Report: Berlin Marathon Data Visualization

Jula McGibbon
jula.mcgibbon@uni-weimar.de

1. Introduction

This visualization of Berlin Marathon time data intends to show the progress of runners over the years and also the increasing popularity of the Berlin Marathon which can be seen by the increasing number of nationalities in the rankings. All data was retrieved from the official Berlin Marathon homepage [1].

2. Overview functionalities

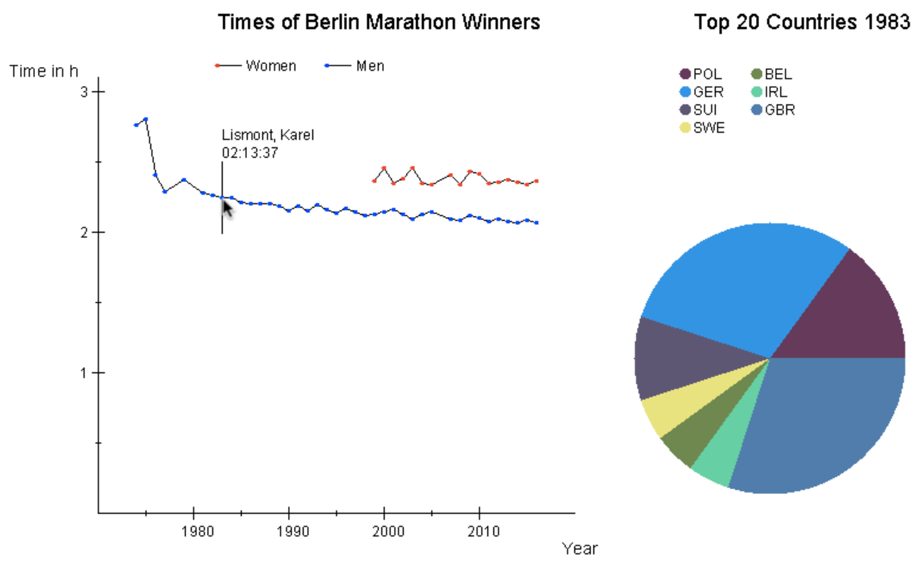

Figure 1: Overview

2.1. Coordinate System

When starting the visualization project a coordinate system will show up on windows the left side (see figure 1). Here one can see the running times of men and women runners who won in a certain year. If one clicks on a data point the corresponding name and the exact time in that year will appear. One can also distort the axes into both directions to see a smaller range, e.g. only the years from 1970 to 1990 and/or times between 2 and 3 hours. All data which is not included in those ranges will vanish (see figure 2).

2.2. Pie chart

Furthermore, when clicking on a data point, a pie chart with a corresponding legend will show up on the windows right side. It represents how often a certain nationality appears in the Top 20 runners in that year. By sliding over the pie slices with the mouse the runners names, their ranks and the country included in that slice will show up at the mouse position (see figure 2). When starting the visualization program, each country gets a randomly generated color assigned to itself, i.e. when Polands pie slice has the color green for one data point, it will also have the color green for all other data points.

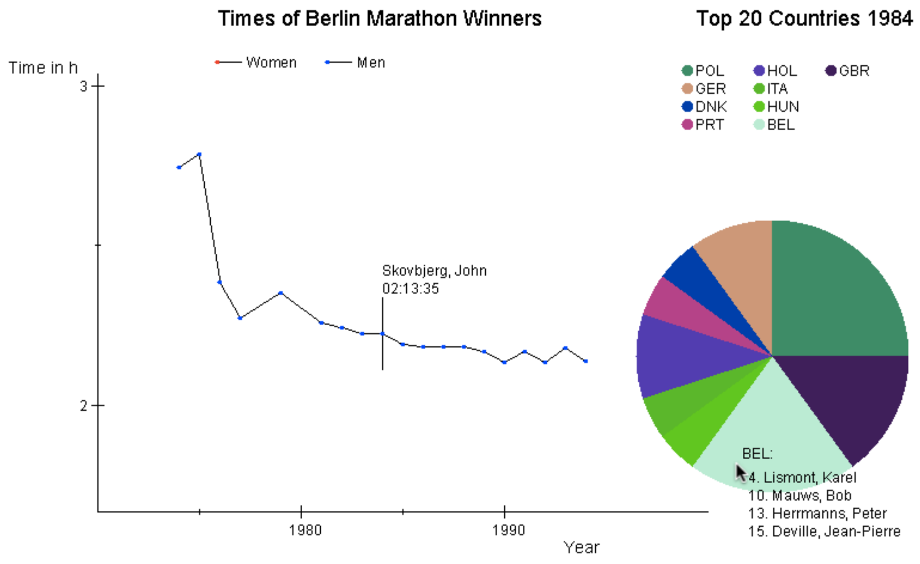

Figure 2: Distorted axes and pie slice information at the mouse position. There is no women data available in the year range 1970-1990.

3. Implemented java classes and their function

The implementation was done in the Java Eclipse IDE using the framework from the InfoVis lab class [2]. All data was converted into a ssv file using python.

3.1. Main program

The framework infovis.paracoods [2] was extended by three important classes: DrawAxes, DrawComponents and DrawPieChart. The paint method in View.java calls the methods drawXaxis(), drawYaxis() in class DrawAxes which are drawing the coordinate system axes and drawData() in class DrawComponents which draws the data points and their connecting lines into the graph. When the mouse is inside one drawn data point, the method drawPie() in class DrawPieChart is called.

```java
import java.awt.Graphics;
import javax.swing.JPanel;
import ...;

public class View extends JPanel {
   private Model model = null;
   private DrawComponents draw_data = new DrawComponents();
   private DrawAxes draw_axis = new DrawAxes();

    @Override
   public void paint(Graphics g) {

      // Storing data in vectors
      // Converting times
      // draw axes
      draw_axis.draw_axis.drawXaxis(g2D, ...);
      draw_axis.draw_axis.drawYaxis(g2D, ...);
      draw_data.drawData(g2D,...);
    }
}
```

```java
public class DrawComponents{

   private DrawPieChart draw_pie = new DrawPieChart();


   public void drawData(Graphics g2D,...){

   // check if axes were manipulated (diff<0 or diff>0)?

   if(mouse_inside_Area(MousePos,..){

      draw_pie.drawPie(g2D,...);


      }
   }
}
```

3.2. DrawAxes

The axes functions are checking if the mouse click was close to the drawn axes lines. When the mouse click was included, the difference between the start click position and the released position is measured. This difference is used to change the line difference between the x or y axes ranges.

3.3. DrawComponents

The function drawData transforms the data retrieved from the data file BM_results.ssv so it can be viewed in the java window coordinates system. It uses the function mouse_inside_Area() to check if a data point includes the mouse position. Is this the case, further information is drawn next to the data point and the drawPie() function is called. Data points are only drawn, when they are still lying in the allowed ranges.

3.4. DrawPie

The drawPie function calls first of all the SetCountryColor() function, so every country is assigned with a certain randomly generated color. Then countElems() is called to count how often a country appears in the required year which is needed for the size of the pie slices. The function g2D.fillArc() draws the pie chart based on the counted frequencies before and a corresponding arc object checks with the function contains() if the mouse is included in a pie slice. If this is the case, further information to that pie slice is shown.

[1] `https://www.bmw-berlin-marathon.com/en/race-day/results-lists.html`

[2] `https://www.uni-weimar.de/de/medien/professuren/medieninformatik/vr/teaching/ss-2016/course-visualization/`