IN4086 Data Visualization
Final Project
“Teamfight Classification and Visualization in Defense of the Ancients 2 (DotA2)”

Tung Phan, ttphan, 4004868 Kevin van Nes, kjmvannes, 4020871

February 10, 2015

Introduction

The following report will discuss the application that was created as part of the final assignment of the IN4086 Data Visualization course. In this report we will explain why this application was created and its functionalities in terms of visualization and providing information to the user will be elaborated upon. The application can be found at http://www.techteach.nl/dota2.

The main role of the application is to extract so-called ‘teamfights’ out of data from the Multiplayer Online Battle Arena game called ‘Defense of the Ancients 2’ or DotA2. The goal is to extract and visualize these teamfights and to give users information about these teamfights in many different, but clear ways.

Preliminary work for this application was done during the first assignment for the Data Visualization course. As a result of this assignment, we had gone through the Data Processing steps, with which we were able to set up a basic application which itself provided a basic visualization of the data and corresponding information. The process behind this preliminary work will be described later on in this report.

Our goals for this project were to improve the ways in which information was visualized and provided to the users of the application. We wanted to improve the application so that it would visualize the match data in such a way that both skilled and unskilled players are able to retrieve the information they seek to find. We also found it important to improve the amount of information that was provided with the given visualization and we wanted to create more extensive visualization and options for users, instead of just having the map of the DotA2 arena.

The outline of the report is as follows. Firstly, in section 1 (“Data Processing”), we will give a definition of how we interpreted teamfights. This will be done to avoid ambiguities in understanding our application and the rest of the report. Furthermore, we will describe what the application looked like and what it could do prior to the work that was done as part of the final assignment. After that, in section 2 (“Manners of Visualization”), the additions and improvements that were made to the application will be expanded upon. Finally, a short conclusion will be given in section 3, in which we will discuss whether all goals were reached and what future work is planned to further improve the application.

1 Data Processing

The data visualization pipeline was followed during the preliminary work that was done as part of the first assignment. While improving the application for the final project, this pipeline was reiterated and improvements were made wherever we thought it was necessary or desirable. Of course, the Data Acquisition part of the pipeline was already done for us, as the files containing the match data were handed to us prior to making the assignment. We will firstly define how we used the word ‘teamfight’ while creating the application. Furthermore, each of the steps of the data visualization pipeline (filtering, mapping and rendering) and their applications to our product will be discussed.

1.1 Our Definition of Teamfight

To make sure that it is clear how we defined 'teamfight' while designing and developing our application, we will define this in the upcoming paragraph. Teamfights are critical to the game, because teamfights with all members involved are events with very high stakes. When one team has more team members alive than the other team after having fought a teamfight, this team can use this situation to their advantage by completing objectives or even finishing the game. They can do this while the dead members of the team that lost the teamfight will have to either buy themselves back into the game with their hard earned gold or wait for the respawn timer. Completing objectives will yield gold and territorial advantages to the winning team, which will cause a snowball effect throughout the game. In other words, teamfights involving all members of both teams determine the flow and outcome of the game.

We define a teamfight as a period of time in which all players of both teams (Team Radiant and Team Dire) are close to their teammates and their enemies. Simply stated, this means that all ten players are near each other, which means that the chances of them fighting each other is very likely. This makes the period of time during which this happens a teamfight. We limit the teamfights to fully balanced teamfights, which means that the teamfight should involve all ten players. The moment one member is either killed or has retreated, the advantage is to the other team and we will not consider it a fully balanced teamfight anymore. That also means that small skirmishes of less than 10 people are not considered to be teamfights, even if they are evenly balanced. This is to ensure that we only capture the teamfights where both teams go 'all in', which are the defining fights of a match.

1.2 Data Filtering

As was stated in the previous paragraph, a teamfight is a period of time during which all players in the game are fighting. We had come up with a way to extract these moments from the raw data files. Firstly, we wrote a simple script in Java that cut up the 'master-distance.csv' and 'master-zones.csv' files into separate files for each match (e.g. match-<match number>-distance.csv and match-<match number>-zones.csv). We were now able to extract the necessary variables to find teamfights out of each match.

From the -zones files, we used the following variables: the x and y positions, the team the player belongs to, which team won and lost (WinLose) and the tsync variable (the current match time). From the -distance files, we used the same variables with the addition of the 'DD' variable. The 'DD' variable estimates the distance between members of each team, so for Team Dire, this variable calculates the distance between the five members of Team Dire, and vice versa for Team Radiant.

For each match, we calculated for each tsync (= second) whether the DD variable was lower than a certain threshold. If it was, that time moment was added to an array along with its corresponding DD. We did this for both teams, resulting in two arrays containing low DD time moments for the winning team and the losing team, which we called winSmallDDArray and loseSmallDDArray, respectively. By intersecting these arrays, the smallDDArray was created, in which moments in time are contained where the DD is low for both teams at the same time. Such a moment means that players from both teams are close to their teammates. The only thing that remains to have a teamfight, is that both teams have to be close to each other at such a moment.

To calculate whether this is the fact, we also calculated for each entry of smallDDArray the x and y positions of each of the ten players, so we also knew where a teamfight took place on top of when it took place. These positions were put into corresponding arrays too, which were again split up into a 'winning team' and 'losing team' version. Out of these positions, we were able to calculate the average positions of each team (i.e. the positions of all five players of a team, summed up and then averaged), with which we were now able to compute the Euclidean distance between these two averages. If this distance is lower than a certain threshold (we set this to 30, as we calculated this was a reasonable distance), a teamfight is found.

With the above described procedure, we were able to find teamfights very accurately. We checked whether our estimations were right by downloading some of the replay videos corresponding to the match numbers and checking whether a teamfight actually took place during the timespans that we estimated and it turned out we were able to find teamfights really accurately. This meant that we could now start on mapping the data to renderable data.

1.3 Data Mapping

As was stated above, the teamfights that we were now able to find needed to be mapped to renderable data. For this purpose, we decided to put each teamfight in a corresponding cluster object, which contains useful properties of the corresponding teamfight. The most important properties for rendering the data were the average x and y positions and the lowest and highest x and y positions of the teamfight. The average x and y positions are simply calculated by summing all of the x and y positions of the teamfight and averaging them. The lowest and highest x and y positions were computed by taking the lowest/highest x/y position that was fought on.

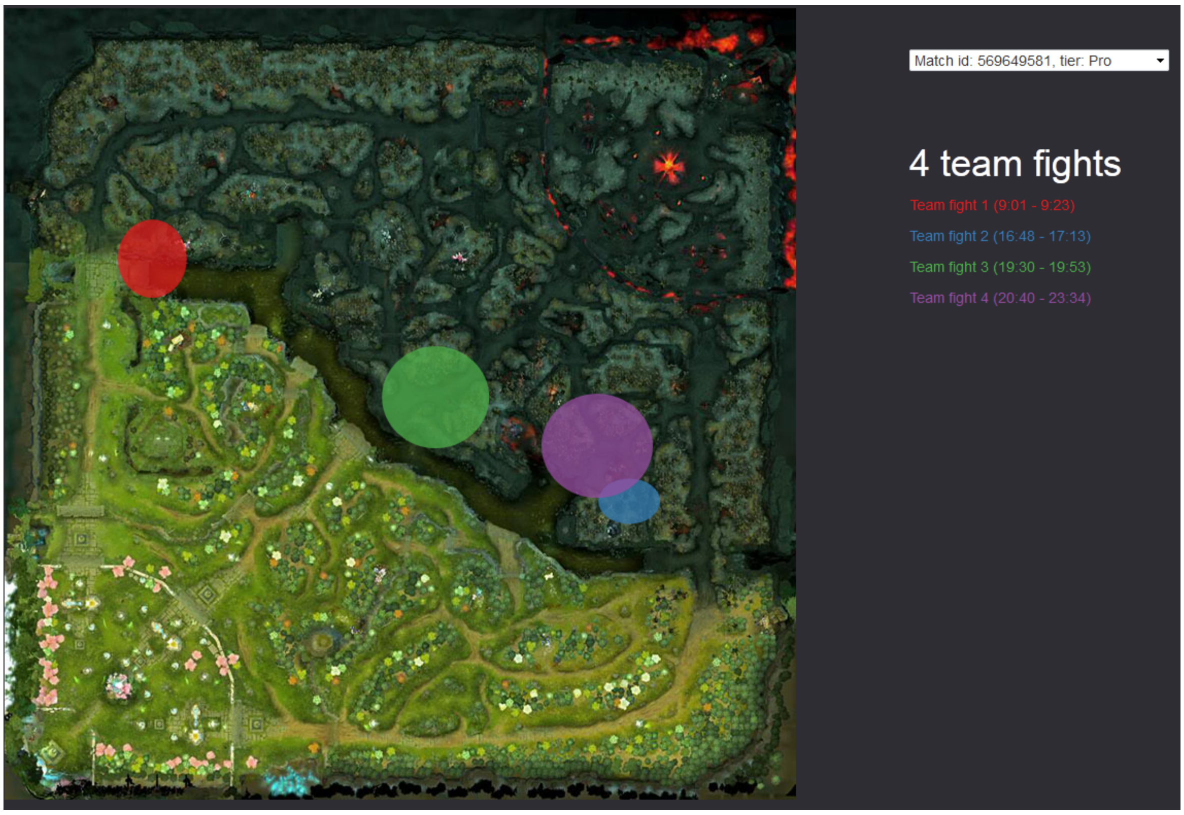
These values could be mapped to create ellipses, which visualize the area of a teamfight in a visually clear and computationally efficient way. The average x and y positions of the clusters (and thus the teamfights) are used to decide where the center points of the ellipses need to be drawn, while the lowest and highest x and y values of the teamfight are used to compute the x and y radiuses of the ellipses. These ellipses will - when rendered - visualize the main part of each teamfight of a match.

Since teamfights are usually fought in areas that are not easily visualized by geometry, we had chosen to use ellipses, as these forms would give the most smooth results when visualizing a teamfight. If we had for example used a rectangle, the result would be too rough. We also experimented with a different kind of visualization (i.e. drawing small circles/dots for each position a teamfight had taken place in). This type of visualization was, even though it showed the area of teamfights more accurately, less optimal, as poor results were obtained in case a lot of teamfights were fought in a match. The user of the application would in these cases quickly lose the overview and the visualization would quickly become polluted and unclear, which is something we had avidly tried to avoid.

1.4 Data Rendering

Now that we mapped the data to the ellipses, rendering was an easy step, as we used the D3 library [1] to draw the ellipses easily using the computed variables and properties of the teamfights. The colors of the ellipses are determined using a combination of three qualitative contrasting color scales. These color scales are extracted from the ColorBrewer application [2]. These scales were used to make sure that all ellipses contrasted as much as possible so that the visualization is not unclear and does not contain ambiguities. The ellipses are also drawn with slight transparency, this is done to accommodate for (complete) overlaps when two teamfights occur in the same area.

To get an idea of what the application looked like after this preliminary work, see Figure 1.



Match id: 569649581, tier: Pro

4 team fights

Team fight 1 (9:01 - 9:23)

Team fight 2 (16:48 - 17:13)

Team fight 3 (19:30 - 19:53)

Team fight 4 (20:40 - 23:34)

Figure 1: An overview of the old application

A new challenge arose when we decided to extend the application in such a way that the teamfights of two matches could be compared, instead of just showing the teamfights of one match. This meant that colors would have to be different for teamfights of different matches and that the shapes would have to differ too. More about this newly arised challenge will be discussed in section 2.1.

2 Manners of Visualization

The previous section described in detail how the application worked after the preliminary work. From this point on, we wanted to improve and extend the application as part of the Data Visualization final project. Our main goals were to extend the amount of visualization that is done by the application and to enlarge the amount of information that can be retrieved from the data for both skilled and unskilled DotA2 players.
To keep the explanation as clear as possible, we have split up the application into three parts, which will be discussed separately below. Of course, any coherency will also be expanded upon. The three parts we have split up our application into are the following: the map, the menu and the timeline and brush.

2.1 Map

The image in the upper left part of our application is called the map. This image represents a top-down view of the arena in which DotA2 matches are played. The map is also the part of our application where the ellipses representing the teamfights are drawn. As can be seen in Figure 1, the application was merely able to show ellipses and their timespans for one match, without any additional information.

Our first goal for enhancing this part of the application was to add the ability to compare two matches with each other, while still maintaining a clear visualization of the teamfights of both matches.
After having created the menu to add two teamfights - which will be elaborated in section 2.2 - we needed to think about how to avoid causing confusion when visualizing the teamfights of two matches together. We wanted to use the same elliptic shapes to visualize teamfights of a second match (for the reasons described in section 1.3), but differences had to be made to clearly visualize which ellipse belongs to which match. The first solution we applied to do so was to use different colors and textures for the ellipses of the second match compared to those of the first match. The ellipses of the second match were given colors from the end of the array containing the qualitative contrasting color scale, as to make sure that reoccurring colors are a rare occurrence. Furthermore, the second match ellipses are shown as the perimeters of an ellipse (so only an elliptic line), instead of a filled ellipse. This was chosen as a suitable solution, because the filled ellipses of one match contrast the elliptic lines of another very well. The colors are also set to be contrasting as much as possible, meaning that confusion will probably not occur very quickly. Furthermore, the elliptic lines will not hide (i.e. overlap) other teamfights, because one can see through the unfilled area in the middle.

Another goal we had, was to give users more information about each of the teamfights. This information was retrieved from the same data as was used to draw the ellipses. To give this information to the user in a clear fashion, we used the Tipsy library [3] to develop tooltips that appear whenever a user hovers over one of the ellipses. These tooltips contain information about the main areas the teamfight was fought in and about its duration. Whenever a user hovers over an ellipse, the other ellipses are turned gray and the text in the menu on the right corresponding to the ellipse is enlarged. This is done to emphasize to the user which teamfight he/she is looking at, so he/she can for example decide to hide this ellipse from the view using the menu.

More emphasis on the connection between ellipses and the menu on the right is made due to the fact that the colors of the ellipses of both matches correspond to the colors used to color the texts in the menu on the right. The colors are also connected to those of the rectangles shown on the timeline, which is discussed in section 2.3.

2.2 Menu

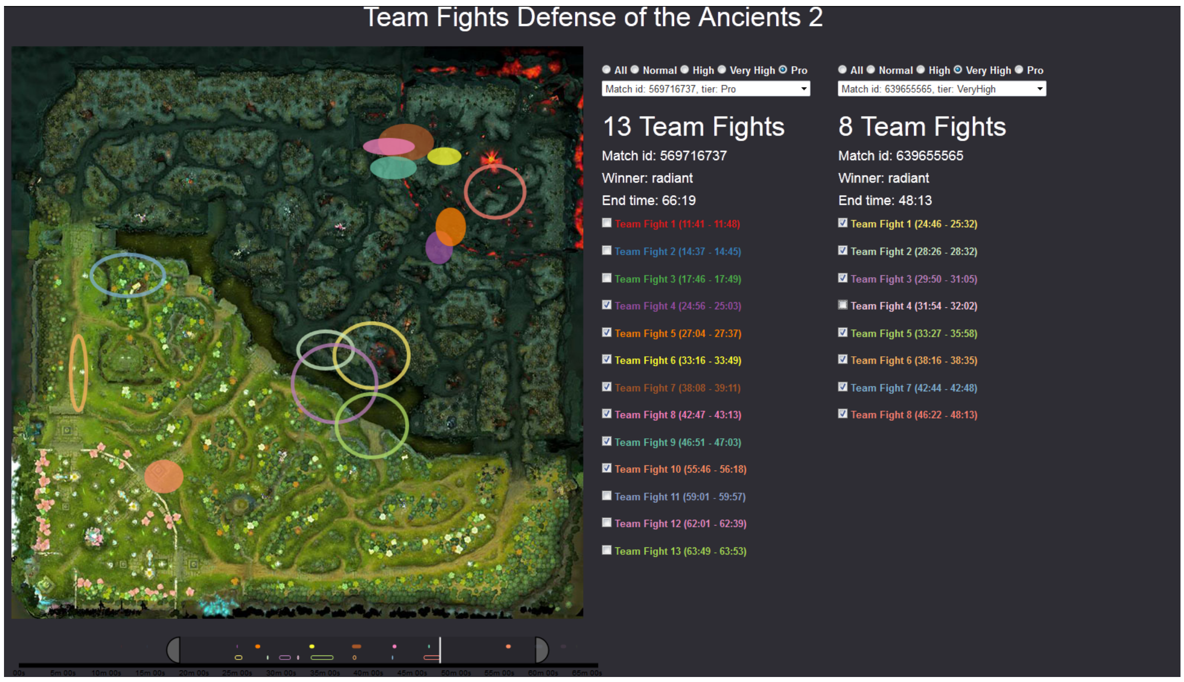
Since we want to offer users of the application as much freedom of choice as possible when it comes to what and how they want to visualize teamfights, we have enhanced the menu to the right of the map as a part of reaching this goal. As was discussed in the previous section, we have added a second dropdown menu to allow users to compare the teamfights of two different matches. We understood that users might not want to scroll through the whole list of matches every time they are looking for a match of a specific tier. Therefore, we have made it possible to filter matches by player skill tier, making it easier for users to make specific comparisons, e.g. comparing two pro matches, or a pro and a normal match. It is also possible to hide and show specific teamfights. Whenever a teamfight is hidden, the corresponding ellipse on the map and rectangle on the timeline are also hidden.

Furthermore, we made sure to stay consistent with the visualization that happens on the map, which means that we also added the functionality of tooltips appearing when a user hovers over one of the teamfight texts. The ellipse corresponding to the text that is hovered over in the menu will be highlighted on the map.

2.3 Timeline and Brush

One of the last goals we had stated in our project proposal was the addition of the possibility to 'slide' through games. We wanted to add this to the application to give users even more options for visualizing teamfights. To this end, we have implemented a timeline with a 'brush', which are also often used in parallel coordinate plots.
The timeline can be found below the map. On the timeline small rectangles are placed, which represent the teamfights and their durations. A longer rectangle represents a longer teamfight. The colors of these rectangles also correspond to those of the ellipses and menu texts. The timeline also visualizes the relative positions and lengths of teamfights of one or both matches. Furthermore, the white vertical line, which is shown on the timeline as soon as two matches are compared, represents the end of the shortest of the two matches. To make fully clear to the users which rectangle on the timeline is connected to which teamfight, we have also added tooltips that appear when hovering over such a rectangle. Furthermore, the other rectangles that aren't hovered over will fade, the ellipses on the map that do not correspond to this teamfight will turn gray and the corresponding text in the menu is emphasized.

The addition of the brush to the timeline gives users the ability to 'slide' through a game and filter specific moments of the game. This gives the users new visualization options, such as for example being able to see all teamfights that happened after the first fifteen minutes of a match have passed.


Team Fights Defense of the Ancients 2

All Normal High Very High [✓ Pro]

Match id: 569716737, tier: Pro

13 Team Fights

Match id: 569716737

Winner: radiant

End time: 66:19

- [ ] Team Fight 1 (11:41 - 11:48)
- [ ] Team Fight 2 (14:37 - 14:45)
- [ ] Team Fight 3 (17:46 - 17:49)
- [x] Team Fight 4 (24:56 - 25:03)
- [x] Team Fight 5 (27:04 - 27:37)
- [x] Team Fight 6 (33:16 - 33:49)
- [x] Team Fight 7 (38:08 - 39:11)
- [x] Team Fight 8 (42:47 - 43:13)
- [x] Team Fight 9 (46:51 - 47:03)
- [x] Team Fight 10 (55:46 - 56:18)
- [ ] Team Fight 11 (59:01 - 59:57)
- [ ] Team Fight 12 (62:01 - 62:39)
- [ ] Team Fight 13 (63:49 - 63:53)

All Normal High [✓ Very High] Pro

Match id: 639655565, tier: VeryHigh

8 Team Fights

Match id: 639655565

Winner: radiant

End time: 48:13

- [x] Team Fight 1 (24:46 - 25:32)
- [x] Team Fight 2 (28:26 - 28:32)
- [x] Team Fight 3 (29:50 - 31:05)
- [ ] Team Fight 4 (31:54 - 32:02)
- [x] Team Fight 5 (33:27 - 35:58)
- [x] Team Fight 6 (38:16 - 38:35)
- [x] Team Fight 7 (42:44 - 42:48)
- [x] Team Fight 8 (46:22 - 48:13)

Figure 2: The end result of our work: the application in its current state

3 End Result

An overview of what the application now looks like can be seen in Figure 2. Below, we will review our goals, discuss in what ways we want to work further on our application to improve it in the future and give a short conclusion.

3.1 Goals Review

We have reached most of our goals during this project. Our first goal was to make sure that both skilled and unskilled players can quickly retrieve information about the most interesting parts of a match, e.g. teamfights. We have done our best to offer a lot of possibilities and options that help the user, either skilled/unskilled, to visualize exactly what he/she wants to find out.
Another goal of ours was to add the ability to ‘slide’ through games. The addition of our timeline and brush has made this possible, which opened up a lot of new ways to visualize specific information about teamfights, such as their relative position to each other.
A goal that was constantly kept in mind was that all of our work should adhere to the principles of Data Visualization, which were taught to us during the course lectures. We have done our best to do this, so that our end result would become as good and clear as possible.

Lastly, a goal that was not fulfilled was the addition of visualization of other DotA2 information. In our proposal we stated that, if time allowed it, we would try and see if we could visualize other interesting features of DotA2 matches, such as fighting near towers or killing jungle creeps. We had taken time to think about how we would do this, but we found out that it would be very hard to accurately visualize these things using only the data that was given to us. More information would have needed to be available. However, we have enjoyed working on this application and visualizing this data very much, so we have decided to continue working on the application after this project has ended. We will talk about this in the next section.

3.2 Future Work

As stated above, we are still very enthusiastic about data visualization and also about its relation with game data in this case. We want to see if there are ways to get more information out of matches, so that we will still be able to visualize such things as were mentioned above (fighting near towers, killing jungle creeps, etc.).
Some of our more ambitious ideas include adding the option to include the replay videos of matches within the application. We would then give users the possibility of clicking a teamfight ellipse, which would then bring the replay video to the point in time where that specific teamfight started. This way, users will be able to watch specific teamfights without having to remember at which times these teamfights took place. They could simply look them up!

3.3 Conclusion

We are happy with the end result of our work. The application does what we wanted it to do and it does this in a smooth and clear way. We think it also conforms to the principles of Data Visualization as they were taught to us during the course lectures. The end result of our application can be found on http://www.techteach.nl/dota2

4 References

Bostock, M. (2013). Data-Driven Documents (Version 3.5.4) [Software]. Available from http://d3js.org/

Brewer, C., & Harrower, M. ColorBrewer (Version 2.0) [Software]. Available from http://www.colorbrewer2.org/

Frame, J. (2012). tipsy (Version 1.0.0a) [Software]. Available from http://onehackoranother.com/projects/jquery/tipsy/