2014

Intelligent Camera Control in Game Replays

Daniel Wei-Shen Phang
Lehigh University

Intelligent Camera Control in Game Replays

by

Daniel Wei-Shen Phang

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

May 2014

Approved and recommended for acceptance as a thesis in partial fulfillment of the requirements for the degree of Master of Science.

Date

Thesis Advisor

Chairperson of Department

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Héctor Muñoz-Avila, for his guidance throughout this research. I would not have been so interested in artificial intelligence in games in the first place without his excellent class on AI Game Programming. His feedback and suggestions were also invaluable when writing this thesis.

To my family, thank you for your love and support during my five years of studies. I hope to be make all of you proud.

Finally, I am grateful for my girlfriend, Sui Ying. You are the love of my life.

Contents

Acknowledgements **iv**

List of Tables **viii**

List of Figures **ix**

Abstract **1**

1 Introduction **2**

2 Background **5**

2.1 What is Dota 2? 5

2.1.1 A Brief Introduction 5

2.1.2 Anatomy of a Match 6

2.2 Camera Control in Spectator Games 11

3 A Hybrid Machine Learning Solution **13**

3.1 Motivation 13

3.2 Game Events 14

3.3 Machine Learning 16

3.3.1 Feature Extraction 16

3.3.2 Classification 18

3.4 Intelligent Camera Design 19
3.4.1 Single Hero Events 19
3.4.2 Multiple Hero Events 19
3.4.3 Default Camera Behavior 20
3.4.4 An Event Importance Hierarchy 22

4 Implementation 23

4.1 Preprocessing Events 24
4.1.1 Labeling Game Events 24
4.1.2 Parsing Game Replays 24
4.2 Classification 26
4.2.1 Classification on Labeled Game Events 26
4.2.2 Classification on Unlabeled Game Events 29
4.3 Generating Camera Movements 30
4.3.1 Single Hero Events 30
4.3.2 Multiple Hero Events 31
4.3.3 Choosing Events to Focus 32
4.4 Applying Camera Movements to Replays 32

5 Empirical Results 34

5.1 Experiment Details 34
5.1.1 Questions 35
5.2 Results 35
5.2.1 Intelligent Camera Control 36
5.2.2 Directed Camera Control 36

6 Closing Remarks 37

6.1 Conclusion 37

6.2 Future Work 38

A Source Code **41**

B Approval Letter **42**

C Consent Form **44**

D Questionnaire **47**

E Results **49**

Biography **70**

List of Tables

3.1 Features for individual events 17

3.2 Features for team events 17

3.3 Heuristics for determining default camera behavior 21

4.1 Performance of *k*-Nearest Neighbors classification 29

List of Figures

2.1 The game map 6

2.2 The Radiant's two barracks, which control creep strength 7

2.3 The Radiant's Ancient, its most important building 8

2.4 The side shop, where players can purchase various items 9

2.5 Roshan's Pit 9

2.6 A hero farms creeps for gold and experience 11

3.1 Example of a region connecting three heroes from The Dire during a team fight 21

3.2 Event importance hierarchy 22

4.1 Labeled events corresponding to a game replay 25

4.2 Player-specific replay data 27

4.3 Algorithm for computing feature vectors 28

4.4 Feature vectors and labels corresponding to a replay 28

4.5 Algorithm for following single heroes 30

4.6 Algorithm for following multiple heroes 31

4.7 Algorithm for choosing most important event 32

4.8 Example camera movement data 33

Abstract

In this thesis, we describe a new method for implementing intelligent automated camera control in spectator games, specifically in replays of the video game *Dota 2*. First, we give a brief overview of *Dota 2* and how spectator mode functions in games. We survey current camera control systems used in industry and conclude that current solutions use only simple heuristics to control the camera. We propose an improved solution that uses a hybrid of both machine learning and heuristics for performing camera control, which would be able to detect important and interesting events. Next, we describe an implementation of our solution in the form of a Python prototype. Finally, we present the results of testing our prototype on users found via the Amazon Mechanical Turk platform. We conclude that although our prototype did not fare well in testing, it could potentially replace current systems used in games. We also explore possible areas for future work, such as use in games other than *Dota 2*.

Chapter 1

Introduction

Video games have evolved to become an important and widespread part of society. Since the advent of the personal computer, many players have enjoyed playing some genre of video games, whether it is casual, first-person shooter, real-time strategy, or adventure. In the modern world, online multiplayer games have become extremely popular. Players usually compete with each other or work in teams. Such popular games include *Counter-Strike*, *Starcraft*, *World of Warcraft*, *Dota 2*, and *League of Legends*, among others, where players are exposed to players from all around the world. *World of Warcraft* is especially popular because of how it immerses players in such a massive community of players.

Recently *electronic sports*, or *esports* has gained significant growth, resulting in large increases in viewership and prize money, especially in the year 2012[1]. Games such as *League of Legends* and *Starcraft II* are played competitively, where players usually compete individually. Players can also compete in teams, which is most common in games such as *Dota 2*, where two teams of five players compete against each other. *Dota 2* is extremely popular since it is not only free to play, but has good replayability. In fact, this is the most played multiplayer game on Valve Corporation's Steam platform[2].

Although such games are very fun to play, many players also enjoy watching them.

[1]Source: `http://readyupgaming.com/2012/12/forbes-2012-the-year-of-esports`

[2]Source: `http://kotaku.com/steams-most-popular-and-unloved-games-1563645958`

Such games are often very exciting to spectate because of its competitiveness. Usually, players watch these games using an interface called spectator mode. However, online streaming platforms such as *Twitch.tv* have also become very popular[3]. Players can then view the events of a live match as they unfold. Alternatively, they can watch game replays, if they were recorded by the game server. These games are often referred to as spectator sports because they are just as fun to watch as they are to play. Spectator mode is extremely important to game players because it has much entertainment value, while at the same time is helpful for improving players' skills.

However, an important but often ignored part of spectator mode is the game camera: where the game is focused when a player watches a live or replayed match. In the past spectator mode usually only allowed manual camera control. That is, players needed to manually pan the camera to interesting parts of the game world. In the present, most games have some form of automated camera control that focuses on different parts of the game as it progresses. Such a system can be used to follow players or capture particular interesting events.

Automated camera control is important for games because it allows players to watch live games or replays more effectively. Players do not have to manually control the camera in order to jump between different game events. One could simply view a game replay or live match and have the camera automatically filter out all but the most important events. Although such automated camera control systems are good, we would like them to be more intelligent. Most current automated camera systems use simple heuristics to decide which region of the game world to focus on. For example, the automated camera control system in *Dota 2*, called the *directed camera*, likely uses mostly unit activity as a heuristic. Although this generally works well, it might miss out important game events or focus on irrelevant ones.

In the rest of this thesis, we present a more intelligent solution to automated camera

[3]Source: `http://www.twitch.tv`

control in games. We specifically design our solution around the game *Dota 2*, using game replay data. This solution incorporates both machine learning techniques and heuristics to create more accurate automated camera control. This intelligent camera system We combine both machine learning and heuristics to implement a camera system that performs well, and is more informative to video game players.

However, automated camera control can be difficult to design because of two reasons. First, it is hard to decide what constitutes an interesting event. Many games use heuristics to try to decide this, but players often behave unpredictably. Second, a player's screen only shows a portion of the entire game, so the camera system must move accurately and smoothly. Otherwise, it might miss important events or cause frustration to the player. Fortunately, games like *Dota 2* function in a limited, two-dimensional world. Players only use one map, and generally only control a single unit (or perhaps a few, depending on the hero choice). Unlike in games like *Starcraft II*, we do not need to keep track of too many events, and we do not have many different map types.

It is interesting to note that in the past, general camera control has received some attention (Christie et al., 2008). However, the problem is complicated because of the degrees of freedom, i.e where the camera can be focused on (Drucker and Zeltzer, 1994). In contrast, there has been much less attention to the problem of automated camera control from the perspective of a spectator. However, this topic has been somewhat discussed in (Rabin, 2004). This problem is somewhat simpler because of the limited environment of *Dota 2*. For example, we do not have to focus on specific camera angles, since we are only concerned with a top-down perspective. The types of game events are also somewhat restricted and predictable. Together, this makes it easier to control camera movement.

Chapter 2

Background

2.1 What is Dota 2?

2.1.1 A Brief Introduction

Dota 2 is a popular multiplayer online battle arena (MOBA) (a subgenre of real-time strategy) video game. Officially released by Valve Corporation in 2013, it is currently their most played multiplayer game, with hundreds of thousands of players every day. *Dota 2* is the standalone sequel of *DotA*, also known as *Defense of the Ancients*, a custom map created in 2003 for Blizzard Entertainment's game *Warcraft III: The Frozen Throne*.

The game consists of individual matches in which two teams of five players compete against each other. Each player controls a single hero with four unique abilities. The objective of the game is to destroy the opposing team's ancient, while simultaneously defending one's own ancient. What makes *Dota 2* extremely interesting is that fact that matches are always unpredictable. Although they follow the same format, players can choose between many different heroes, each with unique and interesting abilities. Teams have the possibility of employing countless combinations of hero abilities and tactics. Players need to have sharp reflexes, risk management skills, and good decision making ability in order to be successful. Teams must collectively try to outplay the other team in order to win team fights and get closer to their objective.

2.1.2 Anatomy of a Match

Although *Dota 2* matches can be unpredictable, they all share common elements. Here we describe the basic layout of the game map, as well as the general structure of a single match.

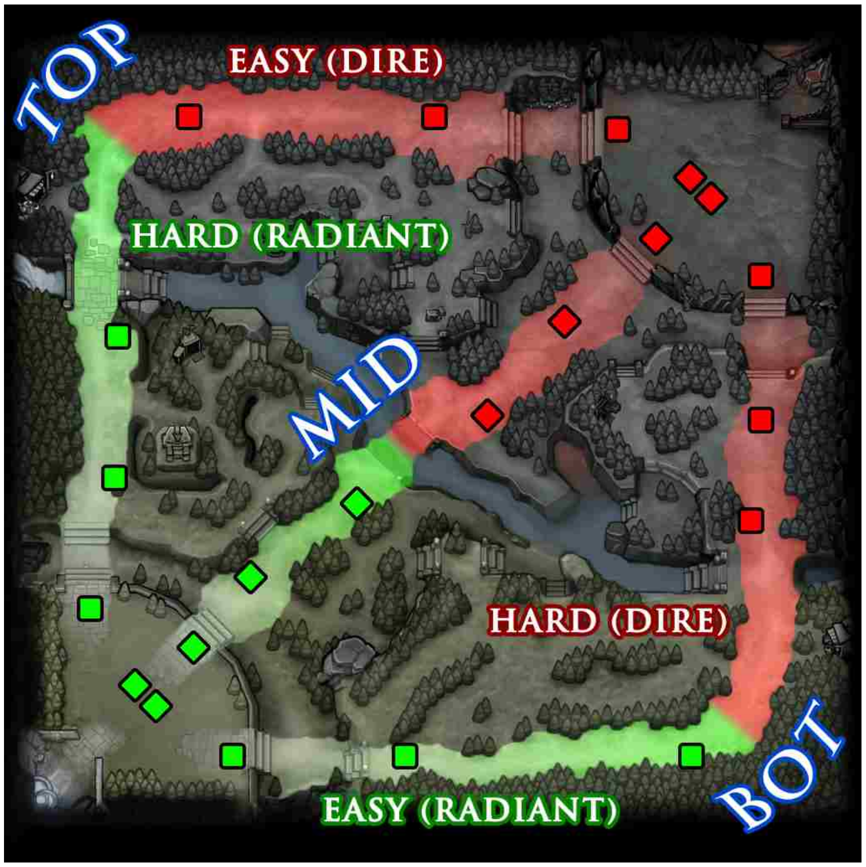


Figure 2.1: The game map[1]

As seen in Figure 2.1, the map itself is symmetric: each team starts out at a certain corner of the map, and is part of a base. The Radiant starts out in the bottom left, while The Dire starts out in the top right. Each of the two bases are connected by three main

[1]Source: `http://www.playdota.com/forums/showthread.php?p=8491882`

paths as highlighted in the figure (referred to as top, middle, or bottom lane). The bases are also divided by a river. Periodically, units known as creeps would spawn from the endpoints of each base and move towards each other along those paths.

The base itself consists of several buildings, each with different functions. For example, towers (the squares along each of the three lanes) help to protect the base. The barracks (Figure 2.2) are buildings that control creep strength. The barracks are located behind the towers protecting the entrance to the base. When the barracks are destroyed, the opposing team gains stronger creeps for that particular lane. The ancient is the most important building (Figure 2.3. The team that destroys this wins the game. Other buildings not highlighted serve as decoys in order to distract enemy creeps from the ancient.

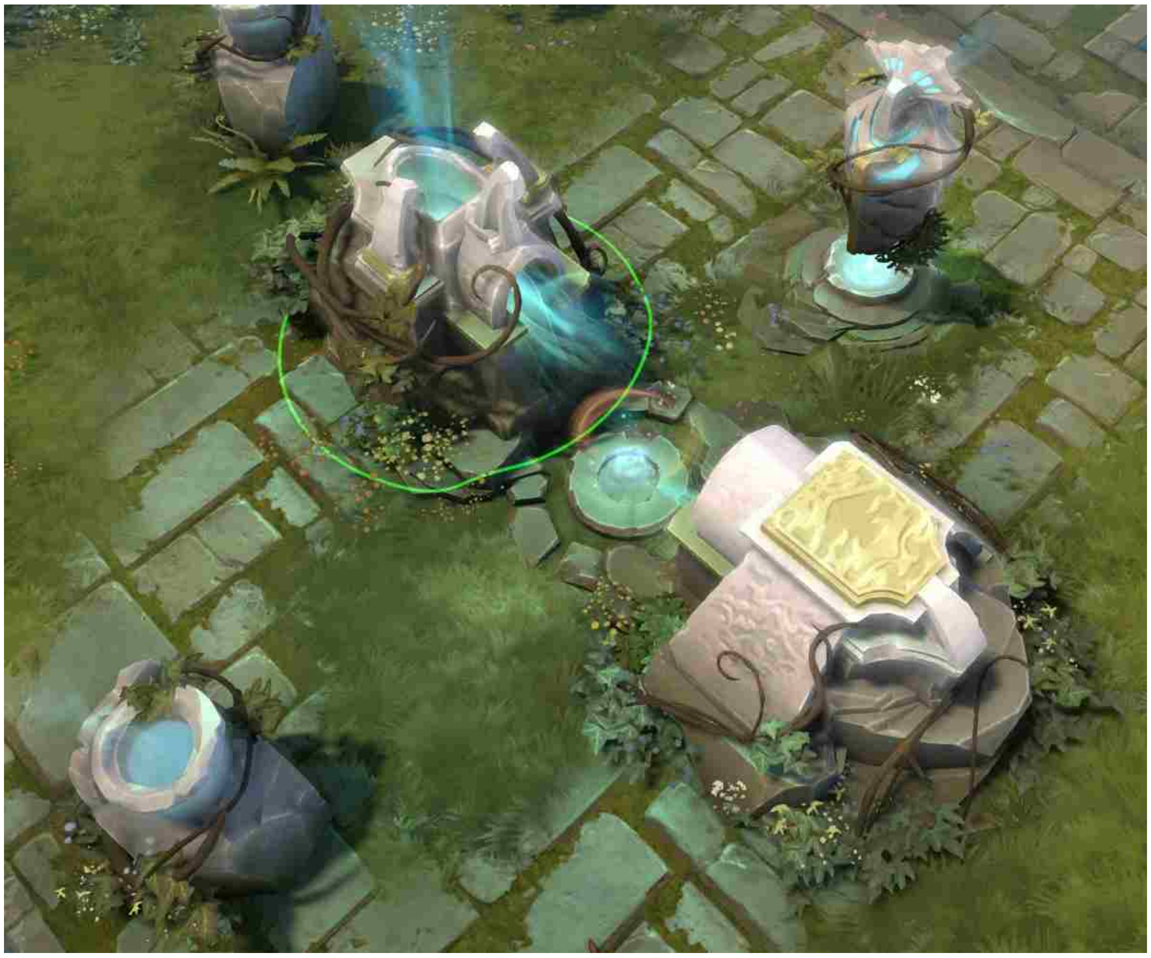


Figure 2.2: The Radiant’s two barracks, which control creep strength

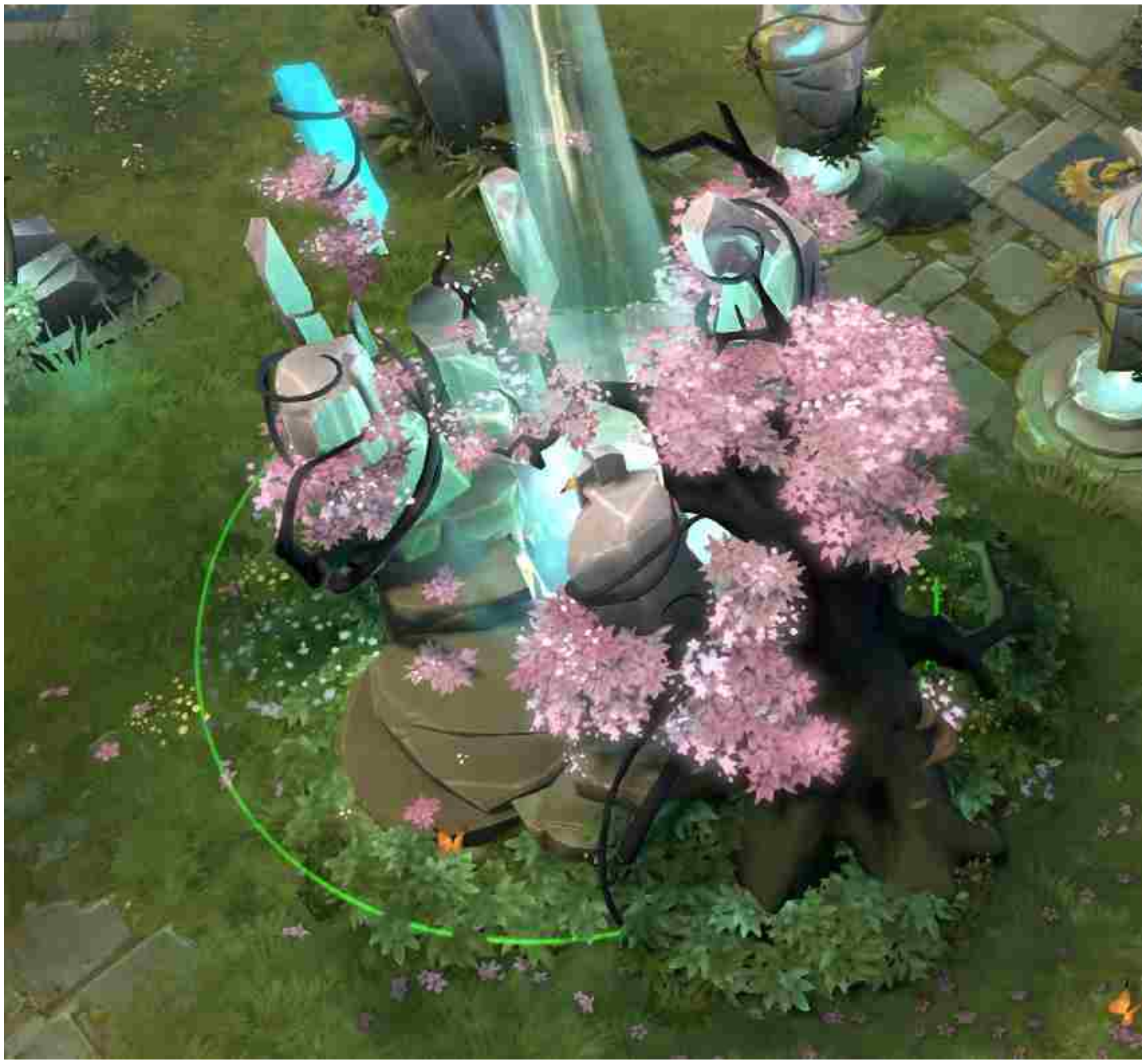


Figure 2.3: The Radiant's Ancient, its most important building

There are also several key locations in the map. For example, there is a side shop (Figure 2.4) where players can purchase various items not available in the base. Runes, which enhance a player's hero (e.g by giving it maximum movement speed, invisibility, etc.) spawn on either side of the river every two minutes. There is also an extremely powerful and durable unit called Roshan (see Figure 2.5), who grants large experience and gold for the team that defeats him. In addition, Roshan drops an item, called *Aegis of the Immortal*, that grants a second life to a player. Natural features of the map are also important. For example, players can hide from enemies in trees, or fight from the high ground, giving evasion and sight advantages.

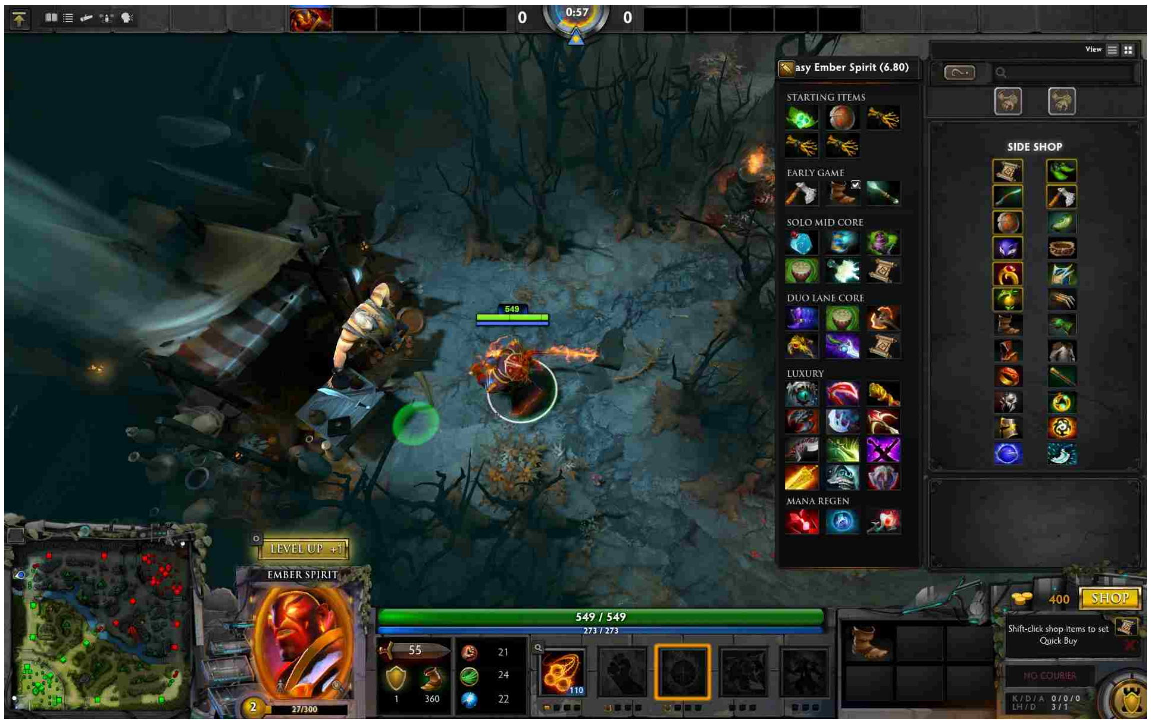
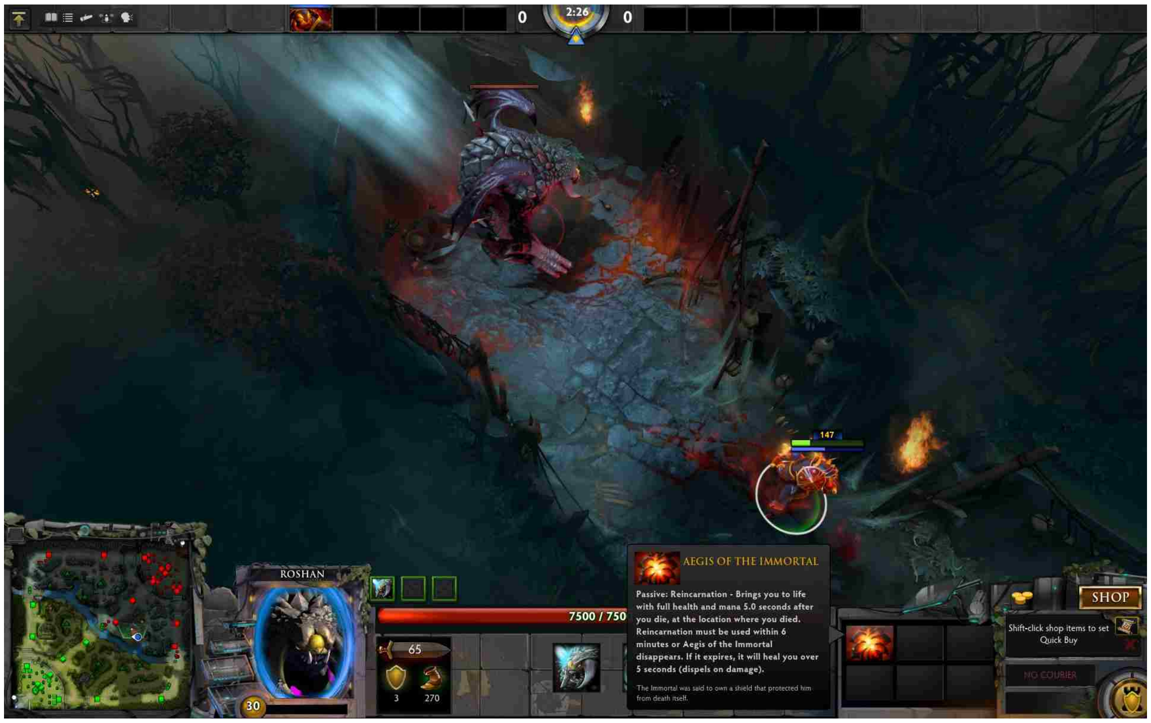


Figure 2.4: The side shop, where players can purchase various items



AEGIS OF THE IMMORTAL

Passive: Reincarnation - Brings you to life with full health and mana 5.0 seconds after you die, at the location where you died. Reincarnation must be used within 6 minutes or Aegis of the Immortal disappears. If it expires, it will heal you over 5 seconds (dispels on damage).

The Immortal was said to own a shield that protected him from death itself.

Figure 2.5: Roshan’s Pit

Matches consist of several phases, described below in more detail.

Laning Phase / Early Game

The focus of this phase is on gaining experience and gold by attacking creeps as in Figure 2.6(also known as *farming*). Players generally divide themselves into their specific lanes, and stay in that lane. Traditionally, there are two heroes in the top and bottom lanes, and one hero in the middle lane. The carry, a hero type that becomes strong late in the game, normally goes to the bottom lane and top lane for The Radiant and The Dire respectively. This is known as the *safe lane* or *easy lane* for that team, because the creeps initially meet on that particular team's side of the river. In this phase, players also try to attack enemy heroes if the opportunity comes (known as *harassing*).

Team Fight Phase / Middle Game

In this phase, players are strong enough to roam around the map and engage in ambushes and small team fights. Teams might start destroying the first tier of towers in each lane to give themselves more map control. Some players might also try to fight neutral creeps in order to gain more experience and money.

Pushing Phase / Late Game

In this phase, players attempt to push, which refers to destroying the enemy's base. This generally involves all players coordinating and grouping up to destroy enemy buildings. Many team fights occur, some of which can ultimately decide the outcome of the game.

As one can see, a *Dota 2* match is quite structured. There are many similar types of events throughout the game, which we shall identify later in this thesis. However, the game play itself is quite unpredictable. Heroes can move in unpredictable fashions, and use unique combinations of skills.

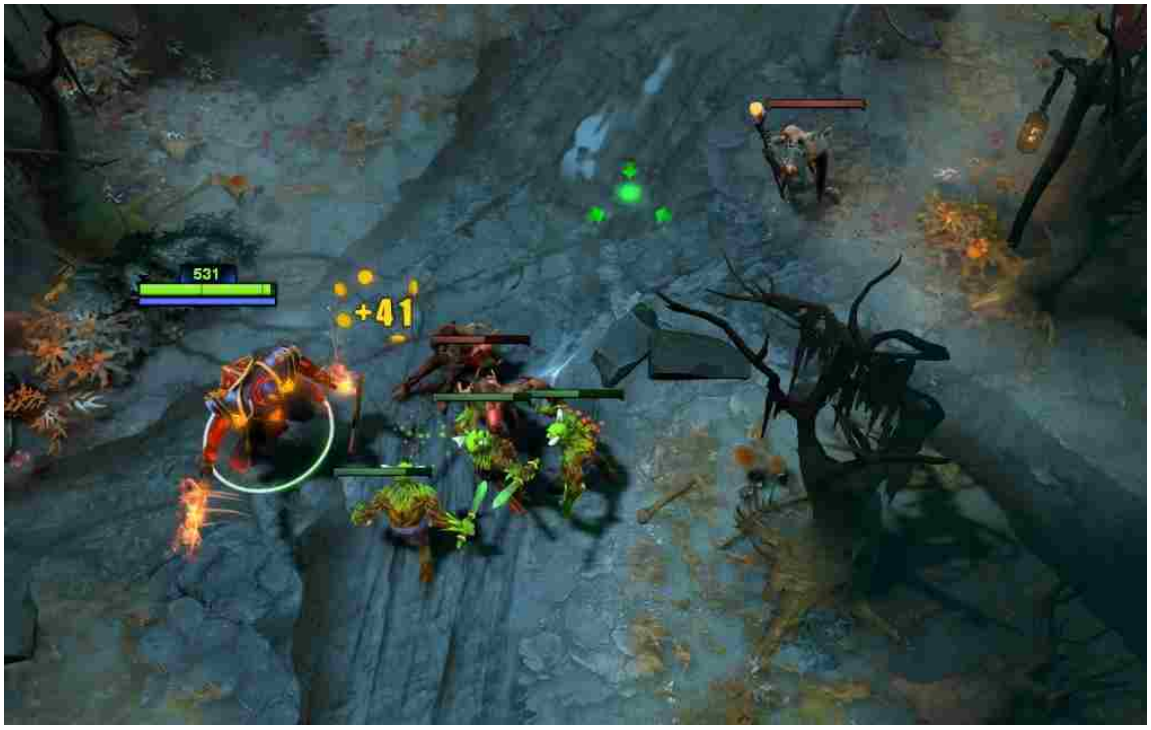


Figure 2.6: A hero farms creeps for gold and experience

2.2 Camera Control in Spectator Games

In modern video games there is often a spectator mode, which is an interface that enables players to watch live matches or replays (if they are recorded by the server). Spectating matches is important to players not only because they are entertaining, but also because players can use them to improve their own skills.

Dota 2 is often called a *spectator game* because so many players watch live matches, whether via spectator mode or other means such as livestreams. In fact, spectating live *Dota 2* matches is so popular that live-streaming sites such as *Twitch.tv* get several million unique viewers per month, and perhaps more during big game tournaments[2].

One important aspect of spectator mode is the in-game camera that controls what part of the map a spectator is looking at. In *Dota 2* specifically, players can choose between three camera modes: *free camera*, *directed camera*, or *hero chase camera*. They are de-

[2]Source: `http://www.twitch.tv/year/2013`

scribed in more detail below:

Free Camera

This mode allows the player to freely control the camera to view whatever part of the map he or she chooses.

Directed Camera

This mode automatically adjusts the camera to view the parts that are deemed most interesting according to its algorithm.

Hero Chase

This mode focuses chasing or following the hero currently selected by the spectator.

In this thesis, we are concerned only with the directed camera. Although its exact algorithm is proprietary, we believe that it primarily uses a unit activity heuristic. Thus, the camera generally focuses on areas with high activity. We also found that it also tends to follow heroes throughout a game match. As a result, the directed camera works decently as a baseline: it focuses on interesting game objects (the heroes, controlled by players) and picks up on important events (team fights, which are often areas of high activity).

Chapter 3

A Hybrid Machine Learning Solution

3.1 Motivation

Although the directed camera works well enough, it does have its disadvantages. Since it primarily measures unit activity, it cannot distinguish events based on their specific event type. For example, complex events such as *ambushing* or *escaping* events are often not captured by this system. In addition, the directed camera mode sometimes focuses on *uninteresting* events simply because of their high activity. Thus, the goal of our research is to develop a more intelligent camera that captures interesting events, not just events with high unit activity (which are not necessarily interesting). With our solution, we believe that the spectator will have a better experience.

How should such a solution be designed? Clearly, unit activity is not always the best heuristic to use when choosing which events should be focused by the camera. We could improve our solution by considering other heuristics, such as what *types* of unit activity is there, or the amount a player *moves* during events. This would allow us to easily detect interesting events. For example, if a hero casts his or her *ultimate* ability, this generally indicates an impending team fight, because those abilities can only be cast every so often.

Although a heuristic-based solution alone could perform well, we instead opted a hybrid solution that incorporates both heuristics and machine learning. This works on the basis of first manually labeling interesting game events from past game replay data. Then one can use those events to attempt to automatically identify interesting game events from different game replay data, by comparing the similarity of the unlabeled replay segments to previously labeled game events. An intelligent camera control system would ideally try to first focus the camera on any identified game events. However, if the current segment of the game stream does not have any interesting game event, the system should fall back to a traditional heuristic-based approach.

We chose this hybrid approach for several reasons. First, we were interested in seeing how machine learning can be applied in identifying interesting game events. Second, we believe it can potentially be much better than just a heuristic-based system. In fact, some heuristics can even be used as inputs to a machine learning algorithm, which would ultimately improve its performance. When there are no interesting events, heuristics such as unit activity can be used.

3.2 Game Events

Since our solution works by identifying interesting game events, we must first define what we mean by a *game event* in *Dota 2*. A game event, as we defined it, is a particular sequence of movements and actions performed by one or more heroes, possibly against one or more heroes or units, along with some label. Examples of movements include: a hero moving around while in his/her lane, a hero moving to a different lane in order to ambush another hero, and so on. Example actions include: a hero simply attacking another unit, a hero casting all of its abilities during a team fight, and so. This definition of a game event concerns only player-controlled heroes, and not other units. Although other types of events could be taken into account (e.g which towers are destroyed, where the creeps are moving, etc.), we felt that player events are the most interesting.

The following describes some of the game events we identified as most commonly occurring throughout a game:

Farming

The act of killing enemy creeps and neutral units in order to gain experience and gold. While this is not necessarily an interesting event (since it involves heroes simply attacking creeps), it is a staple event common in the early parts of matches. Heroes need to do this in order to become stronger and gain gold in order to buy items.

Pushing

The act of destroying enemy units and buildings over a prolonged period of time. This could involve a single hero, or multiple heroes on a team attempting to destroy buildings such as towers and barracks.

Ambushing

The act of ambushing (and potentially killing) an enemy hero, usually from a different lane from the ambushing hero. Heroes that have a stealth ability such as invisibility often participate in ambushing other heroes. Ambushing could also involve multiple heroes teaming up against a single hero.

Harassing the act of damaging an enemy hero in the same lane, either by attacking or casting skills on that hero. Heroes with ranged attacks are notably more proficient at harassing enemy heroes, because they can attack from a distance.

Team fight

The act of multiple heroes from both teams engaging in a fight, usually resulting in several kills spread across one or both teams. This usually involves heroes using multiple skills and attacks consecutively.

Escaping

The act of running away from enemy heroes, usually as the result of a failed team fight or ambush, or being harassed by the enemy team.

Although this is not a complete list of player-related events, we feel these are the most interesting in the perspective of the spectator. We do not include uninteresting events such as purchasing items or returning to a fountain.

3.3 Machine Learning

Given a set of labeled game events, we are then able to classify new game events using machine learning algorithms. The idea is essentially to compare the similarity of unlabeled segments in a game replay to labeled game events. To do this, we need some set of measurable properties about the game event itself, known as *features*. Then we can use some kind of machine learning algorithm to classify unlabeled segments (that also has a corresponding set of features).

3.3.1 Feature Extraction

There are some obvious feature choices that we can associate with game events. For example, we could measure simply unit activity during that game event (e.g how much distance or displacement a hero makes, how many actions are used). We have decided to devise features (Table 3.1) without caring about specific hero types or spell names. We did this because we wanted to simplify the machine learning process. Otherwise, there would be too much to measure about a game event. We also developed two different feature sets: one for individual, or single hero, events, and one for team, or multiple hero, events. The reasoning for this was that there would be too much to keep track of during team events, if we measured complete statistics about each individual hero. The individual event features are described in Table 3.1, while the team event features are described in Table 3.2. We can then develop a two-layer classifier based on these two feature sets.

Table 3.1: Features for individual events

Feature	**Description**
Movement	how much movement is made by the hero involved.
Displacement	how much a hero actually moves.
Creep Attacks	how many attacks on a creep or neutral unit are performed.
Hero Attacks	how many attacks on a hero are performed.
Skill Uses	how many skills are used.
Health Change	how much the hero's health changes.

We believe that these features are good choices for the set of events we are trying to classify. For example, the *displacement* feature can easily determine whether a hero is *farming* or *ambushing*. From our anecdotal experiences, heroes generally do not displace much while staying in lane and farming enemy creeps, but they will need to displace a lot of ground when they move to a different lane to ambush another hero.

Unfortunately, since there are a total of up to ten heroes involved in a game event, we could potentially have a total of $9 \times 10 = 90$ features per event. This is clearly impractical for when we want to classify team fights or pushing events. This was the reasoning for using a separate feature set for team events.

Table 3.2: Features for team events

Feature	**Description**
Movement	average movement made by the team.
Displacement	average displacement made by the team.
Closeness	how close team members are to each other.
Enemy Closeness	how close team members are to the enemy team.
Attacks	total number of attacks by the team.
Skill Uses	total number of skill uses by the team.
Health Change	percentage health change of the team.

As one can see, this feature set is very similar to the feature set for individual events. However, these features are generally *aggregating* features. That is, they summarize the team's characteristics as a whole.

3.3.2 Classification

To actually classify new game events, we need to use some kind of machine learning algorithm. The algorithm that we ultimately decided to use in this research is the *k*-Nearest Neighbors algorithm, which is also described in Duda et al. (2000).

The k-Nearest Neighbors algorithm is simple, though effective. The idea is to classify new examples based on how similar they are to their *k* nearest neighbors or past examples, based on a simple Euclidean, or straight-line distance metric that compares feature vectors. For example, if *k* is 5, we look at the five closest neighbors. The algorithm then uses majority vote to determine a new event's label. For example, if the five closest past examples are all of label "A", then we also label the new example "A".

In our case, our examples are game events and the feature vectors consist of the features we described in the previous section. The *k*-Nearest Neighbors algorithm traditionally weights all features equally, but some features could be more important than others. For example, *farming* events generally have no need for the *health change* feature, because heroes generally do not lose or gain health while farming.

There is also one issue to consider: how do we automatically extract test examples from new replay data? We do not know when a game event will start and end (that's the idea of our research: to identify when and where interesting game events occur). Thus, we must determine a way to divide replay data into finite segments. Clearly, the most comprehensive solution would be to examine every possible segments: e.g every second segment, then every two second segment, and so on. However, this would be highly infeasible because we would need to check too many segments per match.

A more practical solution would be to simply divide a match into *n* second segments. We can choose any *n*, but it should not be too small or large. If it is too small, segments are too short to be able to classify them accurately. If it is too large, our segments might contain multiple interesting events, and we may not be able to capture all of them since our classifier would only give it a single label. We shall describe this in more detail in the next chapter.

3.4 Intelligent Camera Design

Given a set of labeled game events, how do we translate that into a sequence of camera movements? From the previous sections, we have devised an algorithm to classify interesting game events, but we still need to design a camera that moves intelligently to those events.

From analysis of the directed camera mode in *Dota 2*'s spectator mode, we have discovered that a hero-centric camera works effectively. Specifically, the camera should generally always be focused on heroes. However, intelligent camera movement can be tricky in some situations. For example, how do we handle team fights? What about default behavior, i.e when there are no identified interesting game events?

3.4.1 Single Hero Events

When events involve only a single hero, camera control is simple. We need only follow the hero of interest. For example, when a hero is farming neutral or enemy creeps, we need only follow the hero of interest. Although this is uninteresting, this type of event is essential when there are no other interesting events at the current moment. During the early stages of the match, most heroes are farming; we do not want our camera to be idle in this case.

3.4.2 Multiple Hero Events

When events involve multiple heroes, camera control is more sophisticated. There are multiple active heroes during the event, so it is hard to decide where our camera should be focused. In this case, using a heuristic-based approach helps. Suppose we have detected a team fight event: it might be interesting to, for example, focus on the heroes whose health is decreasing quickly during the event, or heroes who cast many skills on enemies. This approach allows us to always capture interesting events such as hero kills. However, its disadvantage is that our camera may jump too much between heroes during a team fight event.

Another more suitable approach is to use a sort of centroid heuristic: focus the camera where the center of activity is, instead of on individual heroes. This has the advantage of the camera not switching focus too much between heroes, while still being able to focus on the area where activity is generally concentrated. For example, if only two heroes are involved in a fight, the camera should be focused on the midpoint between those two heroes.

Figure 3.1 shows an example of a region focusing on three heroes of The Dire. We can then calculate the initial focus point of the camera, i.e the centroid of those heroes. As team fight activity occurs, we shift the camera focus point towards the heroes who initiated that activity. Our camera can then smoothly capture all events in the team fight without jumping too much between heroes.

3.4.3 Default Camera Behavior

We also need to consider the default behavior of our intelligent camera control system. For example, what if our classifier determines that a particular segment is not similar enough to any of our event classes? In this case, we must revert to some default behavior. We use some heuristics to determine which location our camera should focus: unit activity such as ability uses and attacks. This will help avoid focusing on completely uninteresting events.

For example, suppose our classifier is not about to label a game segment as a particular type of event. Then we can simply focus on the hero with the most unit activity, which is given by computing several heuristics shown in Table 3.3. Note that they are almost

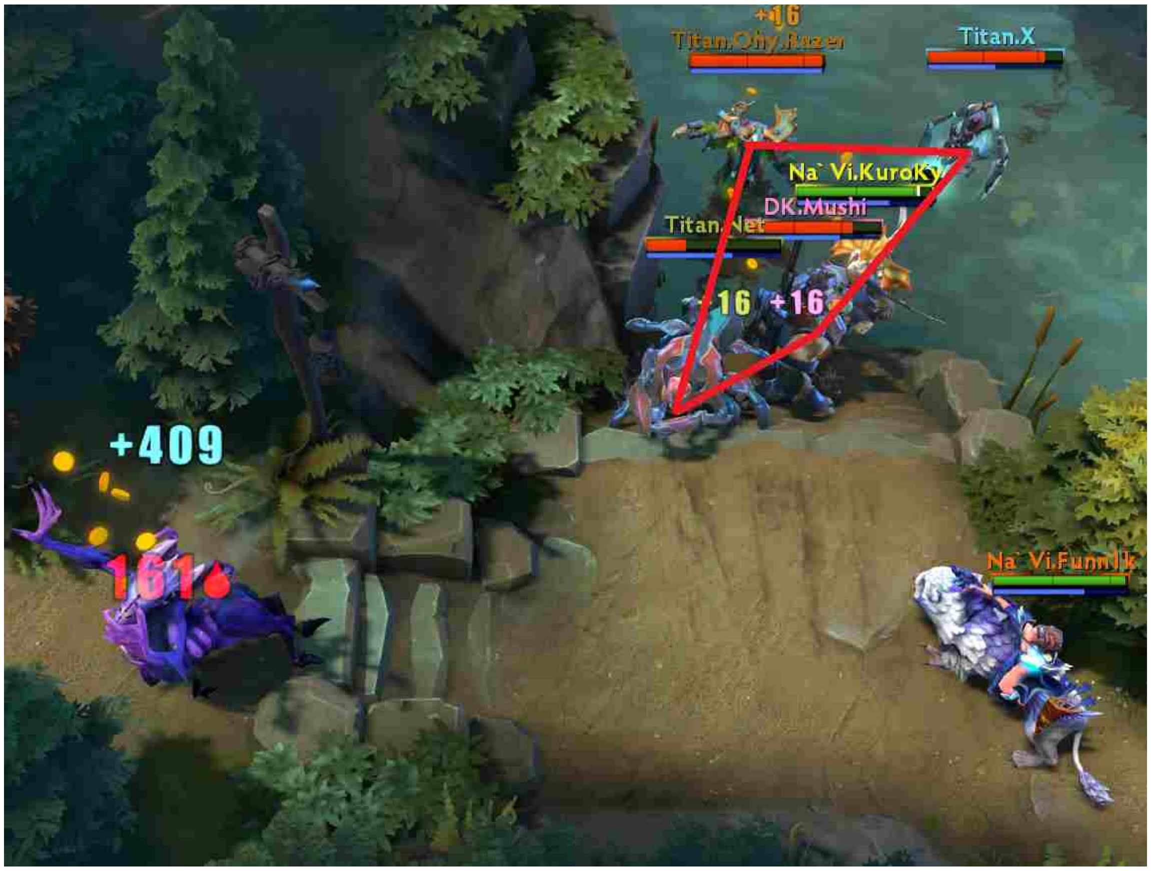


Figure 3.1: Example of a region connecting three heroes from The Dire during a team fight

identical to the set of features associated with individual game events. Although this type of behavior might focus on less interesting events, this is certainly better than focusing on random heroes.

Table 3.3: Heuristics for determining default camera behavior

Heuristic	**Description**
Movement	how much movement is made by the hero involved.
Displacement	how much a hero actually moves.
Attacks	how many attacks are performed.
Skill Uses	how many skill uses are performed.

3.4.4 An Event Importance Hierarchy

Although we have determined what kinds of events there are, there is still the issue of determining what events should be focused on during a replay. What if two interesting events occur at the same time? Our intelligent automated camera control system must focus on only one. For this reason, we can develop an event importance hierarchy. We do not need to use machine learning to determine the importance of events; simple heuristics should suffice.

Team events are clearly more important than individual events, but there is an importance ranking within each group as well. For example, *ambushing* events should be prioritized over *farming* events. In the case of two identical events occurring at the same time, we prioritize the one with more impact. We could use heuristics such as *number of heroes involved*, *number of hero deaths*, and others.

The complete event importance hierarchy is shown in Figure 3.2.

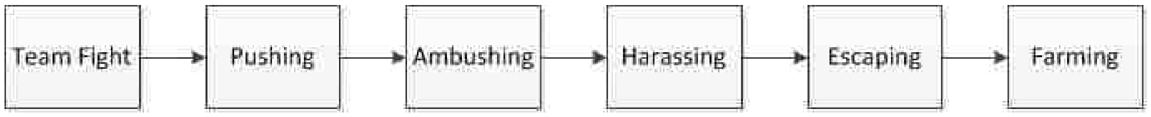


Figure 3.2: Event importance hierarchy

Chapter 4

Implementation

To implement our hybrid solution, we used *Dota 2* game replays as both our training and test data. The reason for this was that *Dota 2* replays are ready available and easy to parse using free and open source tools. Our implementation consisted of four parts: pre-processing game replays, classifying new game replays, generating a sequence of camera movements, and applying those camera movements to a replay.

Unfortunately, *Dota 2* does not have an interface for programmatically controlling the camera while a game replay is running. To workaround this problem, we instead generated a list of mouse events, and applied them using a Python script via clicking the minimap or dragging the mouse.

We chose Python as the only implementation language for several reasons. First, at the time of implementation, a *Dota 2* replay parser was only available in Python. Second, Python has a robust machine learning toolkit known as Scikit Learn (Pedregosa et al., 2011), which allowed us to easily implement our solution. Lastly, we felt that Python will be more understandable should one wish to verify or extend our research. Although Python is slower than statically typed languages such as Java, we believed that its readability and ease of use outweighs this cost. In addition, the scope of our experiment and size of our data sets were small enough that Python is sufficient.

In this chapter, we will only describe the high-level implementation details for our solution. Please refer to Appendix A for details on how to obtain the Python prototype source code used in this research.

4.1 Preprocessing Events

We first needed to get labeled game events from replay data in the form of a feature vector and corresponding label. This involved two steps. First, we manually labeled game events from various replays. Then, we used a Python library to parse the replay data. By having both the labeled events and replay data, we can then build the feature vectors associated with the game events.

4.1.1 Labeling Game Events

Since our solution involved machine learning, we needed to manually label events from game replays. To do this, we simply watched various game replays and looked for the events we described in the previous chapter. We labeled events by their start and end time, hero indices involved, and event type. For example, hero indices 0-4 correspond to the Radiant team, while indices 5-9 correspond to the Dire team. We then manually wrote JSON-encoded files containing these events. Figure 4.1 shows an example of such a file. We chose JSON, short for JavaScript Object Notation, for its easy-to-read notation.

4.1.2 Parsing Game Replays

Although we have labeled our events, this is not enough to perform machine learning on. We needed to parse the replay itself in order to extract information such as hero positions and other important data. However, *Dota 2* replays contain a plethora of data, much of which is not needed for our implementation. For example, we could know what modifiers

```
[
  {
    "start": "13:20",
    "end": "13:53",
    "label": "teamfight",
    "heroes": [1, 3, 4]
  },
  {
    "start": "13:53",
    "end": "13:56",
    "label": "harassing",
    "heroes": [1]
  },
  {
    "start": "13:56",
    "end": "17:00",
    "label": "farming",
    "heroes": [2]
  },
  {
    "start": "18:00",
    "end": "18:18",
    "label": "ambushing",
    "heroes": [3]
  },
  {
    "start": "18:00",
    "end": "18:18",
    "label": "push",
    "heroes": [5, 6, 7, 8, 9]
  }
]
```

Figure 4.1: Labeled events corresponding to a game replay

are on any hero (e.g whether a hero has a poison debuff), what specific spells are used, or what spells are available for a certain hero, during any point of the game. We chose not to use these data in our implementation, because this would increase the size of our feature space significantly. In addition, by parsing *Dota 2* replays, we can get a more compact JSON encoded file.

To do this, we used the Skadi/Tarrasque libraries[1], which are free and open source Python libraries for parsing *Dota 2* replays. Skadi is the low-level parser, while Tarrasque is a more human-friendly parser built on top of Skadi. Although Skadi is more powerful, we used Tarrasque because we did not require extensive low-level data. We primarily required data such as hero positions, attacks, and skill uses, and statistics such as hero health and mana. We will not show the details of this, as this simply consisted of using Tarrasque to loop through a replay stream and save relevant data into a Python dictionary structure. The dictionary data structure is then exported to a JSON-encoded file using Python's JSON library. A simplified example of the output of this process is shown in Figure 4.2. Note that the time is given in minutes. Actual replay data would contain more information such as more player statistics as well as data for multiple times.

The next task involved using the JSON-encoded replay data along with our JSON-encoded replay events in order to compute feature vectors for each game event. Figure 4.3 shows the general algorithm for doing this.

After running this algorithm, we obtain a list of feature vectors and a list of corresponding labels. Figure 4.4 shows an example of what this data would look like, in a JSON-encoded file. This particular example contains only single hero events.

4.2 Classification

4.2.1 Classification on Labeled Game Events

The first step of the machine learning processed involved training and testing a *k*-Nearest Neighbors classifier solely on labeled game events. For this, we used data from multiple replay files. Specifically, we used two replays: *Orange vs. DK, The International 3, Game 1*, and *Orange vs. DK, The International 3, Game 2*. These are tournament level matches

[1]Source: `http://github.com/skadistats`

```
{
  "players": {
    "radiant": [
      "Na‘Vi.Puppey",
      "Na‘Vi.XBOCT  (4)",
      "Na‘Vi.Dendi",
      "Na‘Vi.KuroKy",
      "Na‘Vi.Funn1k"
    ],
    "dire": [
      "Orange.NeoES-Mushi",
      "Orange.NeoES-Net",
      "Orange.NeoES-X",
      "Orange.NeoEs‘kyxY !",
      "Orange.NeoEs-Ohy‘"
    ]
  },
  "time_data": [
    {
      "player_info": [
        {
          "player": 0,
          "health": 568,
          "max_mana": 286.03491210938,
          "mana": 286.03491210938,
          "y": -6784,
          "x": -7168,
          "max_health": 568,
          "events": []
        }
      ],
      "time": 0.02
    }
  ]
}
```

Figure 4.2: Player-specific replay data

held at *The International 3*, a *Dota 2* tournament held by Valve Corporation[2]. Our training set consisted of labeled game events the first replay, and our test set consisted of labeled

[2]Source: `http://www.dota2.com/international/mainevent/results/champions`

```
function BuildFeatureVectors(dictionary replayData, list replayEvents )
    featureVectors = new list
    for event in replayEvents do
        segment = getReplaySegment(replayData, event[‘start’], event[‘end’])
        feature = computeFeatureVector(segment, event[‘heroes’])
        featureVectors.append(feature)
    end for
    return featureVectors
end function
// Gets the relevant segment of the replay data
// Uses binary search
// binarySearchLeft finds the index less than or equal to argument
// binarySearchRight finds the index greater than or equal to argument
function getReplaySegment(list replayData, float start, float end)
    leftIndex = binarySearchLeft(start)
    rightIndex = binarySearchRight(right)
end function
function computeFeatureVector(list segment, list heroes)
    featureVector = new list
    for data in segment do
        Compute different features depending on hero count
    end for
    for feature in computedFeatures do
        featureVector.append(feature)
    end for
    return featureVector
end function
```

Figure 4.3: Algorithm for computing feature vectors

```
[
  [
    [245760, 737280, 4, 0, 0, 11],
    [262144, 131072, 0, 2, 2, 142]
  ],
  [
    "farming",
    "harassing"
  ]
]
```

Figure 4.4: Feature vectors and labels corresponding to a replay

game events from the second replay. There were approximately 40 labeled events in each set, but each event was split into many segments of equal length (five seconds). This gave us a total of a few hundred events in each of the training and test sets.

To train our classifier, we used the Scikit Learn library as described at the start of this chapter. This library allowed us to easily perform k-Nearest Neighbors classification. Scikit Learn also had various tools for preprocessing data sets, which made it very straightforward to perform machine learning on our data sets. The exact details of how to do this can be found in the Scikit Learn documentation.

First, both data sets sets were normalized to have a mean of zero and variance of one. Our k-Nearest Neighbors classifier was then fitted with the training set, and subsequently tested on the test set. We varied the parameter k in order to achieve the best accuracy. The results are presented in Figure 4.1

Table 4.1: Performance of k-Nearest Neighbors classification

k	**Accuracy**
1	0.851
3	0.884
5	0.933

The accuracy of our k-Nearest Neighbors classifier was pretty good, especially when we used three or five neighbors.

4.2.2 Classification on Unlabeled Game Events

The next step of machine learning involved testing our k-Nearest Neighbors classifier solely on unlabeled events. By unlabeled, we mean that these events are essentially consecutive five-second segments split from a replay. For this, we used the game *Orange vs. Natus Vincere, The International 3, Game 1*. Note that there can be potentially up to 10 or more game events per segment, since there are 10 heroes. In addition, we may also have team fight events that consist of two or more heroes.

Unfortunately, we could not measure the performance of our classification because these events are unlabeled (and it would be physically infeasible to manually label each of the possible five-second segments). We simply observed the generated camera movements and noted how it performed *subjectively*. This will be expanded on in the next section.

4.3 Generating Camera Movements

The next part of our implementation involved generating camera movements, which determine where the camera should focus at any time. In the following subsections, we detail how both single hero and multiple hero events are handled by our implementation

4.3.1 Single Hero Events

To focus on single hero events, our camera control used the algorithm shown in Figure 4.5. As described in the previous chapter, this algorithm simply pans and centers the camera on heroes that should be followed. It also ensures that that camera switches focus between heroes in a smooth manner. If the new hero to be focused on is close enough, we smoothly pan to that hero instead of immediately focusing the camera to that location.

```
function SingleHeroEvent(int newHeroIndex, float duration)
    newPos = getHeroPosition(newHeroIndex)
    if currentHeroIndex ≠ null then
        currentPos = camera.getPosition()
        if distance(currentPos, newPos) < threshold then
            camera.smoothPan(newPos)
        end if
    else
        camera.setPosition(newPos)
    end if
    camera.follow(newHeroIndex, duration)
end function
```

Figure 4.5: Algorithm for following single heroes

4.3.2 Multiple Hero Events

When focusing on multiple hero events, our camera control used a more sophisticated algorithm. As described in the previous chapter, jumping between multiple heroes to quick is not ideal behavior, as this may detract from the spectator's experience. Thus, we implemented the centroid approach. During multiple hero events such as team fights, the camera would instead focus on the center of hero activity. As the camera detects new hero activity, it would shift its focus towards that hero. The general algorithm is given in Figure 4.6.

```
function MultiHeroEvent(list involved, float duration)
    newPos = calculateCenter(involved)
    currPos = camera.getPosition()
    if distance(currPos, newPos) < threshold then
        camera.smoothPan(newPos)
    end if
    while currentDuration < duration do
        activeIndex = camera.getActiveHero()
        newPos = recalculatePos(currPos, involved, activeIndex)
        camera.smoothPan(newPos)
    end while
end function

function calculateCenter(list involved)
    center = Center of positions of each hero in involved
    return center
end function

function recalculatePos(position currPos, list involved, int activeIndex)
    activity = (Activity of activeIndex) / (Activity of all heroes in involved)
    shift = activity × (currPos - getPosition(activeIndex))
    center = center + shift
    return center
end function
```

Figure 4.6: Algorithm for following multiple heroes

4.3.3 Choosing Events to Focus

To choose events to focus, we simply sorted all detected events at the current five-second interval, according to the event importance hierarchy shown in the previous chapter. Figure 4.7 shows how this is done. The reason for having a function *random()* is in the case of ties: we simply pick a random event out of all possible choices.

```
1: function ChooseEvent(list activeEvents)
2:     Sort activeEvents by our event importance hierarchy
3:     bestEvents = getFirst(activeEvents)
4:     return bestEvents.random()
5: end function
```

Figure 4.7: Algorithm for choosing most important event

4.4 Applying Camera Movements to Replays

The last part of our implementation was to apply our generated camera movements to a game replay in real-time. Since *Dota 2* has no interface for programmatically controlling a game replay, we resorted to a different approach: we issued mouse commands to control the replay camera. This would be similar to how a spectator might control the camera.

For example, to focus directly at a specific location on the map, we clicked on a part of the game minimap (the small version the game map). This involved translating game coordinates to mouse coordinates. To pan the camera, we used a middle click mouse drag. We used this method for following heroes involved in multiple hero events. However, this was not used to follow heroes simply because its precision was too low. To follow heroes, we simply had the mouse click and hold the hero's portrait, which centered the camera on relevant heroes.

The program for issuing these commands read from a file containing where the camera should be at different times of the game. We essentially translated these commands into mouse movements that would click and move the mouse as a replay was being watched.

Figure 4.8 shows this data might look like. Note that the time is given in minutes. In this case, the data can either be coordinates or a follow command, where the camera will center on the specified hero.

```
[
  {
    "type": "coordinates",
    "x": -2883.0,
    "y": 3822.5,
    "time": 3.55
  },
  {
    "type": "follow",
    "hero_index": 0,
    "time": 4.23
  }
]
```

Figure 4.8: Example camera movement data

To actually test this on a game replay, we would simply launch spectator mode and set the game time to the start of the game replay. Then we would need to run both the replay and the script at approximately the same time. Unfortunately, this synchronization issue was a limitation. In the future, it would be easier if we could control the camera programmatically by sending a stream of camera coordinates. When testing our camera control algorithm on the game *Orange vs. Natus Vincere, The International 3, Game 1*, we found that the camera control performed decently. However, due to how we controlled the camera, movements were sometimes not very smooth.

Chapter 5

Empirical Results

We also tested how well our system worked by performing a research study using the Amazon Mechanical Turk platform. This was important because we wanted to determine whether our camera control algorithm was preferred to the default directed camera used in *Dota 2*.

5.1 Experiment Details

Our study consisted of showing participants two videos of the same match (*Orange vs. Natus Vincere, The International 3, Game 1*, used previously), one of which used our intelligent camera, and the other of which used the directed camera. The participants then answered questions about the game and which video was perceptively better. To reduce bias, participants did not know which video corresponded to which, and the videos were shown in random order. The choice is denoted by a "Special" field in our results: this indicates which video used our intelligent solution.

5.1.1 Questions

To ensure high-quality results, we asked a few background questions. These questions helped filter out participants who did not watch the videos properly, as well as helped us gauge participants' experience levels.

We also asked participants to give positive and negative aspects of both videos, as well as tell us which video was better in various categories. For example, we asked participants to tell us which video was better in three categories: entertaining, generally informative, or informative about important events. For the full question details, please refer to Appendix E

5.2 Results

In total, we were able to sample a total of 13 participants. We found that all participants were able to answer our filtering questions correctly (Q1 and Q2). Thus, we believe that most of the participants provided high-quality answers to our questions.

As for our participants' *Dota 2* knowledge level, we found that nine of the participants never played *Dota 2*. Three participants were beginners, and one participant was of intermediate knowledge level.

In general, our experimental results were quite negative, with most participants preferring the default video. Seven participants preferred the default video in all three categories. Two participants felt that both videos were equally informative generally and about important events, and that the default video was more entertaining. No participants preferred our camera completely.

However, there were some participants who somewhat preferred our video. One participant felt that our video was more informative in general and about important events, but that both videos were equally entertaining. Yet another participant felt that our video was more informative in general and about important events, but that the default video was more entertaining. Specific details about the other questions are given in the following subsections.

One participant had mixed feelings: he felt that our video was more informative about important events, but that the default video was more entertaining. He said that both videos were equally generally informative.

5.2.1 Intelligent Camera Control

Unfortunately, most participants found that our video was not very smooth and sometimes jumped too much between heroes in the game. Many participants also thought it was not clear and sometimes missed some important events; these events were not missed in the default video. This could be attributed to the machine learning algorithm not performing as well as we thought it would. In addition, the camera jumping too much could be fixed by increasing the interval spent on each hero.

We were not able to make the camera control as smooth as we would have liked. However, this was because of technical limitations out of our control. Due to how we control the camera (simulating mouse events), this could not be avoided.

5.2.2 Directed Camera Control

Most participants liked the default video that used the directed camera control. Although this was not the result we would have liked, this made sense given that this is the built-in camera used in *Dota 2*. The directed camera is clearly more complex than we originally thought.

There were not too many negative comments, although some players that did not play the game did not understand what was happening initially. However, this opinion was also reflected in the comments on our intelligent video.

Chapter 6

Closing Remarks

6.1 Conclusion

We have described an intelligent camera control system that potentially performs much better than systems that use simple heuristics such as unit activity. By combining machine learning techniques and heuristics, it is theoretically possible to create a system that would perform much better than traditional camera control systems. Many of the techniques used are novel and potentially very valuable in the gaming industry, since most artificial intelligence in games commonly employ heuristic-based logic instead of machine learning. However, our practical experience as well as results showed otherwise: much of our camera movement was not very smooth as a result of having to simulate mouse click and drag events.

In addition, it is also hard to determine what features characterize our different game events well. Our feature set was very simple and would likely only work well with the simple and general types of events we identified. It would certainly be hard for a machine learning algorithm to classify an extremely complicated event (such as a hero evading attacks from other heroes) since those are unpredictable and rare events.

Despite this, much of the work done in this thesis is very interesting indeed. It helped us truly understand the benefits and challenges of trying to detect game events in *Dota 2* and games in general. We hope that this research will be useful for the future.

6.2 Future Work

There are many potential areas for future study. For example, we simplified much of the replay data and only explored a few features. One could also consider other features that take into account other portions of the replay data: what specific types of abilities are used, what heroes are being played, etc. This could help separate events at a finer granularity. For example, some heroes have more impact in a game simply because of the abilities they possess.

With regards to the machine learning component of our research, we can also explore using different machine learning algorithms others than *k*-Nearest Neighbors. Since we have high dimensional data, the *k*-Nearest Neighbors algorithm might suffer from very sparse data (the curse of dimensionality). We could also increase the number of events in our training set of data in order to produce more accurate results.

In addition, we think there can be large improvements made to how the camera is controlled. For example, most participants in our research study thought that our camera control jumped too much between heroes, and as a result the camera was not very easy to follow. In the future, we should move smoothly between heroes whenever possible.

Lastly, although our research only focused on *Dota 2*, it can theoretically be applied to any game with spectator mode. However, we believe that it will likely function best in two-dimensional top-down environments, since camera control is the easiest. Ideally, we would also like to use such a system in live matches. However, there may be performance issues to consider, so this would require using a faster programming language than Python, such as Java or C++. Fortunately, live game streams such as those in *Dota 2* are usually delayed a few minutes (in order to prevent cheating), which complements well to our intelligent camera system.

Bibliography

Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera control in computer graphics. *Computer Graphics Forum*, 27(8), 2008.

Steven M. Drucker and David Zeltzer. Intelligent camera control in a virtual environment. In *In Proceedings of Graphics Interface '94*, pages 190–199, 1994.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. Rabin. *AI Game Programming Wisdom 2*. Number v. 2 in AI Game Programming Wisdom. Charles River Media, 2004. ISBN 9781584502890.

Appendix A

Source Code

All source code for our project can be found on our GitHub repository, located at `http://github.com/dphang/sage`. Please refer to the repository for documentation on any dependencies required, as well as how to use our prototype.

Appendix B

Approval Letter

Here we attach the Approval Letter for our research study.

Office of Research and
Sponsored Programs
526 Brodhead Avenue
Bethlehem, PA 18015-3046
(610) 758-3021 Fax (610) 758-5994
http://www.lehigh.edu/~inors

DATE:	March 17, 2014
TO:	Hector Munoz-Avila, Ph.D. (Submitted by: Daniel Phang)
FROM:	Lehigh University Institutional Review Board
STUDY TITLE:	[576597-2] Automated Camera Control in Game Replays
IRB REFERENCE #:	14/150
SUBMISSION TYPE:	New Project
ACTION:	DETERMINATION OF EXEMPT STATUS
DECISION DATE:	3/17/14

The reviewer requests that you MUST remove the following sentence from the "Statement of Consent" section of the consent form: "I have had the opportunity to ask questions and have my questions answered." The letter said it had been removed, but the consent form draft uploaded still had the sentence.

Thank you for your submission of materials for this research study. The IRB has approved this project in the EXEMPT category (see below) so that the project is exempt from continuing IRB review according to federal regulations.

EXEMPT CATEGORY #2:

Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless:

i. Information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects;

AND

ii. any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

In the future, if significant changes are made to the study procedures (recruitment, instruments, informed consent process, etc.), a modification/amendment package should be submitted via IRBNet.org for review.

A copy of this correspondence will be on file in IRBNet.

If you have any questions, please contact Susan E. Disidore at (610)758-3020 (email: sus5@lehigh.edu) or Troy Boni at (610)758-2985 (email: tdb308@lehigh.edu). Please include your study title and reference number in all correspondence with this office.

Appendix C

Consent Form

Here we attach the Consent Form used in our research study.

LEHIGH UNIVERSITY

CONSENT FORM

Automated Camera Control in Games

You are invited to be in a research study of Automated Camera Control in Games. We ask that you read this form before agreeing to be in the study. **Your participation in this study is entirely VOLUNTARY.**

Background Information

The purpose of this study is to study automated camera control in games. Specifically our study focuses on the Defense of the Ancients 2 (DOTA 2) game. This is a game where two teams compete to defeat the one another. Each team consists of human players (called heroes in the game) and non-player characters. The team that destroys a special building, called "the ancient", from the opponent's team, wins the game. If you have not played DOTA 2 before, don't worry; we will show you a short video explaining this simple game.

You will be shown two videos of a DOTA 2 game. One uses a default camera control while the other one uses our new machine learning algorithm for camera control. You will not know which of the videos you are watching. So we want your honest opinion when asked to compare the two videos you saw. Your responses will help us understand capabilities and limitations of our algorithm.

DOTA 2 is a very popular game followed by thousands of people around the world. We hope you will enjoy watching these videos while helping us out.

Procedures

If you agree to be in this study, we would ask you to do the following things:
Before the experiment, you will be surveyed for prior knowledge of Defense of the Ancients (DOTA) game, and video games in general. In addition, you will be shown a short 5 minute game explaining the rules of the game. Afterwards, you will observe two videos. After observing these videos, you will be answering a questionnaire about the videos. The total time of this experiment will be approximately one hour total.

Risks and Benefits of being in the Study

The study has several risks:
As with watching videos, there is a risk of stress, fatigue, or hyperventilation which could be the onset of seizures. While photosensitivity and epilepsy are rare, if you experience any of the above symptoms, **please stop the study immediately**.

The benefits to participation are:
While participation in this study will not directly benefit you, the knowledge gained from this study will enable us to evaluate the effectiveness of algorithms for automated camera control.

Compensation

You will be given $3 compensation.

Confidentiality

Amazon Mechanical Turk handles your confidentiality.

Contacts and Questions

If you have any questions or comments contact: cameraControlStudy14@cse.lehigh.edu

Questions or Concerns:
If you have any questions or concerns regarding this study and would like to talk to someone other than the researcher(s), **you are encouraged** to contact to Susan Disidore or Troy Boni (email: inors@lehigh.edu) of Lehigh University's Office of Research and Sponsored Programs. All reports or correspondence will be kept confidential.

Appendix D

Questionnaire

Here we attach the Questionnaire used in our research study.

LEHIGH UNIVERSITY

QUESTIONNAIRE

Automated Camera Control in Games

Questions about game

- Q1. Which team won the game?
- Q2. What was an interesting event in the game?
- Q3. Rate your Dota 2 knowledge level.
  - Never Played ___ Beginner ___ Intermediate ___ Expert ___

Questions comparing videos

- Q4. Indicate positive aspect(s) about video 1.
- Q5. Indicate negative aspect(s) about video 1.
- Q6. Indicate positive aspect(s) about video 2.
- Q7. Indicate negative aspect(s) about video 2.
- Q8. Did you ever get confused while watching video 1? Explain if so.
- Q9. Did you ever get confused while watching video 2? Explain if so.
- Q10. Which video you find more informative about important events in the game?
  - Video 1 ___ Video 2 ___ About the same ___
- Q11. Which video you find generally more informative about the game?
  - Video 1 ___ Video 2 ___ About the same ___
- Q12. Which video you find more entertaining to watch?
  - Video 1 ___ Video 2 ___ About the same ___

Thank you very much for participating in this experience. Your contribution will be invaluable in our research about automated camera control!

Appendix E

Results

Here we attach the results of our research study. Note that the question labels correspond to the labels on the questionnaire in Appendix D.

	A	B	C	D
1	Participant	Answer.Q1	Answer.Q2	Answer.Q3
2	1	Dire	Radiant's middle tower was destroyed.	Never played
3	2	Dire	about half way in the battle the dire found roshan and killed him, therefore gaining more power.	Never played
4	3	Dire	In the first video, the first confrontation on the river banks, and the narrative of I hope you like bloodbaths, great humor right from the beginning from the game narrator.	Never played
5	4	Dire	i saw that the dire were able to get the upper hand in destroying a couple of the radiant's towers, and got many kills in during the first half of the game, though the radiant had the advantage of having some kind of cloaking ability, though it didn't seem to be put to good use.	Never played

	A	B	C	D
6	5	Dire	Radiant tower is attached and the Dire is dominating from the beginning and they got the victory	Intermediate
7	6	Dire	Radiant fortified their structures and there was a mini battle between the two groups near the radiant base. Titan.Ohy.Razer then came out of nowhere and got a double kill	Never played
8	7	Dire	Radiants bottom tower falls. Radiants bottom barracks get attacked and falls. Dires win. Radiants ancient falls in to pieces.	Never played
9	8	Dire	NeoES-X drew first blood on Puppey and tried to get away but KuroKy was able to chase him down through the woods.	Beginner
10	9	Dire	Radiant's barracks was destroyed in a massive attack by the other side and a lot of bright neon aquamarine lights came out.. it reminded me of Independence Day.	Never played

	A	B	C	D
11	10	Dire	In the second video, I really enjoyed the beginning when I watched all the players move around the map to take up strategic positions. It was interesting to watch the choices of each player and try to guess what their strategy would be. I thought they did a really nice job of following each of the players and showing the setup.	Beginner
12	11	Dire	In the first video, around the 14 minute mark, it looks as though Dire attempted to attack Radiant's base. It looked like the Dire had to retreat before returning to finally win.	Never played

	A	B	C	D
13	12	Dire	In Video 2, Dire attacked the Radiant base to win the match. Radiant spotter tower went under siege.	Never played
14	13	Dire	I saw Orange dominate in the team fight near the end of the game. Both and Mushi had an Aegis and survived.	Beginner

	E	F
1	Answer.Q4	Answer.Q5
2	Most of the video is clear. Most of the video was kept on battle. Was able to see Dire's victory.	Parts of the video are blurry and pixelated.
3	good sound	didn't notice any
4	First off for your knowledge... Im not a gamer, havent played a video game of any sort in over 20 years, but it did not state gamer requirements for this HIT... That said, the video is graphically great, but for me it was hard to follow, but I think if I was a gamer following would be much easier. The visual aspect, sound effects, and overall camera ability to follow was quite impressive!	I did find myself lost in many areas, even after watching the tutorial... Basically on the movement from game character to game character and my ability to actually follow what exactly was happening at times.
5	the radiant's cloaking ability seemed like something positive	the radiant team wasn't able to defend themselves enough during the first half.

	E	F
6	The positive aspects are the sounds present through the video and the intimations given. The creatures present in the video and the attacking method. The visualization is good.	The opponents are differentiate properly and cannot differentiate the images are too small and the color variation is not good
7	There was pretty much non stop fighting with lots of mini scale battles.	Although I liked the battles some were so clustered and had so much going on, I couldn't really tell what was going on. There would be some where I would see magic and fire, but had not idea who shot it
8	The main action is always at the center of the screen.	A bit uptight and mechanical.
9	A positive aspect about video 1 is that it was able to keep up with most of the action and I felt I did not miss any important action in the game.	The VOD lacked commentary which is important to me because I am a beginner. I think it would help if there was someone narrating and explaining all the action going on.
10	That multiple ways to attack were available.. it wasn't just limited to swordplay, fire, etc.	At about the 11 minute mark, it seems the characters turned to dust and you couldn't see who was who.

	E	F
11	This video had very high audio quality. I was impressed with the sound effects and the realism that it lent to the video. The video quality itself was good.	The video seemed to randomly jump around from one thing to another, and often didn't seem to correlate with what was being said on the screen from time to time (for instance, it would say one of Radiant's towers was under attack, but the action on the screen did not seem to match what had been said. I personally find one thing that's lacking is active commentary about the action. For novices to the game, it's sort of confusing as to what's happening and why.
12	The video showed a lot of great battle footage. I liked the in-game announcer's comments.	Some of the fighting became repetitive. Cutting to different characters and fights that are going on at the same time could be confusing.

	E	F
13	I really like how the camera follows certain individuals in the game, or that it allows you to choose who you want to follow.	The angle of the camera in these games is amazing, it gives you just the right angle to see where all your characters are at, while maintaining a decent aspect ratio to give you a good view of the playing field.
14	It followed the action properly and I feel I didn't miss any of the action.	I think some commentary would of helped make the game more exciting.

	G	H
1	Answer.Q6	Answer.Q7
2	The entire video was clear.	around and would often switch just as a battle was about to begin. I kept hearing a sound like the person recording was getting notifications on facebook. Spent too much time on players just walking around.
3	better sound it seemed, the writing on the right was white	none
4	As I stated about video 1, the visual aspects are fantastic and the graphics, sounds, narrative are great.	I dont really know what to say here that is negative. As with the first video, I was still learning how to exactly follow each character and how the game controllers would work. That was more my curiousity. When it comes to camera angle, it just seems to always be more of an overhead view, when it would be great to see a eye to eye view, be able to scan the horizon and see what is coming at an eye level instead of always from above...
5	the dire seemed to be doing better than they were in the first video, getting more kills early on in the video.	the radiant team seemed to have stronger attacks at their disposal than they did earlier in the game.

	G	H
6	The domination aspects of the Dire are very interesting. Their attacking methodology are very expressive.	The graphics are not clear and cannot differentiate between Radiant and Dire. The fires are not know that who is firing to whom.
7	The women commenting on the match. I didn't really notice her in the first video but she had a few funny lines in the second video.	Some on the camera transitions were weird. There was one around like 6 minutes in that made me think my computer froze.
8	It is more natural and camera work is action oriented.	The main action is not always at the center of the screen.
9	Video 2 showed a different aspect of the game. I felt it concentrated more on the Radiant side than the Dire side and gave a different perspective.	Video 2â€™s camera seemed miss a lot of action. It would constantly be staring at the fog when there was obvious there was action elsewhere. I felt I miss a part of the game because of it. Again commentary would have been very helpful.
10	Great graphics with tons of colors and maybe it's just me but it seemed like you got a closer view of what was going on.	Kind of hard to tell what each individual character looks like.

	G	H
11	It seemed like the transitions between the characters and the action were a bit smoother in this video, when compared to the first video. I felt like the jumps made more sense in comparison to the on-screen action. I felt like I was following more action in this video as well rather than seeing random stuff. This video does a better job than the first video at making it feel like there is some sort of commentary going on. The camera seemed to follow characters more from the side or rear rather than from the front, which was a welcome change and made everything feel like it flowed more smoothly.	For some reason the sound doesn't feel as good in this video. The music almost feels like it's overpowering the sound effects and the audio. Despite following the action more closely, it doesn't feel like the video really "tells a story" so to speak, which is important. If I'm going to watch a video of a match, I like there to be some kind of narrative with it. However, Video 2 cut off before the end of the match, which was a big negative.
12	This video also showed a lot of nice battle footage. It's also entertaining.	As with the other video, the fighting could be seen as a bit repetitive. Someone who is not familiar with the game-play could see the camera switching as confusing. Video 2 didn't show who won the game, but I assume it was the Dire.

	G	H
13	I dont like how its kind of like a spotlight on wherever you are looking, like where the black edges meet the center. Makes it hard to see.	It confuses me when the video cuts randomly to another character in the game. Maybe make the characters more distinguished in color.
14	It seemed to show other actions about the game.	The camera was all over the place and I missed some actions because the camera would not move sometimes.

	I	J	K
1	Answer.Q8	Answer.Q9	Answer.Q10
2	No	No	Video 1
3	at first , till I understood which team is which	nope, it was somehow clearer than video one.	Video 2
4	Yes, since Im not a gamer at all, it took a lot for me to try to get used to the character jumping. Which character was in action, whose character was whose, etc...	By video 2 I was getting better as distinguishing between characters, but I was still finding myself lost at times. The overhead view is too much, and a different angle would makes things much more visually easier to follow. Atleast for someone like me...	Same
5	it was sometimes hard to what character was on which team at points.	nothing really confusing about the second video, other than maybe keeping track of who is who.	Video 1

	I	J	K
6	No	No	Video 1
7	Yes some of the battle had some much going on I couldn't tell who was attacking and who was dying.	Not really	Same
8	No.	No.	Video 1
9	I did not really get confused watching video 1 other than not knowing some of the spells but that is expected with a beginner.	Video 2, had different names for some reason. I donâ€™t know if that is intentional or not but that was confusing at first.	Video 1
10	Yes.. later in the video, it seemed like all the characters blended together and you couldn't really quite get what was going on.	Not really no.	Video 2

	I	J	K
11	Yes, I did get confused while watching this. I spent a lot of time trying to figure out who was Dire and who was Radiant because nowhere on the game screen did those words show up until the very end, when it said who won and who lost and showed all the relevant game and player statistics. The camera control seemed much more erratic here and seemed to follow players from the front more than the back.	I did. While I was more familiar with the gameplay during the second video and more comfortable making assessments about what was going on, I still found the overall lack of narration somewhat confusing. I'm fairly familiar with gameplay videos in general - I used to play World of Warcraft and often engaged in battlegrounds which had similar layouts and setups to this. I understand how confusing in general gameplay videos can be. Some of the best WoW videos I watched had more narration. That being said, the camera control on video two was definitely superior, and made it easier to follow the action overall. I just feel there are improvements that could be made across the board that would make such gameplay videos compelling to players or gamers who aren't as familiar with DOTA2 as well.	Video 2
12	Switching between characters and such was a bit confusing at first. It became easier to follow when I became more familiar with what was going on.	No. After watching the first video and becoming familiar with the game-play, the second video is easier to follow.	Same

	I	J	K
13	Im not too familiar with the game itself, so I know what the games about but I dont understand the characters and who is killing who. It gets very confusing if you have never played.	Im not too familiar with the game itself, so I know what the games about but I dont understand the characters and who is killing who. It gets very confusing if you have never played.	Video 2
14	Just some parts because I don't know all of the heroes abilities. If there was commentary I think it would help a lot.	Yes, even though it was the same game, the names were different on the heroes and that confused me at first.	Video 1

	L	M	N
1	Answer.Q11	Answer.Q12	Answer.Special
2	Video 1	Video 1	video 2
3	Video 2	Video 2	video 1
4	Same	Video 2	video 1
5	Same	Video 2	video 1

	L	M	N
6	Video 1	Video 1	video 2
7	Same	Video 2	video 1
8	Video 1	Video 2	video 1
9	Video 1	Video 1	video 2
10	Video 2	Video 2	video 1

	L	M	N
11	Video 2	Same	video 2
12	Same	Same	video 2

	L	M	N
13	Video 2	Video 2	video 1
14	Video 1	Video 1	video 2

Biography

Daniel Wei-Shen Phang was born November 12, 1991 in Kuala Lumpur, Malaysia, to Yew Fook Phang and Li Kheng Looi. He graduated with a B.S in Computer Engineering with High Honors at Lehigh University in 2013. He has also published a paper titled "POETIC: Interactive solutions to alleviate the reversal error in student-professor type problems" in the *International Journal of Human-Computer Studies* in 2014.