	-	-	-	-	2694	0.9796
	Death_C_15	7+1		6+1		1+1		3+1		5+1		3	
	Death_C_30	12+1		-		4+1		-		-		7	
	APM (Avg)	16		8		85		109		13		61	
	Kill_C_15	25		23		62		29		37		13	
	Kill_C_30	70		-		213		-		-		71	

Figure 7.3.: Collected Resources C_R and K factor for players 7-12. Additional minor parameters Death_Count, APM and Kill_Count also are included. NOTE: "X+1" in Death_Count means that the player died X times in the actual game and once in the tutorial. Absence of "+1" means he did not die in the tutorial.

7.1.2 Graphs

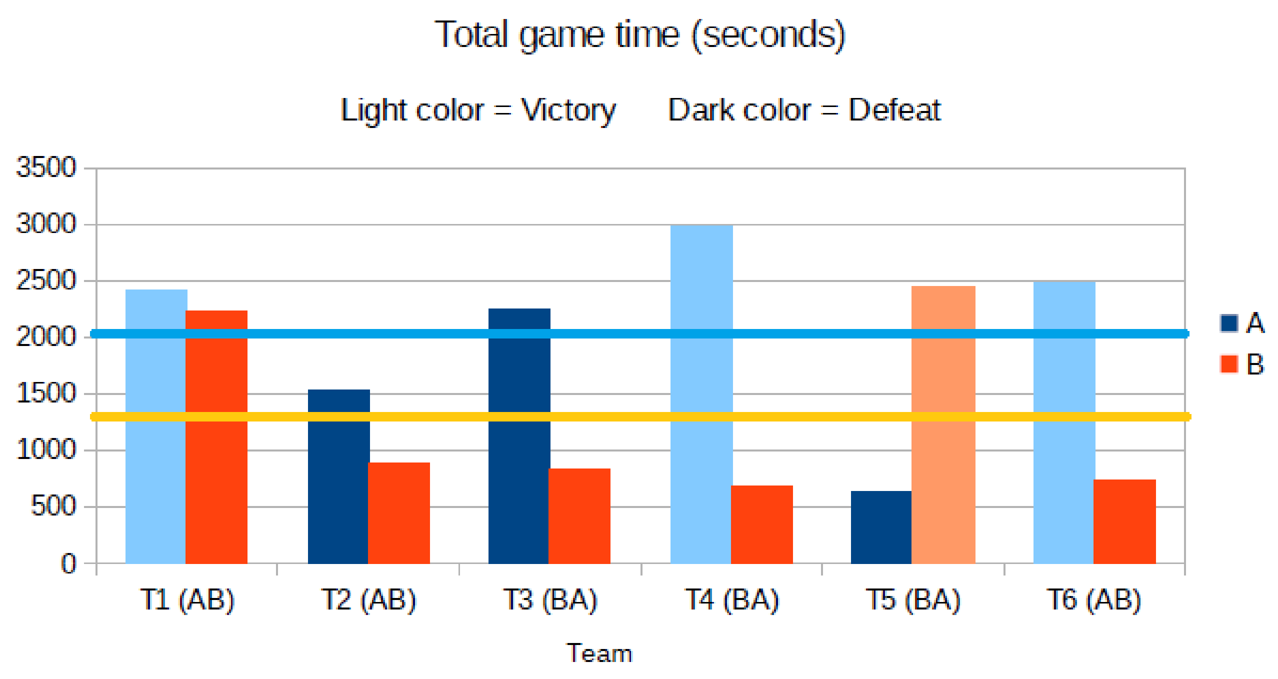


Figure 7.4.: Comparison of total elapsed time $T[secs]$ between games A **(DDA active)** and B **(DDA disabled)** for each team. Outcome (Victory/Defeat) is also shown. AB = Game A was played first, then B. BA = Game B was played first, then A. The highlighted horizontal lines are the average of T in game A ($T_A = 2044.2secs$) and the average of T in game B ($T_B = 1295.8secs$)

As explained in Section 6.2, T and game outcome are the necessary data to analyze the effectiveness of the Milestone DDAA (see Section 4.4.3 for an explanation of this DDAA). The graph in Figure 7.4 precisely contains this data. We can observe the following:

- The standard static difficulty in Co-op Craft (Game B) is generally too difficult for first timers. Most of the teams lost before the 15 minute mark (900 seconds). Team 1 lasted quite longer but still lost (consider also that Game B was their second game). Only Team 5 stands out from the crowd by beating Game B (being their first game too!). They were clearly the best team in performance as we will see later, yet still they needed over 40 minutes to beat Game B.
- The outcomes make sense: Teams 1, 4 and 6 were able to beat Game A when in Game B they struggled; Teams 2 and 3 still lost in Game A but they lasted quite longer than in Game B; Team 5 had such a high performance that Game A adapted to be very difficult for them and they lost quickly.
- Except for Team 5, all teams lasted longer (some much longer) in Game A, which makes the average T be quite bigger in Game A than in Game B. Taking into account that the standard total game time in Co-op Craft was aimed to be between 30 and 40 minutes (1800 to 2400 seconds), the Milestone DDAA has brought players' performance much closer to this standard.
- The standard deviation from the average has more or less stayed the same for both games. Calculating from data in Figure 7.1: $dev_A = 836.63secs$ and $dev_B = 808.5secs$.

The Milestone DDAA influenced the game difficulty generally as we expected (see Section 6.2), thus we can consider the concept implementation of this DDAA a success. Some refined balancing could still see some use. For example, the standard static difficulty of Game B should be lowered (thus we would see a bigger disparity of results than mostly losses at the 15 minute mark). Game A should maybe tune its difficulty in a smoother way, not to the point of making it impossible for good players (like with Team 5). Had Team 5 lasted longer in Game A, the standard deviation for Game A would be much lower. We will now see the effectiveness of the Resources DDAA in the graphs in Figures 7.5 and 7.6.

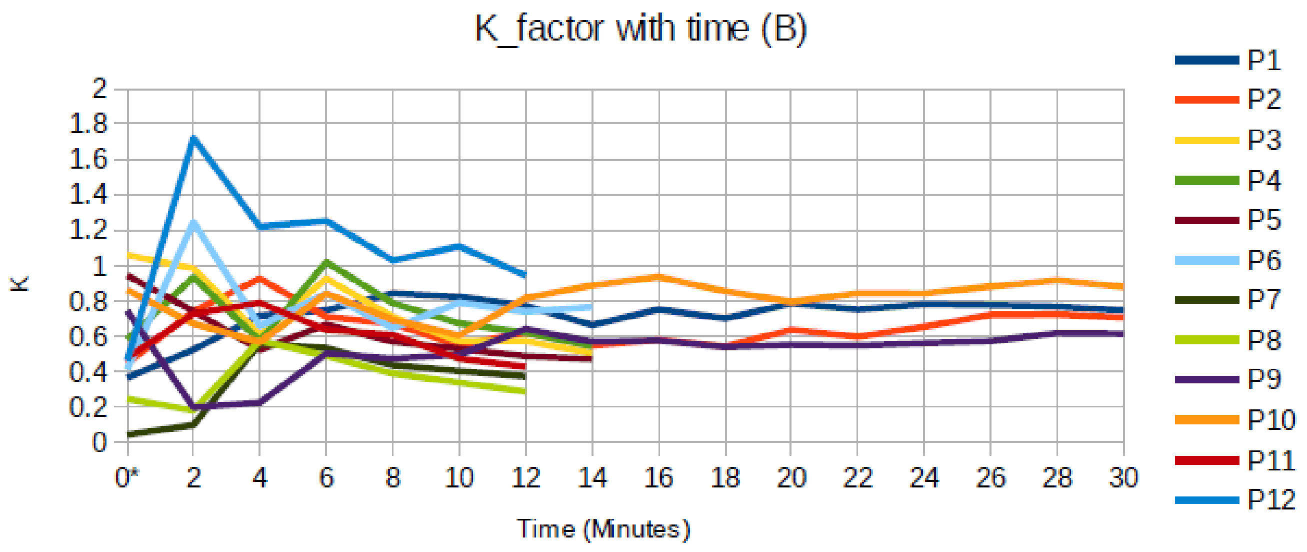


Figure 7.5.: Evolution of K over time (Game Type B - **DDA disabled**) for the first 30 minutes of the game.

Figure 7.5 reflects the real performance and ability of all players, as DDA is not enabled in Game B. As the reader may recall, the constant $K = 1$ is the standard. All players except P12 fall below the standard, although some more than others. As observed, K stabilizes with time (the game can better determine the player's abilities the longer the game has been going on, this is to be expected). The K factor is obviously not measured after the game ends, so we can also appreciate that most teams have lost by the 15 minute mark because their lines are interrupted before then. Seeing this first graph, we would now expect teams to last longer (their graph lines to be prolonged) and also their performance to come closer to the standard $K = 1$ in Game A. Was it like that though? We examine now the performance in Game A.

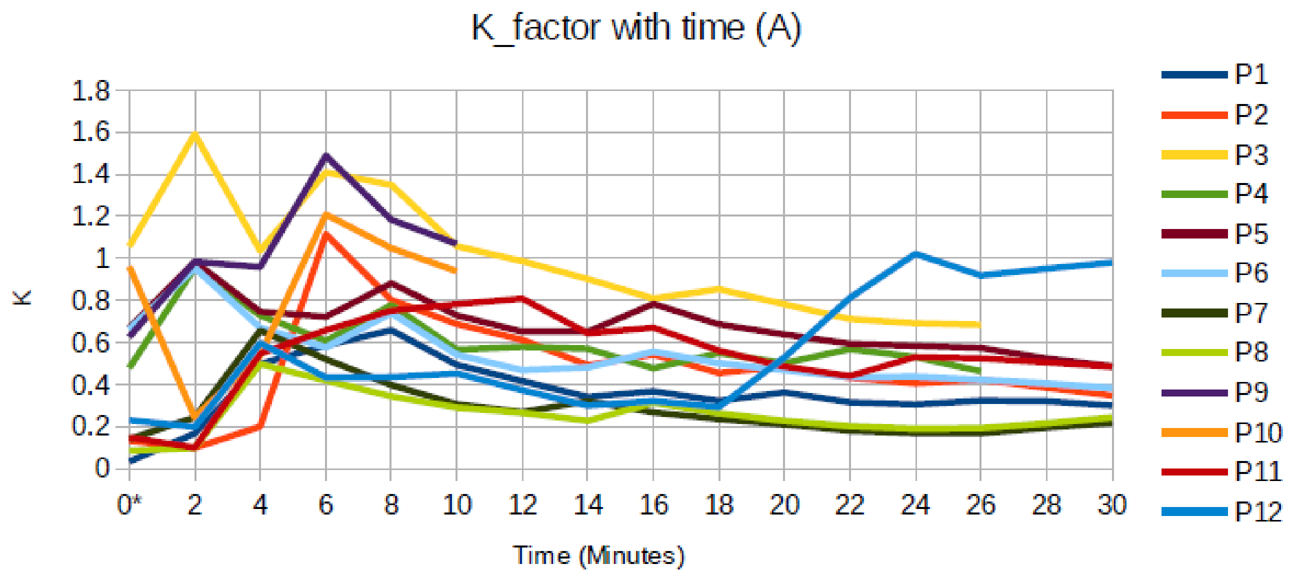


Figure 7.6.: Evolution of K over time (Game Type A - **DDA active**) for the first 30 minutes of the game.

We can see the performance of the players with DDA enabled in Figure 7.6. Players have indeed overall lasted much longer. However, instead of getting closer to the standard $K = 1$, they have stayed mostly on the same K - some even have lower K! This last result was totally unexpected. What happened? It turns out that most of the players barely visited volcanoes during the game. This is an important feature of the game and the players did not behave as expected, which hampered the effectiveness of the Resources DDAA. We will explore this problem further in Section 7.4. There is one exception where the Resources DDAA did get a player's performance closer to the standard. By simple inspection we observe that it is P12. Indeed, P12 was the only player who regularly visited volcanoes to get extra resources, thus producing the expected results. P9 and P10 also visited volcanoes regularly, but they lost by the 10 minute mark in Game A and their progression towards the standard can not be appreciated over time as well as P12's progression.

7.2 Analysis of Player and Team Performance

We will now evaluate the performance of every player and team with the support of the major factors, the minor factors and my appreciation as I watched the game replays.

- **Team 1:** Players 1 and 2 were casual gamers, P2 does not play video games too often, P1 plays other genre of games and dislikes RTS. P1 struggled quite a lot in the first game, especially with controls, but her performance improved quite a lot in the second game, almost doubling her APM and her K closer to the standard. Her death count was high in both games. P2's performance was similar in both games, and quite standard in comparison with other players. Because of P1's increase in performance in the second game, Team 1 were able to last much longer in Game B than they probably would have if they played Game B first.
- **Team 2:** Players 3 and 4 were regular gamers with decent experience in MOBAs. One can see that especially in P4, who has a high APM and a very low Death Count. At the end of their first game P4 had however accumulated over 500 resources and had forgotten to spend them. Had he spent those resources, they would probably have been victorious in Game A.
- **Team 3:** Players 5 and 6 were regular MOBA gamers, more than Team 2. Their APM and Kill Count are both high. Just as Team 2 they were unlucky in Game A and took some wrong decisions when they were close to winning.
- **Team 4:** Players 7 and 8 were very casual gamers, with very little gaming experience. One can observe their very low APM, their high death count and their low kill count. They also struggled a lot with controls in both games. They were of course wiped out quickly in Game B, but in Game A their poor performance made the game very, very easy, and with a lot of patience they were able to beat it.
- **Team 5:** Players 9 and 10 were regular gamers, and they also had quite a lot of experience in StarCraft II. Their APM is very high, they seldom died and they killed a lot (especially P9 in the first game, who ended up with 213 kills at the 30 minute mark!). It is no surprise then that they were able to beat Game B, which has been seemingly too difficult for others. Their performance was so good, however, that in Game A the difficulty was turned to almost impossible, and they lost quickly. As a side note, after knowing the "secret" to the DDA in Game A, they played Game A again and were victorious at the 30 minute mark. This result is of course not recorded in our work.
- **Team 6:** Team 6 is probably the most significant group for the purposes of this work. P11 is a casual gamer, whereas P12 is a regular game with some experience in StarCraft II. This has been the only team where true heterogeneously skilled people have played together. We can observe this in the $K(t)$ graphs, where P12's performance is quite higher than P11's, as well as in the minor factors (P12's APM is much higher than P11's, and he dies much less often). Additionally, P12 was one of the rare players who visited volcanoes, fully receiving the intended effects of the Resources DDAA.

7.3 Questionnaire analysis

We will now lay out and analyze the general answers to the questionnaire written by the players.

- *Co-op Craft is supposed to be a collaborative and cooperative game. Would you agree that it is? Why (not)?*

Everyone agreed here that Co-op Craft is a collaborative game, and most pointed out the distinct-role feature: how heroes complement each other with their abilities, the type of units they can attack and the resources they can pick up. Some mentioned that when they communicated they performed better.

- *Did you feel part of a team with your partner? Were you able to work well together? Why (not)?*

Everyone answered positively to feeling part of a team, some repeating the reasons given in the previous question. Most also felt that they could work well together, although some admitted struggling to cooperate in the beginning when they were still getting used to the game. Most importantly would be the answers from Team 6, who were the heterogeneously skilled team. Both said that in the beginning it was hard to cooperate because they did not fully understand what was going on, yet as their understanding grew their cooperation also grew. Thus it seems that the difference in skills did not hamper teamwork significantly.

The unanimous positive opinion from the players to these two first questions is very well received. It means that Co-op Craft has been successfully implemented as a distinct-role co-op game and that the results obtained about its DDA can be taken seriously as relevant data for DDA in co-op games in general.

- *As you know there were two game types: A and B. Could you notice a difference between the two game types? If so, what do you think was the difference?*

One third of the players thought there was no difference between the two games. Half of them mentioned a big difference in difficulty: for most of them Game B was much more difficult, although Team 5 said Game A was much more difficult. A few of them did notice that the resources in the volcanoes spawned differently and more often depending on the Game Type, yet they did not notice a pattern in the eruptions. The disparity in noticing difficulty was to be expected, as Game A will be made easier or more difficult depending on the team's performance. But nobody appreciated the game adapting the difficulty according to their actions, thus the DDA in Co-op Craft has remained successfully implicit.

This is the actual difference between the two types of games:... (See Section 6.3 for the full text to this question)

- *What are your impressions on this? Do you think it was a good idea to adapt the difficulty of the game according to you and your partner's performance? Was it done well here in your opinion? Would you like to see this kind of adjustments in other video games? Why (not)?*

Perhaps the most important opinions and answers are to be found to this question. Most of the players found the idea of DDA very interesting and appreciated it to be implemented in Game A. Especially players who were more casual and struggled with game controls appreciated that the game slowed down and adapted to their rhythm, for them it was much more fun than having bigger waves of enemies hitting them relatively soon. Those with positive opinions also said that they would definitely like to see DDA implemented in commercial co-op games because it would allow them to have fun with any partner, whether they are skilled or not.

On the other hand, players in Team 5 had a very contrasting opinion. Player 9 literally wrote:

"I don't like that idea because it makes the game unbalanced, especially when it comes to comparison between differently skilled players. I would not like to see DDA, at least in the video games I play because it kinda takes away the feeling of accomplishment, since the game is adjusting when you think you are doing good."

Player 10 thought that players should have an equal chance at collecting resources in this specific game (i.e. he did not agree with the Resources DDAA implementation). But in contrast to his partner Player 9, who flatly said no to DDA, Player 10 said that it depends on the game whether DDA should be implemented or not.

Of course we must consider Team 5's opinions as important of their own right, since Team 5 was after all the only team that was good enough to beat Game B. DDA would definitely disrupt the balance in the competitive scene of gaming and I personally do not think it should be implemented when humans compete against each other, as it does indeed impose a handicap on higher skilled players and makes the game unfair. Yet all casual players find DDA a desirable factor in games, especially when beginning to play and the learning curve is steep. DDA has the potential to make difficult games more fun according to these people, thus the game becomes more commercial and more successful. Additionally, stronger players may agree to have a handicap when they face more inexperienced players in unranked or non-competitive play, so DDA would fit perfectly there. Someone did mention as an answer to this question that it would be interesting to have DDA in a co-op game that he plays because that would let him play alongside professional players without feeling out of place. We can then conclude that DDA in co-op games would be a welcome addition for casual or beginner players who are learning a new complex game, as well as being a potentially fun mode that could effectively balance a game where heterogeneously skilled people play together.

- *Do you have any other comments or suggestions?... (See Section 6.3 for the full text to this question)*

Several answers were given here that are irrelevant for this work. Many people basically suggested improvements on the game (more hero choices, a better UI, more items or weapons...). Half of them did say they had fun playing the game and that they would play again, which is always nice to hear as a developer who had to program this game with certain time constraints. Nobody said they disliked the game, although some did say that one of the Game Types was frustrating. Obviously not everyone will like video games, a certain video game genre, or a certain game in a video game genre - there is also heterogeneity in taste.

7.4 Encountered problems and possible solutions

We will now address some of the problems encountered after analyzing the results. These problems could be solved or polished in another iteration of "The Test", so we will also mention some possible solutions to each problem. Due to time constraints a new iteration of "The Test" was not possible, but should someone wish to continue this work, he or she will probably find these solutions useful.

1. **Most of the players did not take part in an important and characteristic feature/phase of the Co-op Craft game: collecting resources from volcanoes.** This meant that the Resources DDAA could not improve the players' performance as expected, because although it did detect the poor performance of these players and erupt resources more often and these erupted resources lasted much longer, C_R did not get closer to the standard because the players did not collect these resources. The only exception to this was P12, with whom the Resources DDAA worked perfectly.

Why did most of the players avoid going to the volcanoes? There are several reasons. Many of them tried to visit a volcano in its first eruption, but died quickly traversing the map or died in the volcano plateau. The feral monsters, both in the way to the volcano and on the volcano plateau, were much stronger creatures than the regular minions, and it seems they intimidated the players. Finally, most

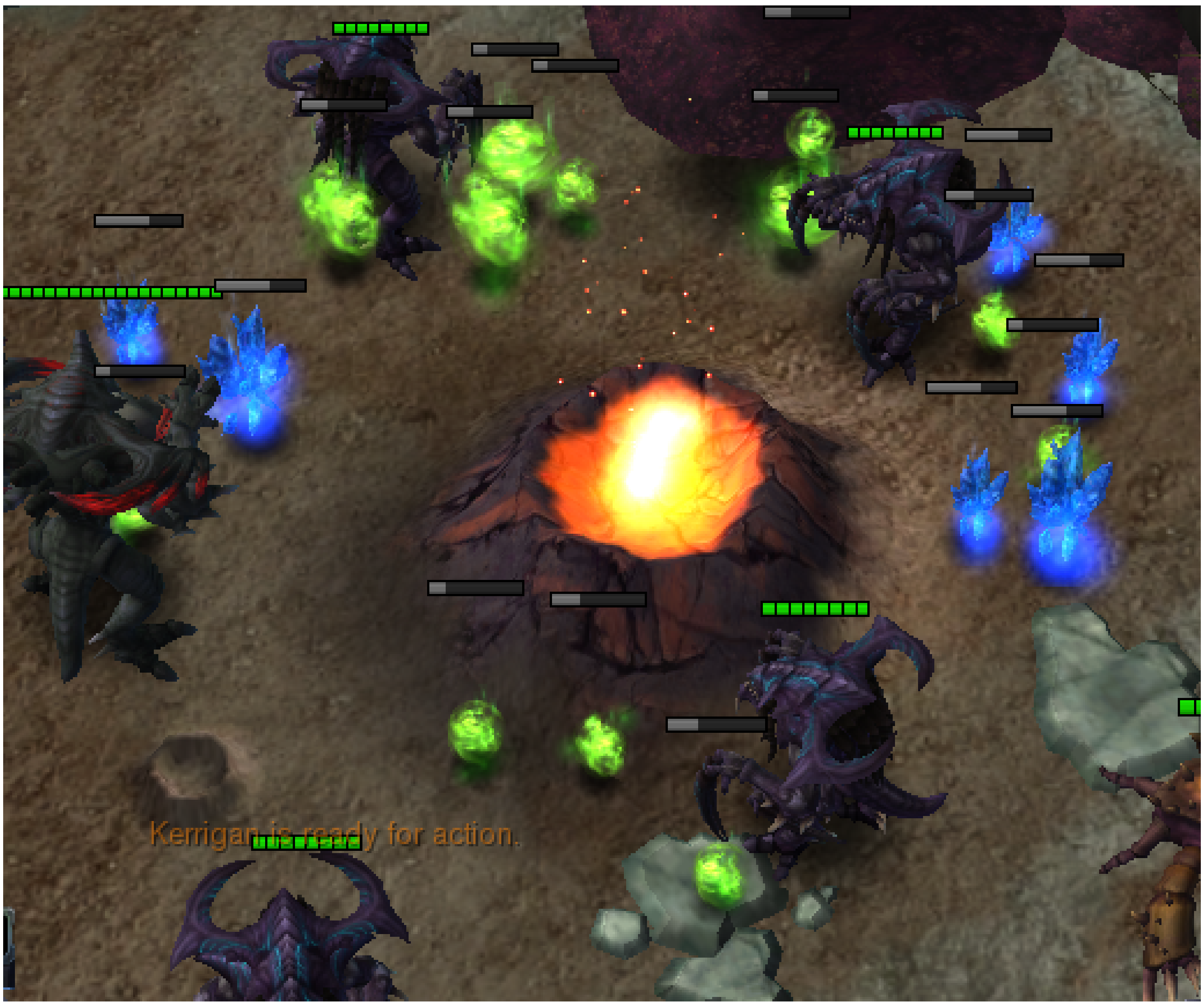


Figure 7.7.: A volcano saturated with resources waiting to be collected by the players, but most of the players never came. This was the situation by the mid and late game with all six volcanoes in the map. An unexpectedly common situation.

players focused all their attention in the main lane, where the main objectives and progressions were localized, after all. These players did not understand that the best strategy was to sneak and collect the erupted resources without fighting the feral monsters, and that even if they did die in the attempt, collecting these resources always pays off because one can upgrade the allied minions' army and the hero much more effectively with the extra resources. Because the players did not collect these resources, their $K(t)$ stayed far from the standard, and the Milestone DDAA kept making the game easier by sending less minions in the enemy waves. This resulted in less enemy units killed, in even less resources gathered and $K(t)$ dropping even farther away from the standard, as seen in 7.6. The actual overall effect of the DDA not only did not improve the players' performance collecting resources, it actually made it even worse! Fortunately the Milestone DDAA, the predominant factor in the DDA in Co-op Craft, did function as expected and kept the overall difficulty correctly attuned.

The idea behind guarding these resources with powerful feral monsters was to test the players' stealth and movement skills and to reward them, but it seems it backfired and discouraged players to keep trying. Only the more experienced and skilled players understood the importance of "farming" resources from feral monsters and volcanoes. Yet I myself would still be reluctant to drastically lower the strength of the feral monsters, as I believe there has to be a minimum challenge to obtaining a better resource income. Other MOBAs do include strong feral monsters that provide good rewards when defeated. However these MOBAs usually include more than one main central lane (three lanes - top, middle and bottom - are the classical standard for MOBAs) and the feral monsters are between the lanes. When switching lanes these players will be forced to sneak around or fight these feral monsters. This contrasts with the main and central lane in Co-op Craft, where feral monsters could seem like a temptation and a distraction from the main objective. Adding extra lanes to Co-op Craft or somehow forcing players to move or fight their way to the volcanoes could be a solution to the problem presented here. Bringing volcano plateaus closer to the main lane is also a possibility.

Aside from possible solutions, this problem brings up an interesting and important lesson for game and DDA designers: when deciding what significant parameters the DDA will observe and/or tweak, one must make sure that the players will mostly behave as expected and not ignore an important feature of the game, like in this case. An example of another instance of this problem: imagine we add DDA to the popular puzzle game Tetris™. Maybe the DDA detects that a certain type of piece would be useful for the player to complete rows, so it increases the chance for this certain type of piece to appear (see Figure 7.8). Now imagine that the player does not know how to rotate pieces, he just lands them the best he can but with the original rotation. In this case the "better" type of puzzle piece might actually become "worse" than other puzzle pieces, and the DDA is actually making the game harder instead of easier! Designers must either make sure that most players behave as expected in all game features, or alternatively that the DDA is ready to react if one of these essential features is ignored by the player (in this last example the DDA might detect that the player never rotates his puzzle pieces, and maybe hints him then with a message to do so; alternatively, the DDA might change the concept of "better" puzzle pieces for this new situation).

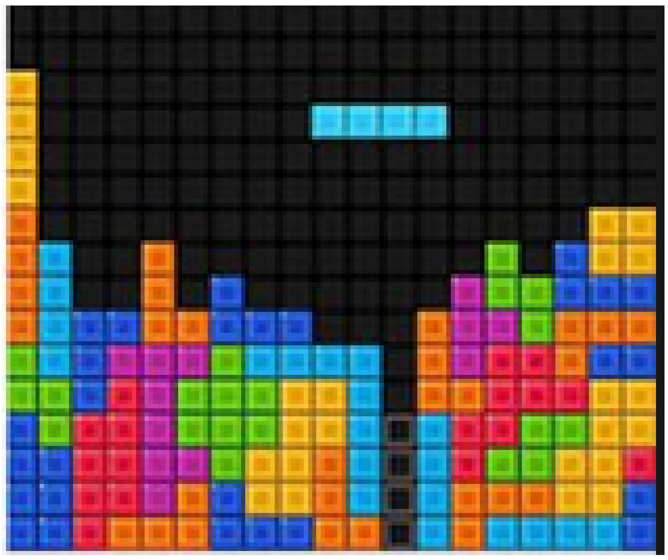


Figure 7.8.: The current piece is perfect to complete several rows in Tetris™... as long as the player knows how to rotate pieces!

2. **Most of the teams were homogeneously skilled.** Whereas the amount of teams was enough to test the concept of DDA in co-op games, in only one occasion (Team 6) did a heterogeneously skilled team participate. The other teams were composed of roughly equally strong, medium or weak players. The players did come from different gaming backgrounds and cultures, but it is true that the partners and teams shared a similar gaming background and culture.

This problem happened due to pairing reasons. It is not easy to get two players to play together for two hours, and when they do they would obviously prefer to meet up and play together with a friend. Additionally, one can not seriously determine the real skills of the players before they actually play the game. It would have been ideal to have, for example, Team 4 and Team 5 split and exchange partners, which would have granted two extremely heterogeneously skilled teams; but as mentioned the problem is anticipating a player's skill.

It does hold true that the only heterogeneously skilled team (Team 6) performed roughly equally as all the other homogeneously skilled team, and that they did feel like a team and have fun despite the skill gap. Yet heterogeneity in the skill of players is one of the main challenges in implementing DDA in co-op games, as we have already explained in Section 3.1 and throughout this whole work - even the title of this Master Thesis work hints at this! It would definitely be better if we had a few more heterogeneously skilled teams try Co-op Craft and analyze their results.

Possible solutions to this problem would be to run a modified or a new version of "The Test". For example, summon several people at a specified time and have them play first the tutorial and training phase (see page 28); afterwards rearrange them in teams where heterogeneity in skills is present. This would require several additional computers to play, and a good Internet connection. An alternative is to have players play Co-op Craft a third time - Game A and with a different human partner. The main reason why we could not carry out these solutions and increment the data for heterogeneously skilled teams was time constraints. But should future work in this subject be done, addressing this problem should be very feasible as well as necessary.

3. **Co-op Craft needs a better game balancing.** This problem is not as important as the two prior problems as the concept of DDA in co-op games has been successfully tested in spite of it, yet we believe it deserves mention. If Game B's standard difficulty were reduced, we would see more disparity in the results of B rather than mostly defeats by the 15 minute mark. Additionally, Game A becomes impossible to beat for good players - it should become more difficult for them but not exceedingly difficult to the point of becoming impossible. With these small changes we would probably see Game A's standard deviation decrease and Game B's standard deviation increase substantially. Thus Game A's standard deviation would be smaller than Game B's, as we expected when measuring the effectiveness of the Milestone DDAA (page 46).

8 Final Remarks

8.1 Conclusion

In this work we have explored the challenges, potential and possibilities of Dynamic Difficulty Adaptation in collaborative multiplayer video games. We were motivated to find a solution to currently popular collaborative games with a steep learning curve (like League of Legends™). We have used previous researches on Dynamic Difficulty Adaptation in single player games to mold, define and classify general approaches to Dynamic Difficulty Adaptation in collaborative games. We have focused on a subgroup of collaborative games - distinct-role collaborative video games, because we believed that the heterogeneously skilled teams challenge can be effectively addressed in this subgroup.

In order to test and prove that Dynamic Difficulty Adaptation can indeed be implemented in collaborative video games we have developed and implemented a collaborative game of our own - Co-op Craft, which includes three Dynamic Difficulty Adaptation Algorithms. We have analyzed, justified and classified these implemented algorithms according to our general conception of Dynamic Difficulty Adaptation in collaborative games. We have also offered alternatives to the implemented algorithms and even to the implemented game itself by presenting a game of a different genre, suggesting that there are many possible paths to implement Dynamic Difficulty Adaptation in collaborative games.

We have tested the Dynamic Difficulty Adaptation in our own developed game by having several teams of different gaming background play the game both with and without Dynamic Difficulty Adaptation. The results were generally positive, although we did meet two major problems: we lacked more heterogeneously skilled teams for testing the game; and the unexpected behavior of most players in one of the gameplay's basic features made the adaptation of one the three algorithms quite useless for them.

Despite the encountered problems we can definitely assure that Dynamic Difficulty Adaptation can be effectively implemented in collaborative games, the implemented game of this work serves as an example to this assertion. The heterogeneously skilled teams challenge can probably be addressed with the general approach in this work, yet we will not ascertain this due to not having enough gathered data of heterogeneously skilled teams in our experiment. We have also learned to be aware of observing and modifying significant key parameters of the game through Dynamic Difficulty Adaptation Algorithms, and that the player's interaction with the game can be unexpected and a hindrance to the desired results of Dynamic Difficulty Adaptation.

Dynamic Difficulty Adaptation has a great potential in collaborative games, especially in helping casual or beginner players learn one of the currently popular collaborative games, which are considered hard to learn. In our example game it helped these players perform better and enjoy the game more. Dynamic Difficulty Adaptation can also provide a balancing handicap for better players who wish to enjoy the game with not-so-skilled players, so it has big potential for a "fun mode". On the other hand, we should be reluctant of adding Dynamic Difficulty Adjustment in a "competitive mode", as it makes the game unfair for better-skilled players and blurs the sense of accomplishment that can be obtained from beating a game with a static level of difficulty.

8.2 Future Work

The subject of DDA in co-op games also has the potential for additional and deeper research. Due to time constraints the problems mentioned in Section 7.4 could not be solved in the final implementation of the Co-op Craft game. One of the first things to do if this work were to be continued is to solve the aforementioned problems: have the Resources DDAA adjusting the difficulty as expected for all players and gather more data for exclusively heterogeneously skilled teams.

Co-op Craft as a game has room for improvement. By adding a slightly bigger roster of heroes that would increase the replay value and also improving the User Interface, Co-op Craft could become a popular game in the StarCraft II Arcade. With a bit of advertisement in community forums many more people could play it and provide a lot of data for DDA in Co-op Craft. Apart from improving the game, new features can also be added and make the game even more interesting. Adding alternative DDA algorithms (Section 4.4.4) is an obvious new feature that can be added. But the gameplay can also be modified and open new DDA possibilities. For example, most of the StarCraft II units fire attacks that can not be dodged, yet a few select units do fire powerful attacks that can be dodged with some micro-management skill (see Figure 8.1). With more of these kinds of attacks, players could show an additional performance in how well they dodge attacks (not just in engaging or fleeing enemies). The DDA could then adjust different features in these kind of attacks (speed at which missiles are fired, number of fired missiles, enemy accuracy...).

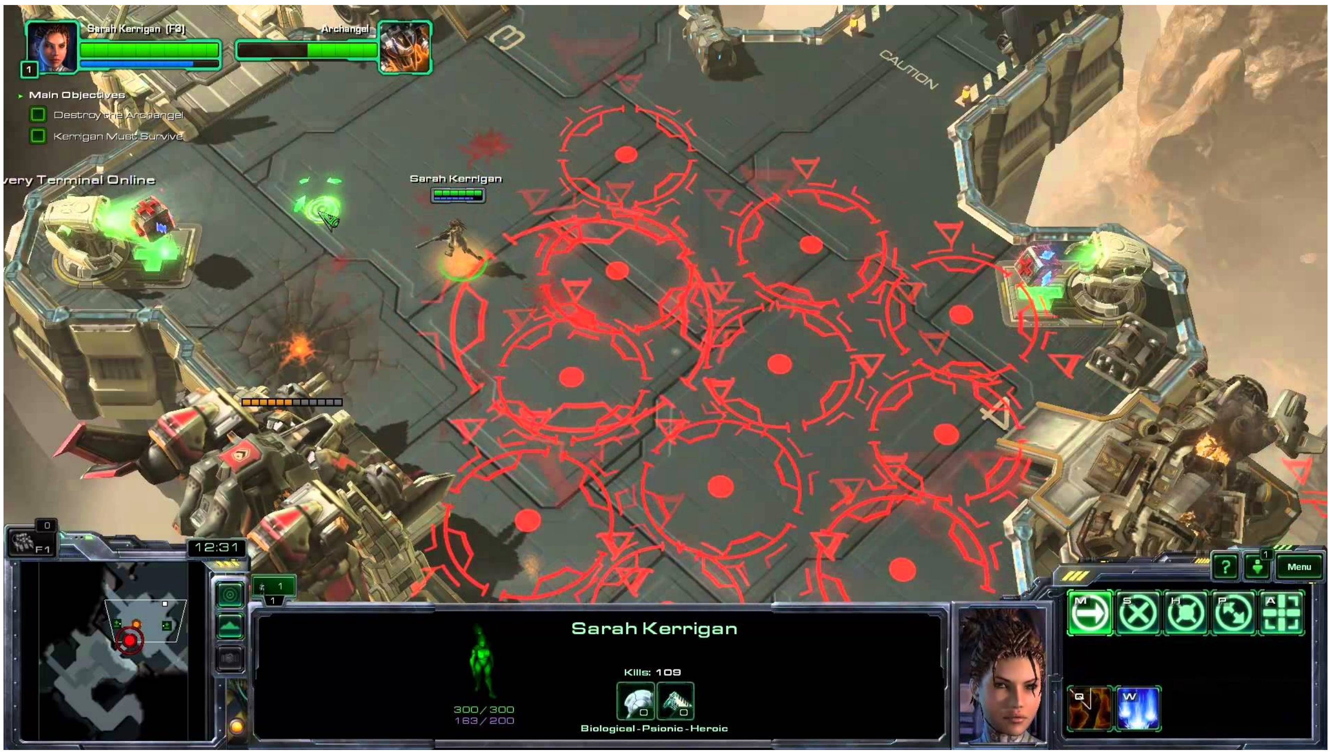


Figure 8.1.: Screenshot of one of the official Campaign Missions in StarCraft II. Here the player faces a map boss which fires powerful attacks that can however be dodged. The red circles indicate the player that the boss will soon fire missiles at the area, giving him time to move his character out of danger. Co-op Craft includes a bit of this gameplay mechanic, but were it to be fully implemented in the game, new possibilities for DDA would emerge.

It would be very interesting to see DDA implemented in other distinct-role co-op games, whether commercially or not. Obviously it would be nice if ADappTowr or a similar game were to be developed, as the idea is already there - explained in Chapter 5 of this work. But distinct-role co-op games can take many other forms.

Finally, the door is of course still open for anyone who desires to try and implement DDA in co-op games where players in the same team do not necessarily have distinct roles. We have chosen to focus on the subgroup of distinct-role co-op games because we believe our approach to overcoming the heterogeneously skilled teams challenge works well in this subgroup. But surely other approaches for DDA in other subgroups of collaborative games (or even collaborative games as a whole) can be conceived and put to the test.

A Glossary of terms

This is an alphabetically ordered list of important terms used in this work. Next to the term is its definition and in brackets the page number where it first appears or where it is defined.

ADappTowr *a puzzle platformer shooter distinct-role collaborative two-player video game. Both players control a character that can move, jump and shoot. These characters must ascend a tower by completing platforming levels, working together and performing different tasks. This game was left in a conceived state, whereas Co-op Craft was implemented* [P41]

Affective computing adaptation (in DDA context) (Chanel et al. [6]) *a kind of DDA that triggers according to a player's expressed feelings instead of his performance in-game* [P11]

APM ≡ **Actions Per Minute** (Lewis et al. [21]) *number of actions (such as selecting units or issuing an order) completed within a minute of gameplay in RTS games, most notably in StarCraft™* [P37]

Co-op Craft *a real-time battle arena distinct-role collaborative two-player video game developed solely to test DDA in co-op games. It is the implemented game in this work. Game A had DDA active and Game B had DDA disabled. See Sections 4.3 and 6.2 for detailed information* [P24]

Co-op games *See "Collaborative Games"*

Collaboration ≡ **Cooperation (lax)** (Roschelle and Teasley [25]) *a coordinated, synchronous activity that is the result of a continued attempt to construct and maintain a shared conception of a problem* [P7]

Collaborative Games ≡ **Co-op games** *the set of video games that involve two or more human players playing and working together as a team to accomplish a common goal* [P8]

C_R ≡ **collected_resources** *the total accumulated amount of resources collected at a specific moment in time in the Co-op Craft game. A key parameter for the Resources DDAA* [P31]

Cooldown *the minimum time that has to elapse between two consecutive uses of a spell. When applied to weapons it refers to the minimum time between consecutive firing rounds* [P29]

Cooperation (lax) *See "Collaboration"*

Cooperation (strict) (Dillenbourg [9]) *activity where partners split the work, solve sub-tasks individually and then assemble the partial results into the final output* [P7]

CSCL ≡ **Computer Supported Collaborative Learning** *a pedagogical approach wherein learning is done through interaction with other players in a collaborative serious game. DDA has potential to be used effectively in CSCL, as it can adapt the difficulty of the game in order to bolster learning* [P9]

DDA ≡ **Dynamic Difficulty Adaptation/Adjustment** ≡ **Dynamic Game [Difficulty] Balancing** ≡ **DGB** *the process of analyzing a player's ability and performance in a video game, and then accordingly adjust and change significant parameters, scenarios, and AI behaviors in order to adapt the game's difficulty and thus avoid the player from becoming bored (if the game is too easy) or frustrated (if it is too hard)* [P13]

DDAA ≡ **Dynamic Difficulty Adaptation Algorithm** *an algorithm that executes DDA in a video game* [P13]

Distinct-role collaborative games *collaborative video games where players in a same team have clearly distinct and complementary roles. Players depend on each other to achieve their common goal, and one player alone should not be able to complete the game with little to no help from the teammates* [P14]

Death Count *the total number of times a player dies in the Co-op Craft game* [P37]

Dynamic Difficulty Adaptation/Adjustment *See "DDA"*

eSports ≡ **Electronic Sports** *a new form of sports where the primary aspects of the sport are facilitated by electronic systems (normally computers). It has recently attracted large audiences that like to see professional video game players* [P4]

FPS ≡ **First Person Shooter** *a video game genre where players shoot projectiles in combat and where the game camera is in first-person perspective, i.e. simulates what the player would see through his eyes if he were inside the game* [P13]

Galaxy Editor (StarCraft II) *the official StarCraft II modding tool. Considered the best RTS engine development tool, but also hard to learn and master. The implemented game Co-op Craft was developed with this tool* [P22]

Heterogeneously skilled player groups/teams challenge *the dilemma of modifying the general difficulty of a game where players of different skills play together in a certain team. If the game becomes generally easier, the skilled players will become even more bored; and if the game becomes generally more challenging, the less-skilled players will find it even more challenging and frustrating* [P13]

Interference challenge *the malfunctioning of DDA due to the unexpected influence of other players* [P13]

K ≡ **K_factor** *a factor defined in Co-op Craft to estimate a player's performance according to his collected resources. See equation 4.1 for its definition.* $K = 1$ *marks the standard performance,* $K < 1$ *indicates under-performance and* $K > 1$ *indicates over-performance. It is a key factor for the Resources DDAA and the Milestone DDAA* [P31]

Kerrigan *the ranged hero in Co-op Craft. She can attack airborne units, is responsible for recruiting air allied units and collects vespene gas. She has the role of an "assassin". Her spells are more oriented to increase her damage output* [P30]

Kill Count *the total amount of units killed by a player in the Co-op Craft game* [P37]

LoL ≡ **League of Legends™** *a currently popular MOBA game with over 67 million monthly players in 2014* [P3]

Melee attacker *a video game unit that attacks body-to-body, at a short range. Swordsmen are usually considered melee attackers. Zeratul is the melee hero in Co-op Craft* [P29]

M ≡ **milestone_parameter** *a Co-op Craft parameter defined as the sum of the health of all enemy structures minus the sum of the health of all allied structures (See equation 4.4). It is an easy to calculate parameter which decreases when enemy towers are damaged and which increases when allied structures take damage. It is a key parameter for the Milestone DDAA* [P34]

MMOG ≡ **Massive Multiplayer Online Games** *an online video game capable of supporting large numbers of players simultaneously in the same world/map/game instance. MMOGs can potentially benefit from DDA, as the always changing community of active players influences the overall difficulty of these games* [P12]

MOBA ≡ Multiplayer Online Battle Arena ≡ ARTS ≡ Action Real-Time Strategy *a subgenre of RTS games which defines games where teams of human players control different hero characters and push along weaker allied computer-controlled creatures to progress in a map and eventually destroy the enemy's main structure to win. Examples of MOBAs are League of Legends™and DotA 2™, which are currently very popular in the eSports.* [P3]

Modding (in video games) *alteration of the original content from a video game in order to make it operate in a different manner from its original version, usually done by the general public or a developer. Mods can be entirely new games in themselves and can be created for any genre of game (they are especially popular in FPS, RPG and RTS games). Mods are not stand-alone software and require the user to have the original game in order to run* [P21]

Ranged attacker *a video game unit that attacks from a distance or range, usually throwing projectiles. Archers are usually considered ranged attackers. Kerrigan is the ranged hero in Co-op Craft* [P30]

R_t ≡ **respawn_time** *the time a Co-op Craft player must wait to get back into the action after dying. It is defined differently for disabled DDA (see equation 4.2) and active DDA (see equation 4.3). It is a key parameter for the Respawning DDAA* [P33]

RTS ≡ Real Time Strategy *a subgenre of strategy video games where the action takes place in real time and NOT incrementally in turns (like how it happens in the TBS ≡ Turned Based Strategy subgenre)* [P3]

Rubber-band effect *a notorious effect in racing games that apply DDA. If a human racer gets too far ahead in the lead his opponents always catch up, and if he gets too far behind, they always slow down* [P10]

Scalability challenge *refers to the difficulty of extending single player DDA to multiplayer DDA when the single player DDA already substantially consumes the physical system's resources. Extending resource demanding single player DDA to multiplayer DDA can lead to game crashes and online lag, none of which are ever desired* [P13]

Sense/suspension of disbelief (`www.dictionary.com`) *willingness to suspend one's critical faculties and believe the unbelievable; sacrifice of realism and logic for the sake of enjoyment. If a video game disrupts a player's sense of disbelief, it is producing the opposite effect: the player can not enjoy the game because it is too unrealistic or illogical* [P17]

StarCraft™ *a military science-fiction RTS game by Blizzard Entertainment, released in 1998. A highly acclaimed and popular game that has sold millions of copies worldwide. Critics consider it one of the most influential games of all time due to it being the first RTS game that introduced unique and distinct factions that play very differently from each other, but still keep a balanced gameplay* [P19]

StarCraft II™ *the 2010 sequel to Blizzard Entertainment's acclaimed StarCraft™. StarCraft II includes improved graphics and a much more user-friendly User Interface, while preserving the lauded gameplay and balance from the original. For its RTS map editor or modding tool, see "Galaxy Editor"* [P19]

"The Test" *A step-by-step procedure that all testing teams and players took in order to test DDA in co-op games. It involves watching an introductory video, playing Co-op Craft twice (once with DDA active, once with DDA disabled, and exchanging heroes between games) and answering a questionnaire. See Chapter 6 for more information* [P45]

Zeratul *the melee hero in Co-op Craft. He can attack ground units, is responsible for recruiting ground allied units and collects minerals. He has the role of a "tank". His spells are more oriented to assist allied units or cripple enemy units* [P29]

Bibliography

[1] Ernest Adams. The designer's notebook: Difficulty modes and dynamic difficulty adjustment. *Gamasutra: http://www. gamasutra. com/(last access 01/2009)*, 2008.

[2] Wilfried Admiraal, Jantina Huizenga, Sanne Akkerman, and Geert Ten Dam. The concept of flow in collaborative game-based learning. *Computers in Human Behavior*, 27(3):1185–1194, 2011.

[3] Allbrecht. Lol: The brutality of the newbie experience. *The Mittani: https://www.themittani.com/features/lol-brutality-newbie-experience*, 2013.

[4] D Arey and E Wells. Balancing act:«the art and science of dynamic difficulty adjustment». In *Game Developers Conference*, 2001.

[5] Sander Bakkes, Pieter Spronck, and Jaap Van den Herik. Rapid and reliable adaptation of video game ai. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(2):93–104, 2009.

[6] Guillaume Chanel, Cyril Rebetez, Mireille Bétrancourt, and Thierry Pun. Emotion assessment from physiological signals for adaptation of game difficulty. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(6):1052–1063, 2011.

[7] Nathan Combs. The intelligence in the mmog: From scripts to stories to directorial ai. In *Proceedings of the Other Players conference*, 2004.

[8] Mihaly Csikszentmihalyi and Isabella Selega Csikszentmihalyi. *Optimal experience: Psychological studies of flow in consciousness*. Cambridge university press, 1992.

[9] Pierre Dillenbourg. What do you mean by collaborative learning? *Collaborative-learning: Cognitive and Computational Approaches.*, pages 1–19, 1999.

[10] Simon Ferrari. From generative to conventional play: Moba and league of legends. *Proceedings of DiGRA 2013: DeFragging Game Studies*, 1(1):1–17, 2013.

[11] Kiel M Gilleade and Alan Dix. Using frustration in the design of adaptive videogames. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 228–232. ACM, 2004.

[12] Andrew Glassner. Interactive storytelling: Techniques for 21st century fiction. 2004.

[13] Peter Haas. Blizzard, valve settle dota lawsuit. *CinemaBlend: http://www.cinemablend.com/games/Blizzard-Valve-Settle-DOTA-Lawsuit-42430.html*, 2012.

[14] Raija Hämäläinen, Tony Manninen, Sanna Järvelä, and Päivi Häkkinen. Learning to collaborate: Designing collaboration in a 3-d game environment. *The Internet and Higher Education*, 9(1):47–61, 2006.

[15] Juho Hamari and Max Sjöblom. What is esports and why do people watch it? *Available at SSRN 2686182*, 2015.

[16] Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.

[17] Robin Hunicke, Marc LeBlanc, and Robert Zubek. Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, page 1, 2004.

[18] David W Johnson and Roger T Johnson. *Learning together and alone: Cooperative, competitive, and individualistic learning* . Prentice-Hall, Inc, 1987.

[19] Yubo Kou and Bonnie Nardi. Regulating anti-social behavior on the internet: The example of league of legends. 2013.

[20] Alex Kuang. *Dynamic Difficulty Adjustment*. PhD thesis, Worcester Polytechnic Institute, 2012.

[21] Joshua M Lewis, Patrick Trinh, and David Kirsh. A corpus analysis of strategy video game play in starcraft: Brood war. In *Proceedings of the 33rd annual conference of the cognitive science society*, pages 687–692, 2011.

[22] Santiago Ontanón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A survey of real-time strategy game ai research and competition in starcraft. *Computational Intelligence and AI in Games, IEEE Transactions on*, 5(4):293–311, 2013.

[23] José Bernardo Rocha, Samuel Mascarenhas, and Rui Prada. Game mechanics for cooperative games. *ZON Digital Games 2008*, pages 72–80, 2008.

[24] Andrew Rollings and Ernest Adams. On game design. 2003.

[25] Jeremy Roschelle and Stephanie D Teasley. The construction of shared knowledge in collaborative problem solving. In *Computer supported collaborative learning*, pages 69–97. Springer, 1995.

[26] Antonio A Sánchez-Ruiz. Predicting the outcome of small battles in starcraft. 2015.

[27] Magy Seif El-Nasr, Bardia Aghabeigi, David Milam, Mona Erfani, Beth Lameman, Hamid Maygoli, and Sang Mah. Understanding and evaluating cooperative games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 253–262. ACM, 2010.

[28] Paul Tassi. Riot's 'league of legends' reveals astonishing 27 million daily players, 67 million monthly. *Forbes. Recovered from: http://www. forbes. com/sites/insertcoin/2014/01/27/riots-league-of-legends-reveals-astonishing-27-million-daily-players-67-million-monthly*, 2014.

[29] Viktor Wendel, Felix Hertin, Stefan Göbel, and Ralf Steinmetz. Collaborative learning by means of multiplayer serious games. In *Advances in Web-Based Learning–ICWL 2010*, pages 289–298. Springer, 2010.

[30] Viktor Wendel, Michael Gutjahr, Stefan Göbel, and Ralf Steinmetz. Designing collaborative multi-player serious games for collaborative learning. *Proceedings of the CSEDU*, 2012, 2012.