

## 24. BUNDESWETTBEWERB INFORMATIK 2005/2006

Einreichung zur zweiten Runde von Fabian Immler

Verwaltungsnummer 24.013.01

### (a) Allgemeines

Ich bearbeitete für meine Einsendung die Aufgaben 1 und 3. Die Museumsaufgabe schien ein recht anschauliches Thema zu sein, und als begeisterter Fussballspieler und –Fan *musste* ich natürlich berechnen wer Weltmeister wird.

Meine Programme sind alle objektorientiert in C++ geschrieben, als Entwicklungsumgebung habe ich **Bloodshed Dev-C++** verwendet. Die Projektdateien finden sich beim jeweiligen Quelltext auf der CD.

Klassennamen besitzen immer ein vorangestelltes ‚C‘ und sind, wie auch Variablen, Funktionen o.ä. *kursiv* gedruckt.

Der Quelltext zum gesamten Programm ist immer am Ende der Bearbeitung zur betreffenden Aufgabe abgedruckt. Er ist nach Dateien alphabetisch sortiert, wobei als erstes immer ‚Main.cpp‘ abgedruckt ist. Klassen befinden sich immer in den Dateien ‚Klassenname.h‘ und ‚Klassenname.cpp‘. Bei längeren Funktionen wurden wichtige Bereiche **fett** markiert, um ihr auffinden zu erleichtern.

Die Programmablaufprotokolle sind, vor allem bei Aufgabe 3 meist nur gekürzt wiedergegeben und konzentrieren sich auf die wichtigsten Aspekte der Aufgabenstellung.

Auf der beiliegenden CD befinden sich neben den ausführbaren Programmen auch die Quelltexte.

## (b) Unterlagen zur Aufgabe 1: Museum

**1. Erstelle ein Programm, mit dem man die Form des Museums einlesen und visualisieren kann. Das Programm muss mit der unten gezeigten Beispielform, aber auch mit anderen Formen zurechtkommen.**

### Lösungsidee:

Zum Einlesen der Form werden jeweils die Koordinaten nach ‚polygon‘ eingelesen und in einer Klasse gespeichert, die das Museum als mehrere Polygone speichert.

Zum Anzeigen des Museums müssen die Gleitkommakordinaten der Polygone in Bildschirmkoordinaten(Pixel) umgerechnet werden. Beim Zeichnen der Polygone auf den Bildschirm wird das Rand-Polygon heller ausgefüllt als die Löcher, um ein ähnliches Ergebnis zu erzielen wie in der Aufgabenstellung.

### Dokumentation:

Da von einer „vernünftigen“ Eingabe ausgegangen werden kann, und das erste Polygon den Rand des Museums repräsentieren soll (nach FAQ-Seite des BWINF), wird beim Einlesen der Koordinaten auf jegliche Fehlerüberprüfungen verzichtet.

Zur Lösung dieser Teilaufgabe wurden folgende Klassen und Strukturen verwendet:

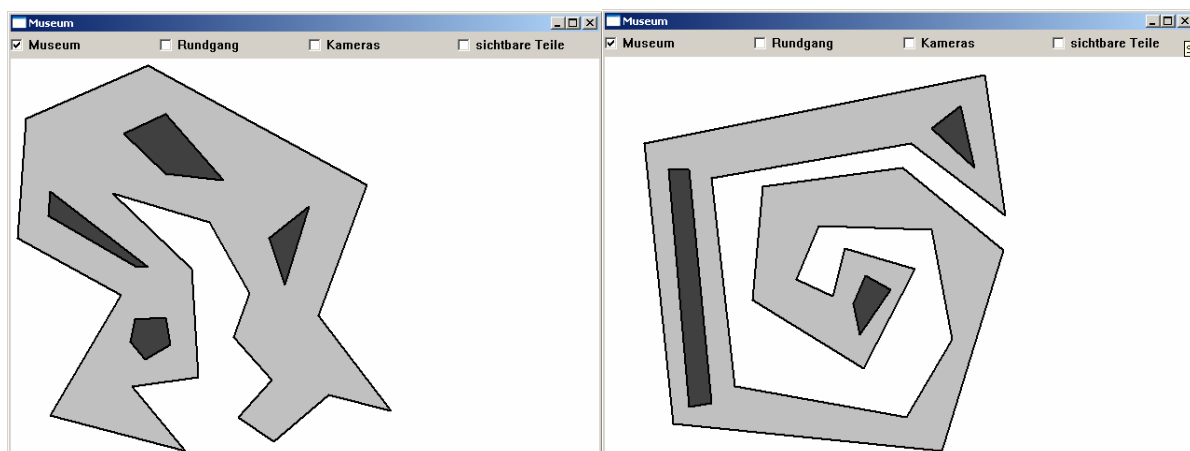
- *SPunkt*: besitzt eine X- und eine Y-Koordinate in Gleitkommaform. Die Ecken der Polygone werden als *SPunkte* eingelesen.
- *CPolygon*: Speichert ein Feld von *Punkten*.
- *CMuseum*: Beinhaltet ein Polygon als Rand und ein Feld aus Polygonen für die Löcher im Museum.

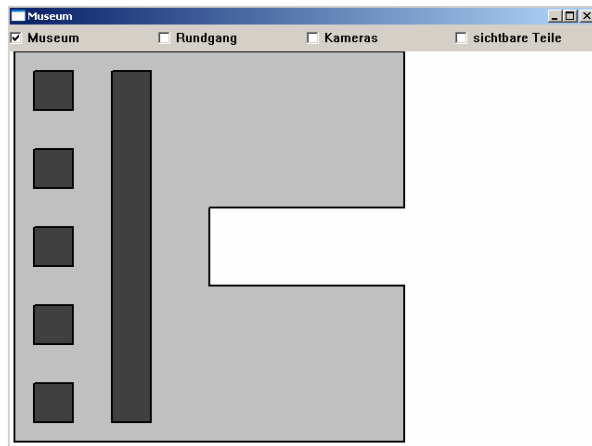
Beim Einlesen werden also Koordinate für Koordinate Punkte gebildet, aus diesen Punkten Polygone gebildet, die dann als Museum gespeichert werden, wobei das erste Polygon den Rand darstellt.

Zum Umrechnen der Koordinaten wird eine Variable *schrittweite* eingeführt, die die Entfernung zwischen zwei Pixeln in ‚Polygonkoordinateneinheiten‘ angibt. Wenn zum Beispiel ein Polygon der Breite 24,1 auf einem 640 Pixel breiten Bildschirm angezeigt werden soll, beträgt die Schrittweite  $24,1/640$  [Polygoneinheiten/Pixel].

### Programmablaufprotokolle:

Das erste Beispiel ist natürlich das Museum aus der Aufgabenstellung. Es ist Allerdings an der horizontalen gespiegelt, da das Koordinatensystem des Beispielmuseums in y-Richtung anders zählt als mein Programm. Um zu zeigen, dass das Programm auch mit anderen Formen zurechtkommt, habe ich noch weitere Museen angefertigt. Eines ist noch ein bisschen verwinkelter als das erste, das andere beweist, dass das Programm auch ‚normale‘, rechteckige Museen anzeigen kann.





Hier sind die Eingabedateien der zwei selbst erstellten Museen:

Spirale		Rechteck	
<p>           polygon 21            18.3 0.9            19.3 7.7            14.7 4.2            5.0 5.9            6.1 16.0            14.5 17.5            16.7 13.7            15.7 8.4            10.2 8.2            9.1 10.8            10.9 11.6            11.5 9.3            14.9 10.3            12.4 15.1            7.0 11.8            7.5 6.3            14.3 5.4            19.2 9.4            16.2 19.1            3.1 17.8            1.7 4.2         </p>	<p>           polygon 3            17.1 2.4            15.7 3.5            17.8 5.4            polygon 4            12.5 10.6            11.9 12.0            12.2 13.5            13.7 11.3            polygon 4            2.9 5.5            3.9 17.0            5.0 16.8            3.9 5.5         </p>	<p>           polygon 8            0.0 0.0            20.0 0.0            20.0 8.0            10.0 8.0            10.0 12.0            20.0 12.0            20.0 20.0            0.0 20.0            polygon 4            1.0 1.0            3.0 1.0            3.0 3.0            1.0 3.0            polygon 4            1.0 5.0            3.0 5.0            3.0 7.0            1.0 7.0         </p>	<p>           polygon 4            1.0 9.0            3.0 9.0            3.0 11.0            1.0 11.0            polygon 4            1.0 13.0            3.0 13.0            3.0 15.0            1.0 15.0            polygon 4            1.0 17.0            3.0 17.0            3.0 19.0            1.0 19.0            polygon 4            5.0 1.0            7.0 1.0            7.0 19.0            5.0 19.0         </p>

## 2. Erweitere dein Programm so, dass zusätzlich ein Rundgang des Nachtwächters eingelesen werden kann.

### Lösungsidee:

Als Format zum Einlesen eines Rundgangs des Nachtwächters eignet sich dasselbe Format wie aus Aufgabe 1. In der Rundgang-Datei darf sich nur ein Polygon befinden, welches für den Pfad des Nachtwächters steht. Angezeigt wird der Rundgang dann als nicht ausgefülltes Polygon.

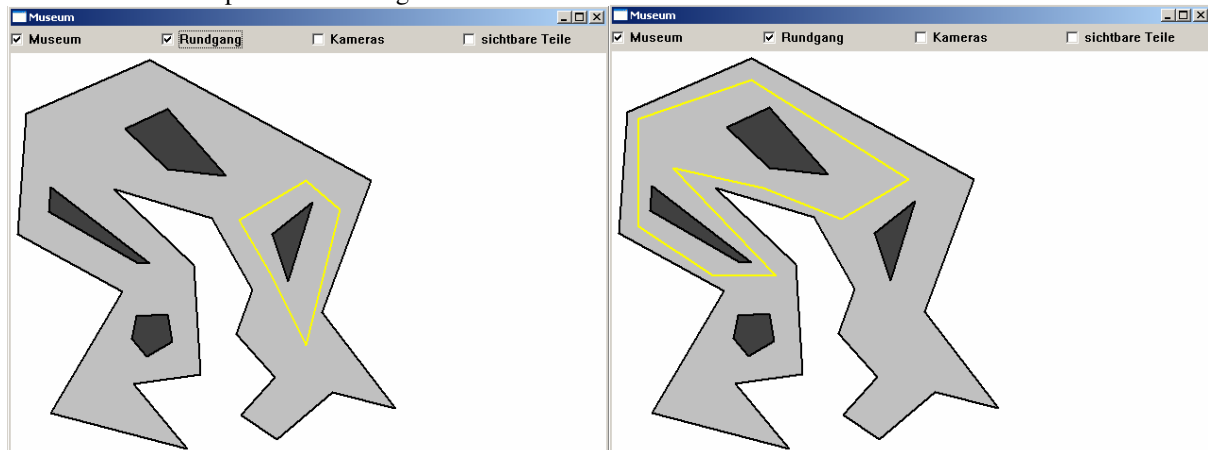
### Dokumentation:

Eingelesen wird der Pfad genauso wie in Aufgabe 1, er wird als *CPolygon* gespeichert. Dabei wird wieder eine ‚vernünftige‘ Eingabe erwartet.

Das Programm lädt beim Start die Datei „pfad.txt“. Wenn andere Pfade gewünscht sind, müssen sie manuell vor Programmstart aus den Unterverzeichnissen kopiert werden. Zum Anzeigen des Pfades das Häkchen vor ‚Rundgang‘ markieren.

### Programmablaufprotokolle:

Zur Demonstration der Funktionalität sind hier zwei verschiedene Rundgänge mit den jeweiligen Dateien für das Beispielmuseum dargestellt:



<pre> polygon 5 13.0 5.7 10.0 7.5 11.5 10.0 13.0 13.1 14.5 7.0 </pre>	<pre> polygon 9 6.0 1.3 13.0 5.7 10 7.5 6.5 6.1 2.5 5.2 7.1 10 4.3 10 1.0 7.8 1.0 3.0 </pre>
---	--

### 3. Erweitere dein Programm zusätzlich so, dass es die Teile des Ausstellungsraums anzeigt, die der Nachtwächter während seines Rundgangs mindestens einmal sehen kann. Die Ausstellungsgegenstände sind so klein, dass sie die Sicht nicht behindern; aber der Nachtwächter kann natürlich nicht durch Wände sehen.

#### Lösungsidee:

Als ‚sichtbarer Bereich‘ werden die Teile des Bodens des Museums bezeichnet, die der Nachtwächter sehen kann. Der Nachtwächter kann von jedem Punkt auf seinem Rundgang 360° in alle Richtungen sehen (Wenn er stehen bleibt und sich umsieht). Sie sollen vom Programm grün gefärbt werden. Der Nachtwächter kann natürlich auch die Teile der Wände sehen, die allerdings nicht gefärbt werden um eine gewisse Übersichtlichkeit zu erhalten. Die Teile der Wände die an grünen Bden grenzen werden vom Nachtwächter gesehen.

Da es nur um das Anzeigen von Sichtbaren Bereichen geht wird der sichtbare Bereich nicht durch Polygone repräsentiert. Vielmehr wird für jedes Pixel des Museums geprüft, ob es vom Wächter gesehen werden kann. Dazu wird eine Linie zwischen dem Pixel und dem Pfad des Wächters gezogen. Wenn diese Linie das Museum nicht schneidet ist das Pixel für den Wächter sichtbar und wird grün gefärbt. Sollte die Linie das Museum schneiden muss das Pixel mit anderen Punkten des Pfades verbunden werden, denn vielleicht sieht der Wächter das Pixel später auf seinem Rundgang. Wenn jede Verbindungslinie das Museum schneidet, ist das Pixel für den Wächter nicht zu sehen und wird rot eingefärbt.

#### Dokumentation:

Zur Realisierung der Lösungsidee wurden die Klassen aus Aufgabe 1 um einige Funktionen erweitert und noch folgende Klassen hinzugefügt:

##### CLinie:

Stellt eine Strecke zwischen zwei Punkten dar, die in vektorieller Punkt-Richtungsform gespeichert wird:

$$\vec{X} = \begin{pmatrix} \text{px} \\ \text{py} \end{pmatrix} + m * \begin{pmatrix} \text{ux} \\ \text{uy} \end{pmatrix}$$

Die Variablen px,py,ux,uy legen eine Gerade fest, so dass für Punkte auf der Strecke zwischen Anfangs- und Endpunkt gilt:  $0 \leq m \leq 1$

Die Klasse liefert Methoden zum:

-Zurückgeben des zum Parameterwert  $m$  gehörigen Punktes  $\rightarrow$  *PUNKT CLinie::Punkt(float m)*

-Berechnen ob eine Linie eine andere schneidet  $\rightarrow$  *bool CLinie::Schnitt(CLinie)*

Das Überprüfen auf einen Schnittpunkt erfolgt indem die zwei

$$\text{Geraden } \vec{X} = \begin{pmatrix} \text{px} \\ \text{py} \end{pmatrix} + m * \begin{pmatrix} \text{ux} \\ \text{uy} \end{pmatrix} \text{ und } \vec{X} = \begin{pmatrix} \text{qx} \\ \text{qy} \end{pmatrix} + n * \begin{pmatrix} \text{vx} \\ \text{vy} \end{pmatrix} \text{ gleichgesetzt}$$

werden. Das 2/2 Gleichungssystem besitzt als Determinante  $D = \text{uy} * \text{vx} - \text{ux} * \text{vy}$  und für  $m$  die Lösung

$$m = \frac{(\text{qy} - \text{py}) * \text{vx} - (\text{qx} - \text{px}) * \text{vy}}{D}$$

Die Lösung für  $n$  ist

$$n = \frac{(\text{qy} - \text{py}) * \text{ux} - (\text{qx} - \text{px}) * \text{uy}}{D}$$

Die beiden Strecken schneiden sich, wenn  $0 \leq m \leq 1$  und  $0 \leq n \leq 1$ , andernfalls liegt der Schnittpunkt außerhalb der Strecken, oder die Strecken sind, falls  $D=0$  parallel.

- Berechnen des Abstandes von einem Punkt zur Linie  $\rightarrow \text{float } CLinie::Abstand(PUNKT P)$

Der Abstand wird mittels Hesse-Normalform der Geraden berechnet:

$$\mathbf{d} = \vec{n}^0 \circ (\vec{P} - \vec{A})$$

*CPolygon*: Die Klasse aus Aufg. 1 wurde um eine Methode erweitert, die überprüft, ob eine Linie das Polygon schneidet. Dazu wird schlichtweg jede Randlinie des Polygons auf einen Schnitt mit der Linie überprüft. Zudem kann die Klasse Verbindungslinien zwischen ihren Punkten zurückgeben.

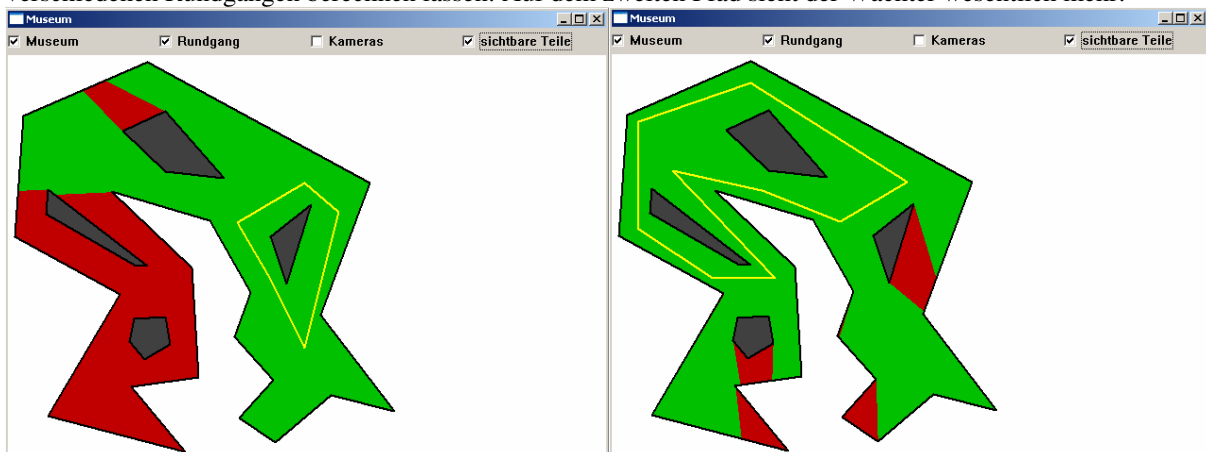
*CMuseum*: Die Klasse aus Aufg. 1 wurde ebenfalls um eine Methode erweitert, die überprüft, ob eine Linie das Museum schneidet. Dazu wird schlichtweg jedes Polygon des Museums auf einen Schnitt mit der Linie überprüft.

Funktion *WindowProcedure*(in Datei main.cpp):

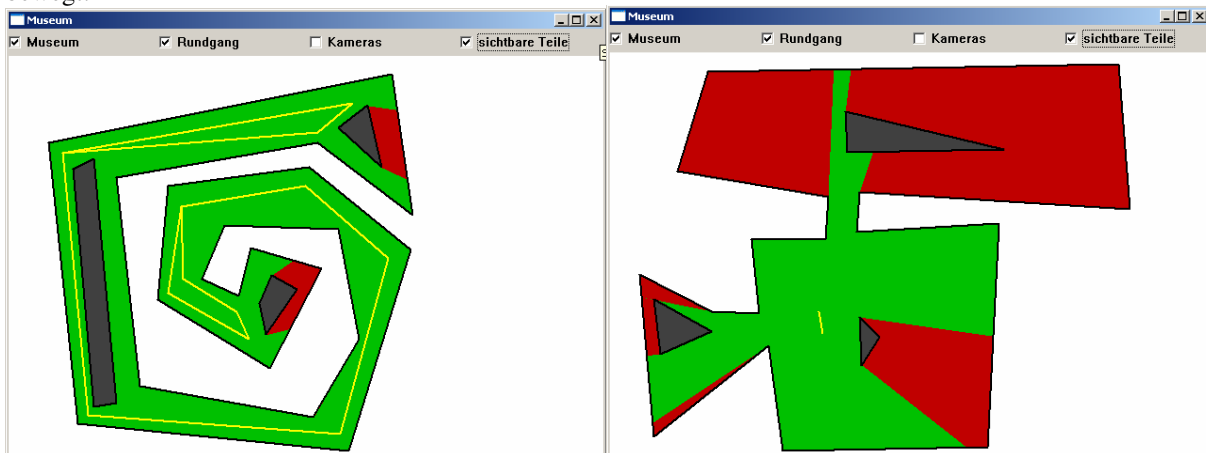
In der Zeichenfunktion wird zum Markieren der Sichtbaren Bereiche jeder Punkt des Bildschirms überprüft. Wenn ein Punkt hellgrau ist, bedeutet das, dass er zum Museum gehört und noch nicht überprüft wurde. Zwischen diesem Punkt und einem Punkt des Pfades wird eine *CLinie* gebildet und auf einen eventuellen Schnitt mit dem Museum überprüft (*CMuseum::Schnitt(CLinie)*). Als Punkt des Pfades wird jeder Punkt jeder Linie des Rundgang-Polygons verwendet.

### Programmablaufprotokolle:

Als erstes Beispiel habe ich das Sichtfeld des Wächters im Beispielmuseum mit den beiden verschiedenen Rundgängen berechnen lassen: Auf dem zweiten Pfad sieht der Wächter wesentlich mehr:



Interessant fand ich auch das Sichtfeld in einem Museum mit ‚Nebenräumen‘ in dem sich der Wächter kaum bewegt:



**4. Erweiterung: Einen Nachtwächter anzustellen ist teuer, und wie an den obigen Beispielen deutlich wurde, sieht der Nachtwächter meist nur Teile des Ausstellungsraumes. Zudem kann er nie das ganze Museum auf einmal überblicken. Somit ist der Wächter für seine Kosten sehr uneffektiv.**

**Deshalb würde ich als Museumsbesitzer Überwachungskameras einbauen, die den Ausstellungsraum überwachen.**

**Es muss zwar immer noch jemand angestellt sein, der vor den Monitoren sitzt, dieser kann aber bequem sitzen und sieht alles auf einmal.**

**Das Programm soll vorschlagen, wo Kameras angebracht werden könnten, damit der gesamte Raum von so wenig Kameras wie möglich (Kameras sind teuer und der Überwacher will ja nicht vor 100 Monitoren sitzen, da geht ebenfalls der Überblick verloren) überwacht ist.**

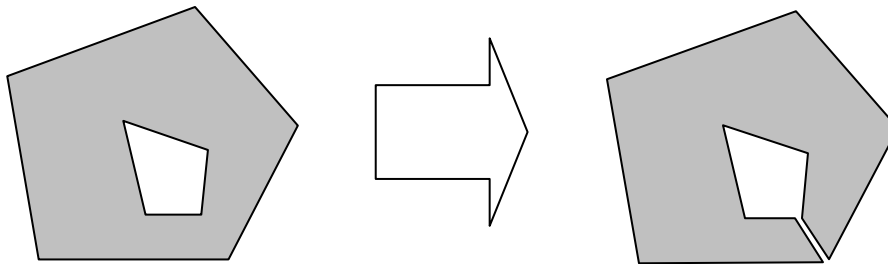
**Lösungsidee:**

Die verwendeten Kameras sind entweder schwenkbar oder haben ein extremes Weitwinkelobjektiv, sodass sie von ihrem Standpunkt aus ein 360° Blickfeld haben. Weiterhin wird davon ausgegangen, dass die Montage der Kameras in den Ecken des Museums erfolgt.

Um einen Vorschlag für die Kameraverteilung zu machen wird das Museum zunächst in Dreiecke zerlegt. Wenn sich eine Kamera in einem Eck vom Dreieck befindet, kann sie das ganze Dreieck überwachen. Nun soll an die Ecken des Museums Kameras platziert werden, und zwar so, dass jedes Dreieck mindestens eine Kamera enthält.

Ein Dreieck wird so markiert, dass jedes Eck eine andere Farbe erhält(rot, grün oder blau). Die angrenzenden Dreiecke werden dann so gefärbt, dass auch bei ihnen jedes Eck anders gefärbt ist. Nun könnte man in alle roten Ecken Kameras platzieren, oder in alle grünen oder in alle blauen und das ganze Museum wäre von den Kameras zu sehen, da ja jedes Dreieck eine dieser Farben hat und es somit genügt nur zu einer Farbe eine Kamera zu platzieren. Die Kameras werden dann zu der Farbe platziert, die am seltensten vorkommt.

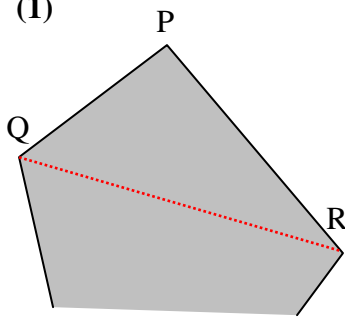
Ein Problem dabei ist, das Museum in Dreiecke zu teilen. Ein Hindernis dabei sind die Löcher. Sie werden einfach in das Polygon mit aufgenommen. Dabei werden quasi zusätzliche Wände durch das Museum gezogen:



So werden alle Löcher eliminiert, immer an den Ecken mit der kürzesten Entfernung zum Rand.

Das eigentliche Zerteilen des Polygons geschieht nach folgendem Schema:

(1)



(1) Zunächst wird der Punkt mit der minimalen y-Koordinate gesucht (P), dann werden seine beiden Nachbarpunkte(Q,R) miteinander verbunden und das Polygon dort geteilt. P muss der am höchsten gelegene Punkt sein, da sonst die Verbindung QR ausserhalb des Museums liegen könnte.

(2) Sollte QR nicht komplett innerhalb des Museums liegen wird der Punkt T gesucht, der von QR den größten Abstand hat. So liegt die Verbindung PT sicher innerhalb des Museums. Das Museum wird dann bei PT geteilt, und mit den beiden entstehenden Teilen wird weiter so verfahren, bis nur noch Dreiecke übrig bleiben.

### Dokumentation:

Die zur Realisierung der Lösungsidee notwendigen Funktionen finden sich in der Datei ,polyfunkt.cpp':

- *eliminiereLoecher(CPolygon[] RandUndLoecher)* eliminiert die Löcher eines Polygons, indem die Punkte des Lochs in das Punktfeld des Randes eingefügt werden, sodass der Polygonverlauf korrekt bleibt. Zum einfacheren Einfügen wird der Drehsinn des Randes und der der Löcher entgegengesetzt orientiert.
- *CPolygon[] Trianguliere(CPolygon)*: Zerteilt (über die Fkt. *Teile()*) ein Polygon in Dreiecke und gibt sie als Feld zurück
- *CPolygon Faerben(CPolygon[] Dreiecke)*: Färbt ein Dreieck zunächst willkürlich und sucht angrenzende ungefärbte

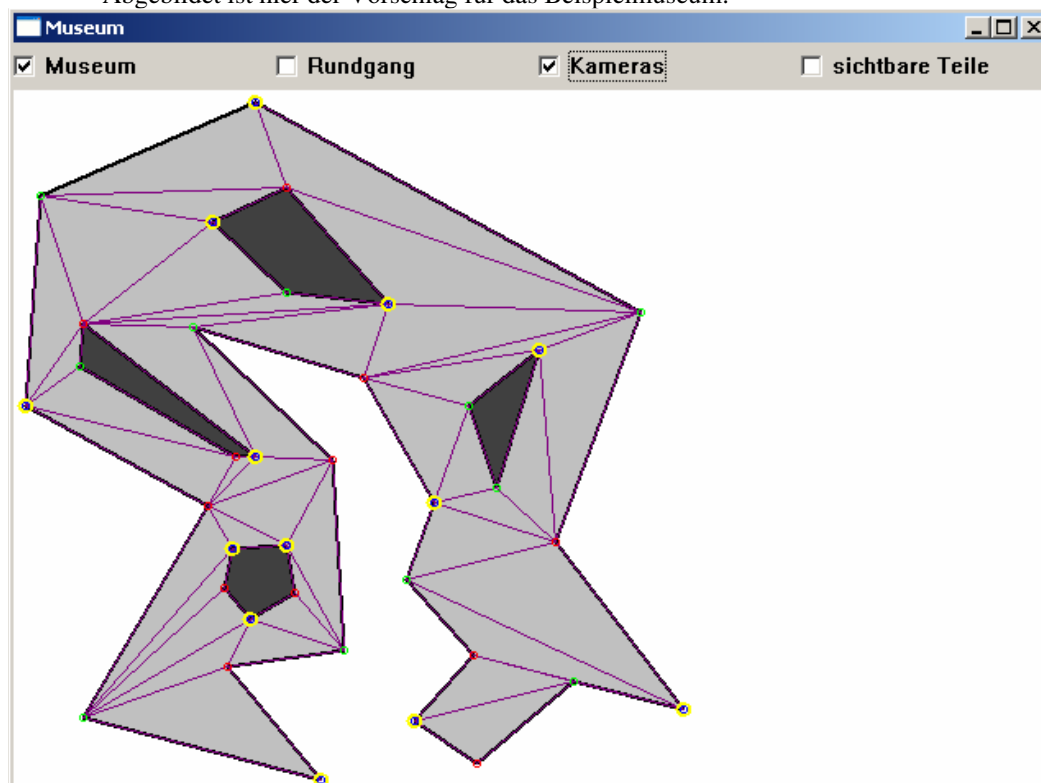
Dreiecke, die entsprechend gefärbt werden

- *WinMain*(in Datei main.cpp): im Hauptprogramm wird nach dem Einlesen der Dateien das Museum zunächst von seinen Löchern befreit: *CPolygon berichtigt=eliminiereLoecher(Polygone)*. Das entstehende Polygon *berichtigt* wird dann trianguliert
- *WindowProcedure*(in Datei main.cpp): in der Zeichenfunktion wird die Triangulierung gefärbt, dann die am seltensten vorkommende Farbe gesucht und die Kameras platziert.

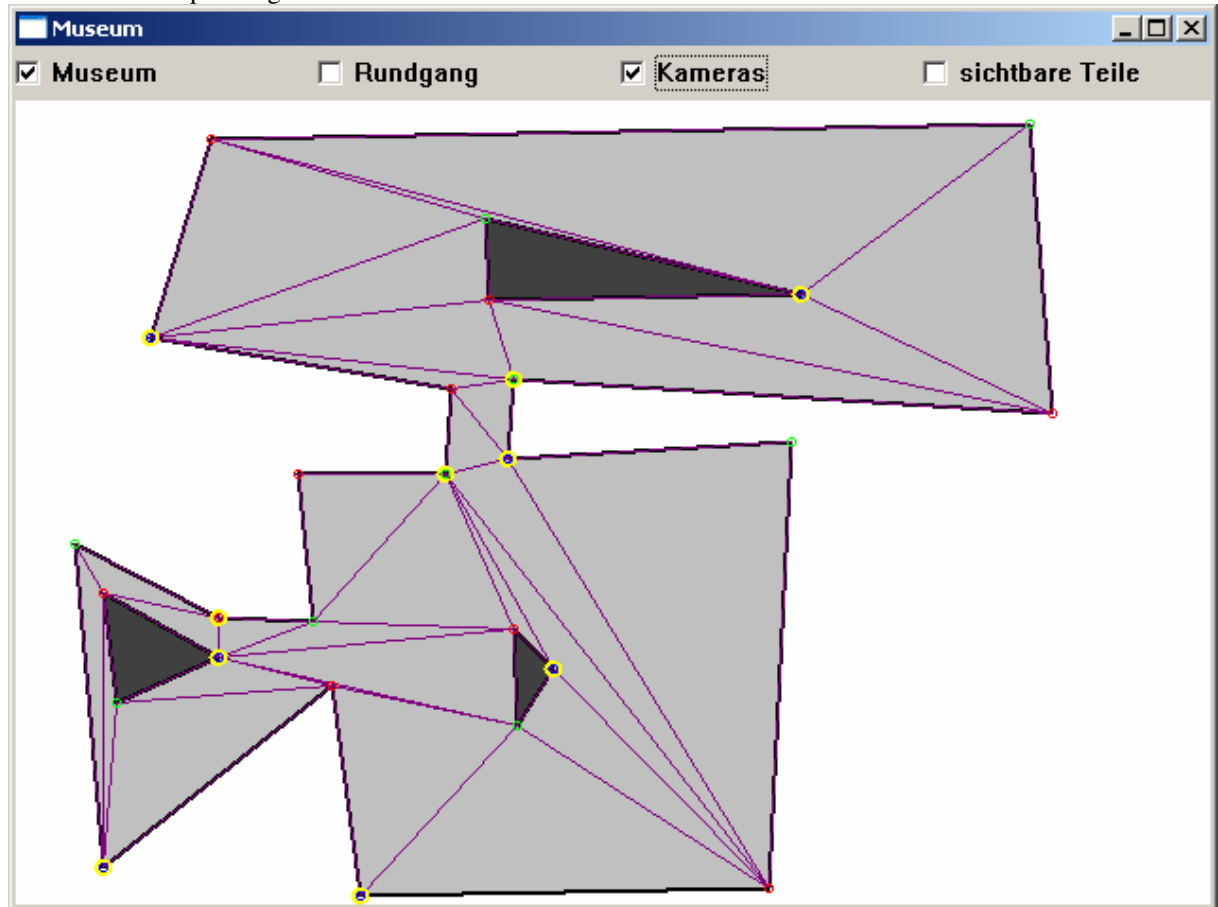
Im Programm wird dann (zwecks Nachvollziehbarkeit) Die Triangulierung lila angezeigt, und jedes Eck ist mit seiner Farbe versehen. Die Kameras werden durch gelbe Kreise symbolisiert.

### Programmablaufprotokolle:

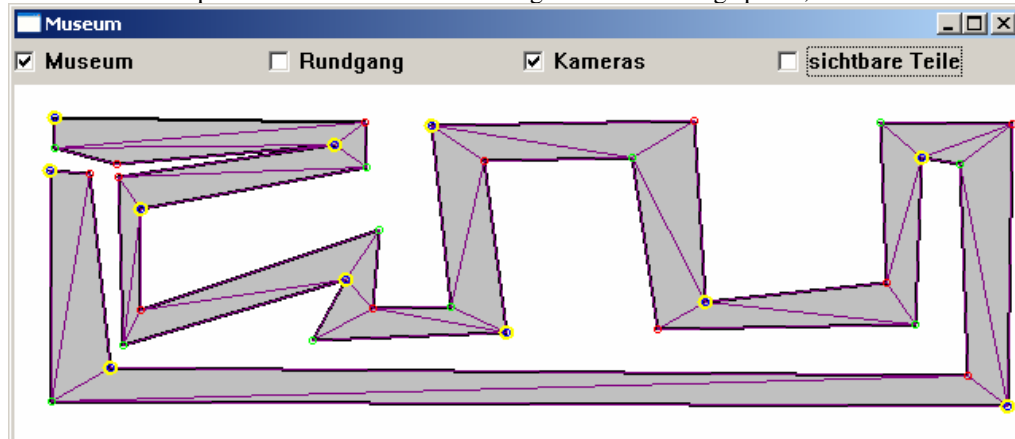
Abgebildet ist hier der Vorschlag für das Beispielmuseum:



Optimal ist der Vorschlag noch nicht, insbesondere aufgrund der künstlich eingezogenen Wände (beim Eliminieren der Löcher) sind ein paar überflüssige Kameras vorgeschlagen.  
Das nächste Beispiel zeigt die Funktionalität anhand eines anderen selbst entworfenen Museums.



Beim letzten Beispiel kann sich der Wächter einige Meter Laufweg sparen, wenn er Kameras einbauen lässt.





**Programmtext:****Main.cpp (Auszüge)**

```

//Globale Variablen
CMuseum Museum;
CPolygon Pfad; //Pfad des Nachtwächters
CPolygon berichtet; //polygon ohne Löcher
vector <CPolygon> Triangulierung;
vector <PUNKT> Kameras;

WinMain(){
    //Museum laden
    ifstream Museum_file; //Datei-Objekt
    Museum_file.open("museum-form.txt", ios::binary|ios::in);
    //Fehler beim öffnen??
    if(!Museum_file){
        MessageBox(hwnd, "Datei konnte nicht geöffnet werden", "FEHLER", 0);
    }
    string Input;
    //Datei auslesen
    vector<CPolygon> Polygone;
    Museum_file >> Input;
    while(!Museum_file.eof()){
        if(Input=="polygon"){//Beginn eines neuen Polygons
            CPolygon akt_poly; //Speicherort während des Auslesens
            akt_poly.Loeshen(); //zurücksetzen
            Museum_file >> Input;
            int groesse=atoi(Input.c_str()); //Anzahl der Punkte des Polygons ermitteln
            PUNKT punkt;
            for(int p=0;p<groesse;p++){//Alle Punkte auslesen
                Museum_file >> Input;
                punkt.X=tofloat(Input.c_str());
                MinX=min(punkt.X,MinX);
                MaxX=max(punkt.X,MaxX);
                Museum_file >> Input;
                punkt.Y=tofloat(Input.c_str());
                MinY=min(punkt.Y,MinY);
                MaxY=max(punkt.Y,MaxY);
                akt_poly.Add(punkt);
            } //Punkte auslesen
            Polygone.push_back(akt_poly);
        } //Neues Polygon in Datei
        Museum_file >> Input;
    }
    //Polygone in Museum übertragen
    Museum.SetzeRand(Polygone[0]);
    for(int index=1; index<Polygone.size(); index++) Museum.AddLoch(Polygone[index]);

    //Museum berichtigen und Triangulieren
    berichtet=eliminiereLoecher(Polygone);
    Triangulierung=Trianguliere(berichtet);

    //Pfad des Nachtwächters einlesen
    ifstream Pfad_file; //Datei-Objekt
    Pfad_file.open("pfad.txt", ios::binary|ios::in);
    //Fehler beim öffnen??
    if(!Pfad_file){
        MessageBox(hwnd, "Datei pfad.txt konnte nicht geöffnet werden", "FEHLER", 0);
    }
    //Datei auslesen
    Pfad_file >> Input;
    if(Input=="polygon"){//Beginn eines neuen Polygons
        Pfad_file >> Input;
        int groesse=atoi(Input.c_str()); //Anzahl der Punkte des Polygons ermitteln
        PUNKT punkt;
        for(int p=0;p<groesse;p++){//Alle Punkte auslesen
            Pfad_file >> Input;
            punkt.X=tofloat(Input.c_str());
            Pfad_file >> Input;
            punkt.Y=tofloat(Input.c_str());
            Pfad.Add(punkt);
        } //Punkte auslesen
    } //Neues Polygon in Datei
}

// F E N S T E R F U N K T I O N
LRESULT CALLBACK WindowProcedure (...)
{

```

```

static int BreiteR, HoeheS;
static int cxChar,cyChar;//Breite bzw. Höhe von Zeichen
static HWND hCheckMuseum,hCheckPfad,hCheckKamera,hCheckSicht;
static bool bMuseum,bPfad,bKamera,bSicht;//Flags für die Kontrollkästchen
HDC hdc;
PAINTSTRUCT ps;
HPEN hPen;
switch (message)                /* handle the messages */
{
    case WM_SIZE:
        BreiteR=LOWORD(lParam);
        HoeheS=HIWORD(lParam);
        break;
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps);
        float schrittweite;//Pixel pro Relativkoordinate
        //Seitenverhältnis überprüfen=> Umrechnungsfaktor für Koordinaten
        if((BreiteR-10)/(float) (HoeheS-7*cyChar/4-5)>(MaxX-MinX)/(MaxY-MinY)){
            schrittweite=(HoeheS-7*cyChar/4-5)/(MaxY-MinY);
        }
        else schrittweite=(BreiteR-10)/(MaxX-MinX);

        //Fürs Zeichnen benötigte Variablen
        POINT apt[500]; //Array aus Punkten

        //Museum malen
        if(bMuseum){
            //Rand malen
            int punkte=Museum.Rand().AnzahlPunkte();
            for(int pktnr=0;pktnr<punkte;pktnr++){
                apt[pktnr].x = 5 + LONG((Museum.Rand().Punkt(pktnr).X - MinX) *
                                                    schrittweite);
                apt[pktnr].y = 7*cyChar/4 + LONG((Museum.Rand().Punkt(pktnr).Y - MinY) *
                                                    schrittweite);

            }//Jeden Punkt ins Punkttarray speichern.

        }

        //Kameras berechnen und malen
        if(bKamera){
            //BERECHNEN
            //Dreiecke färben
            Faerben(berichtigt,Triangulierung);
            //Am wenigsten benutzte Farbe suchen
            int r=0,g=0,b=0;
            for(int d=0;d<Triangulierung.size();d++){
                for(int n=0;n<3;n++){
                    if(Triangulierung[d].Faerbung(n)==1)r++;
                    else if(Triangulierung[d].Faerbung(n)==2)g++;
                    else if(Triangulierung[d].Faerbung(n)==3)b++;
                }
            }
            int minf=(r<g)?(r<b?1:3):(g<b?2:3);
            //Kameras Platzieren
            for(int d=0;d<Triangulierung.size();d++){
                for(int n=0;n<3;n++){
                    if(Triangulierung[d].Faerbung(n)==minf)
                        Kameras.push_back(Triangulierung[d].Punkt(n));
                }
            }
            //Kameras anzeigen

        }

        //SICHTBARE TEILE ANZEIGEN
        PUNKT startpkt,endpkt;
        if(bSicht){
            for(int s=0;s<HoeheS;s++){//Bildschirmkoordinaten(Pixel)
                for(int r=0;r<BreiteR;r++){if(GetPixel(hdc,r,s)==RGB(192,192,192)){// -"-
                    startpkt.X=(r-5)/schrittweite+MinX;//Polygonkoordinaten des zu
                                                    untersuchenden Pnktes
                    startpkt.Y=(s-7*cyChar/4)/schrittweite+MinY;// -'"
                    bool zusehen=false;
                    //Alle Sichtlinien zu den Kameras untersuchen
                    if(bKamera){
                        for(int k=0;k<Kameras.size();k++){
                            endpkt=Kameras[k];
                            Clinie strahl(startpkt,endpkt);
                            if(!Museum.Schnitt(strahl)) zusehen=true;
                            if(zusehen)break;
                        }
                    }
                    //alle Richtungen zum Pfad untersuchen(alles in Poly-koordinaten)
                }
            }
        }
    }
}

```

```

        //jedes pixel vom Pfad untersuchen
        for(float param=0; param<=1; param+=2./schrittweite){
            //Jedes Teil des Pfades untersuchen
            for(int index=0;index<Pfad.AnzahlPunkte();index++){
                endpkt=Pfad.Linie(index).Punkt(param);
                CLinie strahl(startpkt,endpkt);
                if(!Museum.Schnitt(strahl)) zusehen=true;
                if(zusehen)break;
            }//for(Jedes Teil vom Pfad)
            if(zusehen)break;
        }//for(jedes pixel vom Pfad)
    }
    if(zusehen){
        SetPixel(hdc,r,s,RGB(0,192,0));
    }
    else{
        SetPixel(hdc,r,s,(RGB(192,0,0)));
    }
} //for(s)
} //for(r)
} //if(Sicht)
return 0;
}

```

### Linie.h

```

//Klasse zum Verwalten/Zeichnen und vergleichen von Linienabschnitten.
//Linie ist in Vektorform definiert:
//
//          (px)      (ux)
// Linie: X = (py)   m*(uy)   ; wobei 0<=m<=1
#ifndef linie_h
#define linie_h linie_h

#include "punkt.h"
#include <windows.h>

class CLinie{
public:
    //Initialisieren
    CLinie();
    CLinie(PUNKT Start, PUNKT Ende);
    void SetStart(PUNKT Start);
    void SetEnde(PUNKT Ende);
    //Aktive Methoden
    bool comp(CLinie);
    bool Schnitt(CLinie Linie2);
    float Abstand(PUNKT p);
    bool Ueber(PUNKT p);
    //Rückgabewerte
    PUNKT Punkt(float m){PUNKT p; p.X=px+m*ux; p.Y=py+m*uy;return p;}
    float RichtungX(){return ux;}
    float RichtungY(){return uy;}
private:
    float px,py;//Koordinaten des Aufpunkts
    float ux,uy;//Koordinaten des Richtungsvektors.
};

#endif //linie_h

```

### Linie.cpp

```

//Implementation der Klasse zum Verwalten/Zeichnen und vergleichen von Linienabschnitten.
//Linie ist in Vektorform definiert:
//
//          (px)      (ux)
// Linie: X = (py)   m*(uy)   ; wobei 0<=m<=1
#include "linie.h"

CLinie::CLinie(){
    px=py=ux=uy=0;
}
CLinie::CLinie(PUNKT Start, PUNKT Ende){
    px=Start.X;
    py=Start.Y;
    ux=Ende.X-px;
    uy=Ende.Y-py;
}

void CLinie::SetStart(PUNKT Start){

```

```

    px=Start.X;
    py=Start.Y;
}
void CLinie::SetEnde(PUNKT Ende){
    ux=Ende.X-px;
    uy=Ende.Y-py;
}

float CLinie::Abstand(PUNKT p){
    //Abstand nach HNF der Gerade
    //Laenge des Normalenvektors der Gerade
    float n=sqrt(1+ux*ux/(uy*uy));
    //Koordinaten des Einheits-Normalenvektors n0
    float n0x=1./n;
    float n0y=(-ux/uy)/n;
    return (n0x*(p.X-px)+n0y*(p.Y-py));
}

bool CLinie::comp(CLinie l){
    return ((px>=l.Punkt(0).X-.001&&py>=l.Punkt(0).Y-.001&&ux>= l.RichtungX()-0.001&&uy>=
l.RichtungY()-0.001)&&(px<=l.Punkt(0).X+.001&&py<=l.Punkt(0).Y+.001&&ux<=
l.RichtungX()+0.001&&uy<= l.RichtungY()+0.001))
    ||((px>=l.Punkt(1).X-.001&&py>=l.Punkt(1).Y-.001&&ux>=-l.RichtungX()-0.001&&uy>=-
l.RichtungY()-0.001)&&(px<=l.Punkt(1).X+.001&&py<=l.Punkt(1).Y+.001&&ux<=-
l.RichtungX()+0.001&&uy<=-l.RichtungY()+0.001));
}

bool CLinie::Schnitt(CLinie Linie2){
    //Überprüfen, ob die Linie sich mit Linie2 schneidet;
    //      (qx)      (vx)
    //Linie2:X=(qy) + n*(vy)
    PUNKT Startpunkt=Linie2.Punkt(0);
    float qx=Startpunkt.X;
    float qy=Startpunkt.Y;
    float vx=Linie2.RichtungX();
    float vy=Linie2.RichtungY();

    // Schnittpunktberechnung mit Cramer-Regel/Determinantenmethode
    float m,n;//Parameter der beiden Geraden, wenn sie BEIDE zwischen 0 und 1 liegen, dann
    schneiden sich die Linien
    float D=uy*vx-ux*vy;
    if(D!=0){//Überprüfen auf Parallelität
        m=((qy-py)*vx-(qx-px)*vy)/D;
        n=((qy-py)*ux-(qx-px)*uy)/D;
        return (m>=.000001 && m<=.999999 && n>=.000001 && n<=.999999);//Bedingung für
    Schnitt;Schnitt an Endpunkten zaehlt nicht
    }else return false;//bei Parallelität natürlich kein Schnitt
}

```

## Museum.h

```

// Klasse zum Verwalten des Museums
#ifdef museum_h
#define museum_h museum_h

#include <windows.h>
#include <vector>
#include "polygon.h"
using namespace std;

class CMuseum{
public:
    CMuseum();
    CMuseum(CPolygon derRand, vector<CPolygon> dieLoecher);
    void SetzeRand(CPolygon derRand);
    void AddLoch(CPolygon dasLoch);
    CPolygon Rand(){return Randpoly;}
    UINT Loecher(){return Lochpoly.size();}
    CPolygon Loch(UINT nr){return Lochpoly.at(nr);}
    bool Schnitt(CLinie Linie);// Schneidet eine Linie das Museum?
private:
    CPolygon Randpoly;
    vector<CPolygon> Lochpoly;
};

#endif //museum_h

```

**Polyfunkt.cpp**

```
//Implementation der Funktionen mit Polygonen
#include "polyfunkt.h"

//Entf. zw 2 Punkten
float Entfernung(PUNKT p1,PUNKT p2){
    return sqrt(pow(p1.X-p2.X,2)+pow(p1.Y-p2.Y,2));
}

//Das Loch in das Polygon miteinbauen
CPolygon eliminiereLoecher(vector<CPolygon> polys){
    //Orientierung der Polygone ändern(==> Rand: Rechts, Löcher: Links)
    //höchsten Punkt finden(y=minimal)
    for(int p=0;p<polys.size();p++){
        int imin=200;
        float ymin=200;
        for(int i=0;i<polys[p].AnzahlPunkte();i++){
            if(polys[p].Punkt(i).Y<ymin){
                ymin=polys[p].Punkt(i).Y;
                imin=i;
            }
        }
        //höchsten Nachbarpunkt finden(y=minimal)
        int indexlinks=(imin-1);
        int indexrechts=(imin+1);
        if(imin==0)indexlinks=polys[p].AnzahlPunkte()-1;
        if(imin==polys[p].AnzahlPunkte()-1)indexrechts=0;
        bool rechts_orientiert=true;
        if(polys[p].Punkt(indexlinks).Y<polys[p].Punkt(indexrechts).Y){
            //Wenn Linie vom Punkt aus nach rechts schneidet,
            //dann ist das Polygon richtig orientiert
            PUNKT start=polys[p].Linie(indexlinks).Punkt(0.1);
            PUNKT ende=start;ende.X=start.X+1000;
            CLinie hlinie(start,ende);
            if(hlinie.Schnitt(polys[p].Linie(imin)))rechts_orientiert=true;
            else rechts_orientiert=false;
        }else{
            //Wenn Linie vom Punkt aus nach links schneidet, dann ist das
            //Polygon richtig orientiert
            PUNKT start=polys[p].Linie(imin).Punkt(0.9);
            PUNKT ende=start;ende.X=start.X-1000;
            CLinie hlinie(start,ende);
            if(hlinie.Schnitt(polys[p].Linie(indexlinks)))rechts_orientiert=true;
            else rechts_orientiert=false;
        }
        //Loecher andersherum orientieren:
        if(p>0)rechts_orientiert=!rechts_orientiert;
        //Falls nicht rechts orientiert:Punktreihenfolge invertieren
        if(!rechts_orientiert){
            CPolygon temp;
            for(int i=polys[p].AnzahlPunkte()-1;i>=0;i--){
                temp.Add(polys[p].Punkt(i));
            }
            polys[p].Loeschen();
            for(int i=0;i<temp.AnzahlPunkte();i++)polys[p].Add(temp.Punkt(i));
        }
    }
}

//Kürzesten Abstand suchen
while(polys.size()-1){//Solange, bis nur noch ein Polygon im Feld ist
    float d_min=999999;
    int poly1min,poly2min,pkt1min,pkt2min;

    //Nur mit Rand kombinieren
    int poly1=0;
    for(int poly2=poly1+1;poly2<polys.size();poly2++){
        for(int pkt1=0;pkt1<polys[poly1].AnzahlPunkte();pkt1++){
            for(int pkt2=0;pkt2<polys[poly2].AnzahlPunkte();pkt2++){
                float d=Entfernung(polys[poly1].Punkt(pkt1),polys[poly2].Punkt(pkt2));
                if(d<d_min){
                    //Ist die Verbindung durchgehend?
                    CLinie vbdg(polys[poly1].Punkt(pkt1),polys[poly2].Punkt(pkt2));
                    bool schnitt=false;
                    for(int index=0;index<polys.size();index++){
```

```

        schnitt=polys[index].Schnitt(vbdg);
        if(schnitt)break;
    }
    if(!schnitt){
        //Ist ein Punkt der Vbdg ein doppelter?
        // ==> Vielleicht ist der andere Punkt der Richtige
        bool doppelt=false;
        for(int n=0;n<polys[poly1].AnzahlPunkte();n++){
            if(n!=pkt1 && comp(polys[poly1].Punkt(n),
                               polys[poly1].Punkt(pkt1))) doppelt=true;
        }
        for(int n=0;n<polys[poly2].AnzahlPunkte();n++){
            if(n!=pkt2 && comp(polys[poly2].Punkt(n),
                               polys[poly2].Punkt(pkt2))) doppelt=true;
        }
        if(!doppelt){
            d_min=d;
            pkt1min=pkt1;
            pkt2min=pkt2;
            poly1min=poly1;
            poly2min=poly2;
        }
    }
}

//Verbindung zwischen poly[poly1].Punkt(pkt1) und dem zweiten herstellen und das
//zweite Polygon loeschen
CPolygon t_poly;
//Alle Punkte bis zur Vbdg hinzufuegen
for(int p1=0;p1<=pkt1min;p1++)t_poly.Add(polys[poly1min].Punkt(p1));
for(int p2=pkt2min;p2<polys[poly2min].AnzahlPunkte();p2++){
    t_poly.Add(polys[poly2min].Punkt(p2)); //Von der Vbdg an alle
//Pkt vom Loch hinzufuegen
for(int p2=0;p2<=pkt2min;p2++)t_poly.Add(polys[poly2min].Punkt(p2)); //restliche
//Punkte vom Loch hinzufuegen
for(int p1=pkt1min;p1<polys[poly1min].AnzahlPunkte();p1++){
    t_poly.Add(polys[poly1min].Punkt(p1)); //restliche Punkte vom
}

//Poly hinzufuegen
//Erstes Polygon ersetzen
polys[poly1min]=t_poly;
//Zweites Polygon loeschen
polys[poly2min]=polys.back(); polys.pop_back();
}
return polys[0];
}

void Teile(const CPolygon& zuTeilen,int teilpkt1,int teilpkt2,CPolygon& Teil1,CPolygon&
Teil2){
    bool zuteill=true; //Gehört der aktuell untersuchte Punkt zu Teil1?
    for(int index=0;index<zuTeilen.AnzahlPunkte();index++){
        if(index==teilpkt1||index==teilpkt2){
            Teil1.Add(zuTeilen.Punkt(index)); //Dieser Punkt gehört zu beiden Polygonen
            Teil2.Add(zuTeilen.Punkt(index));
            zuteill=!zuteill;
        }else{
            if(zuteill)Teil1.Add(zuTeilen.Punkt(index));
            else Teil2.Add(zuTeilen.Punkt(index));
        }
    }
}

vector<CPolygon> Trianguliere(const CPolygon& poly){ //poly wird nicht verändert; Rückgabe der
Dreiecke
    vector<CPolygon> Dreiecke; //Feld für Dreiecke
    Dreiecke.push_back(poly);
    //Polygon Immer weiter Teilen, bis nur noch Dreiecke da sind
    int teil=0;
    while(teil<Dreiecke.size()){
        //Ist das Polygon schon ein Dreieck?
        if(Dreiecke[teil].AnzahlPunkte(>3){
            //Punkt mit minimalem y suchen(Vbdg der zwei Nachbarpunkte liegt nie außerhalb
            int imin=200;
            float ymin=200;
            for(int i=0;i<Dreiecke[teil].AnzahlPunkte();i++){
                if(Dreiecke[teil].Punkt(i).Y<ymin){

```

```

        ymin=Dreiecke[teil].Punkt(i).Y;
        imin=i;
    }
}
//Höchster Punkt nun bei Dreiecke[teil].Punkt(imin); Y=ymin

//Nachbarpunkte Verbinden
int indexlinks=(imin-1);
int indexrechts=(imin+1);
if(imin==0)indexlinks=Dreiecke[teil].AnzahlPunkte()-1;
if(imin==Dreiecke[teil].AnzahlPunkte()-1)indexrechts=0;
CLinie vbdg(Dreiecke[teil].Punkt(indexlinks),Dreiecke[teil].Punkt(indexrechts));
if(Dreiecke[teil].Schnitt(vbdg)){
    //Polygon Teilen bei Punkt mit größter Entfernung zur Verbindung
    //Auf welcher Seite der Gerade liegt der Hoechste Punkt?
    float d0=vbdg.Abstand(Dreiecke[teil].Punkt(imin));
    //Am weitesten Entfernten Punkt auf gleicher Seite wie hoechster Punkt finden
    float dmax=0;
    int imax=0;
    for(int i=0;i<Dreiecke[teil].AnzahlPunkte();i++){
        float d=vbdg.Abstand(Dreiecke[teil].Punkt(i));
        if(i!=imin){
            if(abs(d0-d)<abs(d0-dmax)){
                //vbdg muss innerhalb des Plygons liegen
                CLinie vbdg0,vbdg1;
                vbdg0.SetStart(Dreiecke[teil].Punkt(i));
                vbdg0.SetEnde(vbdg.Punkt(0));
                vbdg1.SetStart(Dreiecke[teil].Punkt(i));
                vbdg1.SetEnde(vbdg.Punkt(1));
                if(!(Dreiecke[teil].Schnitt(vbdg0)
                    || Dreiecke[teil].Schnitt(vbdg1)))
                {
                    dmax=d;
                    imax=i;
                }
            }
        }
    }
    //Polygon Teilen zwischen Punkt imin und imax
    CPolygon Teil1,Teil2;
    Teile(Dreiecke[teil],imin,imax,Teil1,Teil2);
    Dreiecke[teil]=Dreiecke.back();Dreiecke.pop_back();
    Dreiecke.push_back(Teil1);
    Dreiecke.push_back(Teil2);
}else{
    //Polygon durch Vbdg teilen
    CPolygon Teil1,Teil2;
    Teile(Dreiecke[teil],indexlinks,indexrechts,Teil1,Teil2);
    Dreiecke[teil]=Dreiecke.back();Dreiecke.pop_back();
    Dreiecke.push_back(Teil1);
    Dreiecke.push_back(Teil2);
}

    teil=(-1);//Von vorne beginnen
}
    teil++;
}

return Dreiecke;
}

void Faerben(CPolygon& Rand,vector<CPolygon>& Triangulierung){
    //Setze Triangulierung[0] farbig
    for(int i=0;i<3;i++)Triangulierung[0].SetFaerbung(i,i+1);

    //Suche ungefärbtes Dreieck, welches 1 gemeinsame Seite mit einem gefärbten hat
    for(int u=0;u<Triangulierung.size();u++)
        if(!Triangulierung[u].Faerbung(0)){//ein ungefärbtes reicht-->Entweder eines oder alle
            //Ungefärbtes Dreieck mit index 'u' gefunden
            //Passendes gefärbtes finden
            for(int f=0;f<Triangulierung.size();f++)
                if(Triangulierung[f].Faerbung(0)){//ein gefärbtes reicht-->Entweder eines oder alle
                    //farbiges Dreieck gefunden
                    //gemeinsame Linie zw Dreieck 'u' und 'f' finden
                    for(int ulinie=0;ulinie<3;ulinie++)
                        for(int flinie=0;flinie<3;flinie++)
                            if(Triangulierung[u].Linie(ulinie).comp(Triangulierung[f].Linie(flinie))){

```

```

//Die Linie darf maximal mit einer Außenkante des Polygons identisch sein
int gleicherrand=0;
for(int randlinie=0;randlinie<Rand.AnzahlPunkte();randlinie++)
    if(Rand.Linie(randlinie).comp(Triangulierung[u].Linie(ulinie)))
        gleicherrand++;

if(gleicherrand==0){//Bei weniger als 1 gleichen Randlinie Dreieck färben
    //Dreieck färben
    //Gleiche Punkte suchen und färben
    for(int upkt=0;upkt<3;upkt++)
        for(int fpkt=0;fpkt<3;fpkt++)
            if(comp(Triangulierung[u].Punkt(upkt),
                    Triangulierung[f].Punkt(fpkt))){
                Triangulierung[u].SetFaerbung(upkt,
                    Triangulierung[f].Faerbung(fpkt));
            }
    //farblosten Punkt suchen und übrige Farbe geben
    int upkt=-1;
    while(Triangulierung[u].Faerbung(++upkt));
    do{
        Triangulierung[u].SetFaerbung(upkt,
            Triangulierung[u].Faerbung(upkt)+1);
    }while(Triangulierung[u].Faerbung(0)==Triangulierung[u].Faerbung(1)
        || Triangulierung[u].Faerbung(0)==Triangulierung[u].Faerbung(2)
        || Triangulierung[u].Faerbung(1)==Triangulierung[u].Faerbung(2));
    //Dreieck gefärbt
    //Suche von vorn beginnen
    u=0;f=Triangulierung.size();
    ulinie=flinie=3;
    }
    }//Die gleiche Linie?
    }//Gefärbtes Dreieck finden
    }//Ungefärbtes Dreieck finden
}

```

### Polygon.h

```

//Klasse zum Verwalten/Zeichnen von Polygonen.
#ifndef polygon_h
#define polygon_h polygon_h

#include <windows.h>
#include <cmath>
#include "punkt.h"
#include "linie.h"
#include <vector>
using namespace std;

class CPolygon{
public:
    CPolygon();
    void Loeschen(){Punkte.clear();}
    void Add(PUNKT);
    void Ausgefüllt(bool flag);
    void Fuellfarbe(COLORREF farbe);
    int Faerbung(int n){return Faerbungen[n];}
    void SetFaerbung(int n, int c){Faerbungen[n]=c;}
    PUNKT Punkt(UINT nr) const {return Punkte[nr];}
    CLinie Linie(UINT nr) const;//Linie von Punkt nr zu Punkt nr+1
    UINT AnzahlPunkte() const {return Punkte.size();}
    bool Schnitt(CLinie mitLinie);// Schneidet eine Linie das Museum?
    bool Ausserhalb(PUNKT p);// Liegt p Ausserhalb vom Polygon?
protected:
    bool ausgefüllt;
    COLORREF fuellfarbe;
    vector<PUNKT> Punkte;
    vector<int> Faerbungen;
};

#endif //polygon_h

```



**Polygon.cpp**

```
//Implementation der Klasse zum Verwalten/Zeichnen von Polygonen.
#include "polygon.h"

CPolygon::CPolygon(){
    Punkte.clear();
    ausgefuellt=true;
    fuellfarbe=0xFFFF00FF;
    Faerbungen.clear();
    Faerbungen.push_back(0);
    Faerbungen.push_back(0);
    Faerbungen.push_back(0);
}

void CPolygon::Add(PUNKT denPunkt){
    Punkte.push_back(denPunkt);
}

void CPolygon::Ausgefuehlt(bool flag){
    ausgefuellt=flag;
}

void CPolygon::Fuellfarbe(COLORREF farbe){
    fuellfarbe=farbe;
}

CLinie CPolygon::Linie(UINT nr)const{//Linie von Punkt nr zu Punkt nr+1
    return CLinie(Punkte[nr],Punkte[(nr+1)%Punkte.size()]);
}

bool CPolygon::Schnitt(CLinie mitLinie){
    bool schnitt=false;
    //Schnitt mit Rand
    for(int a=0;a<AnzahlPunkte();a++){//Jeden LinienAbschnitt untersuchen
        if(mitLinie.Schnitt(Linie(a))){schnitt=true; //schneidet die Linie den Pfadabschnitt?
        if(schnitt)break;
        }
    }
    return schnitt;
}

bool CPolygon::Ausserhalb(PUNKT p){
    //Schnittpunkte zaehlen, die eine horizontale Linie durch p mit dem Polygon hat
    // Gerade Zahl: p liegt au erhalb
    // Ungerade Zahl: p liegt innerhalb

    //Startpunkt der horiz. Linie muss au erhalb vom Polygon liegen
    PUNKT StartPkt;//Startpunkt der Linie
    for(int n=0;n<AnzahlPunkte();n++){if(Punkte[n].X<StartPkt.X)StartPkt.X=Punkte[n].X-1;
    StartPkt.Y=p.Y;
    CLinie horizLinie(StartPkt,p);
    int schnitte=0;
    for(int n=0;n<AnzahlPunkte();n++){
        if(horizLinie.Schnitt(Linie(n))){schnitt++;
    }
    return !(schnitt%2);
}
```

## (c) Unterlagen zur Aufgabe 3: Wer wird Weltmeister?

Bei der Bearbeitung der folgenden Aufgaben wird davon ausgegangen, dass die ‚beliebte Mannschaftssportart‘ Fußball ist, da diese Tatsache für einige Begründungen durchaus von Bedeutung ist.

### 1. Entwickle eine Schätzfunktion für Spielergebnisse.

#### Lösungsidee:

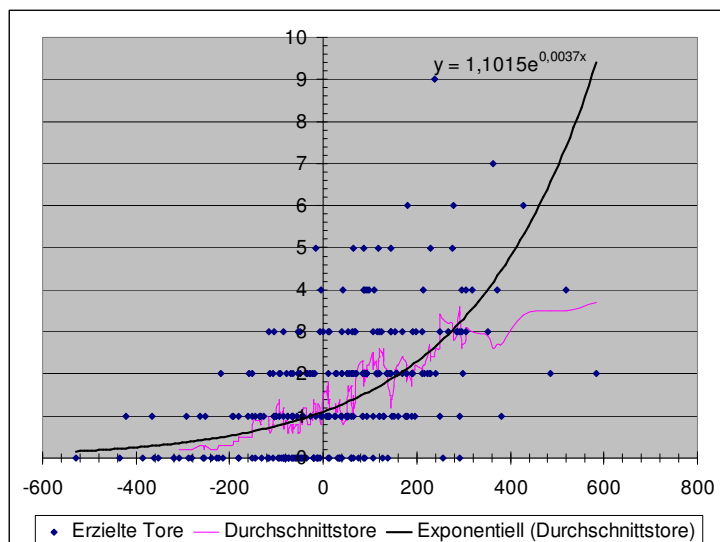
Grundsätzliches: Um Spielergebnisse möglichst einfach aber trotzdem realistisch schätzen zu können ist die Bildung eines guten Modells vonnöten: In meinem Modell startet jede Mannschaft pro Spiel eine gewisse Anzahl von Angriffen, die jeweils mit einer bestimmten – von der Weltranglistenposition abhängigen – Wahrscheinlichkeit zum Torerfolg führen. Als Maß für den Unterschied zwischen zwei Teams wird die Differenz der Weltranglistenpunkte genommen.

Wie viele Tore eine Mannschaft in einem Spiel erzielt kann demnach als Bernoulli-Experiment angesehen werden, für das nur noch die Kettenlänge  $n$  und die Trefferwahrscheinlichkeit  $p$  bestimmt werden muss.

Bestimmung der Parameter: Um möglichst realistische Parameter zu finden, und um herauszufinden wie diese von der Punktdifferenz in der Weltrangliste abhängt, habe ich knapp 300 von der FIFA registrierte Spiele (in der Excel-Datei „Auswertung der FIFA-Spiele.xls“ auf CD) ausgewertet.

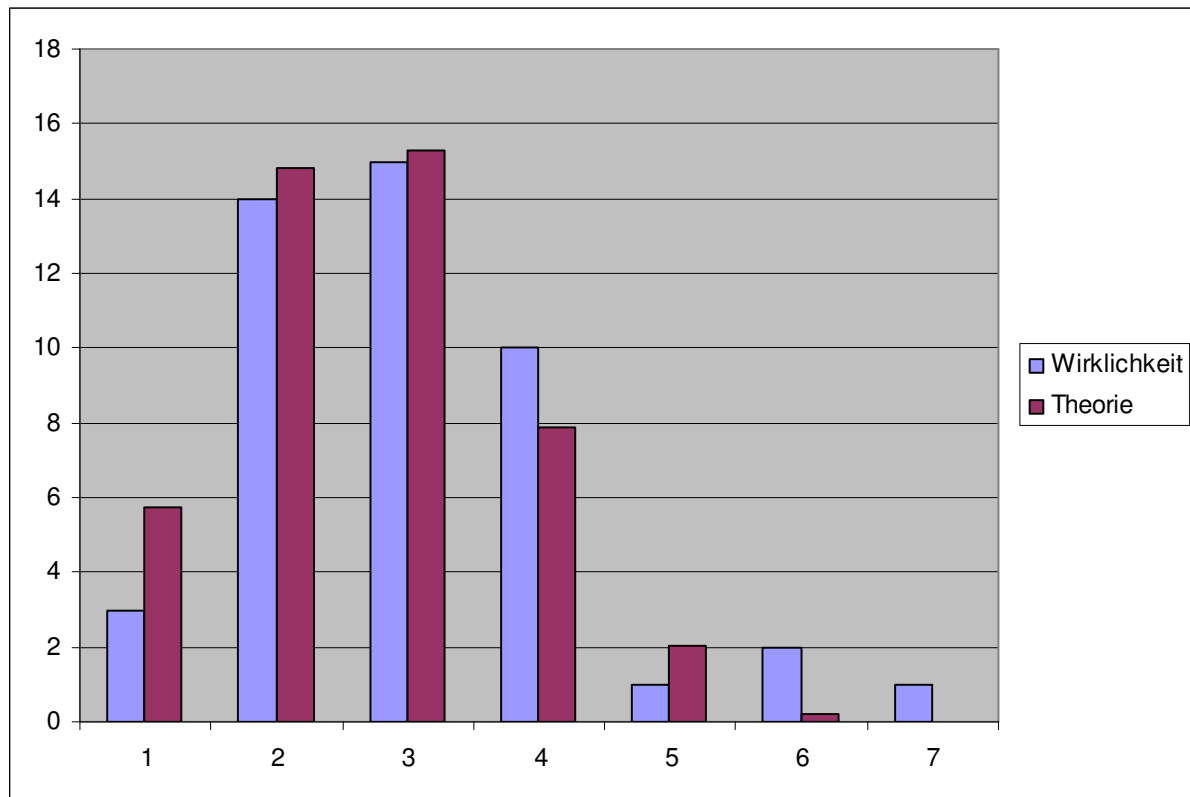
Team1	Team2	Erg	Weltranglistenpunkte		
			Eig. Pkt.	Gegn. Pkt.	Diff.
San Marino	Spanien	0 6	237	765	-528
Dschibuti	Sudan	0 4	46	482	-436
Kasachstan	Dänemark	1 2	314	735	-421
Kambodscha	Singapur	0 2	95	482	-387
Bolivien	Brasilien	1 1	470	837	-367
Armenien	Niederlande	0 1	430	791	-361
Dschibuti	Äthiopien	0 6	46	398	-352
Hongkong	Kroatien	0 4	382	701	-319
Vereinigte Arabische Emirate	Brasilien	0 8	518	837	-319
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

In einer Tabelle hielt ich die Begegnungen zwischen verschiedenen Teams fest (die Daten hierzu stammen von „<http://www.fifa.com/de/mens/results/index.html>“). Nach Errechnen der Punktdifferenz erstellte ich ein Diagramm, welches die Anzahl der erzielten Tore über der Punktdifferenz nach Weltrangliste anträgt. Pink ist dabei die Anzahl der durchschnittlich erzielten Tore bei einer gewissen Punktdifferenz.



Die durchschnittlich pro Partie erzielten Tore (pink) können als Exponentialfunktion der Form  $y=1,1015e^{0,0037x}$  angenähert werden (im Diagramm schwarz). Y ist dabei die Zahl der Tore die die Mannschaft durchschnittlich in einem Spiel gegen einen Gegner mit der Punktdifferenz x erzielt.

Nun soll noch eine Zufallskomponente dafür sorgen, dass nicht jedes Spiel mit der Durchschnittstoranzahl beendet wird. Also betrachtete ich, wie die Toranzahl bei einer Punktdifferenz von ~150 verteilt ist. Die Verteilung aus 43 Spielen ist in folgendem Diagramm blau dargestellt. Bei dieser Punktdifferenz werden 2 beziehungsweise 3 Tore ungefähr gleich oft (14 und 15 mal) erzielt. Die Häufigkeiten für andere Toranzahlen sind dem Diagramm zu entnehmen.



Die Anzahl der Tore ist binomial verteilt (→ Bernoulli-Experiment) mit Erwartungswert  $\mu = 1,1015e^{0,0037x}$  (→ Anzahl der pro Partie durchschnittlich erzielten Tore). Um für die Parameter  $n$  und  $p$  passende Werte zu finden, wurde die Kettenlänge  $n$  gesucht, bei der die Binomialverteilung mit Trefferwahrscheinlichkeit  $p = \mu/n$  am besten die Torverteilung aus obigem Diagramm annähert. Es ergab sich für  $n=5$  die beste Übereinstimmung mit der Realität (Im Diagramm rot dargestellt). Diese Zahl stimmt auch mit der Torverteilung bei einer Punktdifferenz von -300 überein, die Anzahl der Angriffe pro Partie bleibt also unabhängig von den Spielstärken.

Zusammengefasst sieht mein Modell so aus: Die Anzahl der erzielten Tore ist Binomialverteilt mit Kettenlänge  $n=5$  (Es werden also pro Spiel 5 ernsthafte Torchancen herausgearbeitet) und Trefferwahrscheinlichkeit  $p = 1/5 * 1,1015e^{0,0037x}$ , wobei  $x$  die Differenz der Weltranglistenpunkte darstellt.

Zum Schätzen des Spielergebnisses wird für beide Teams das Bernoulli-Experiment je einmal durchgeführt. Dabei muss auch darauf geachtet werden, ob es sich um ein Gruppen- oder ein KO-Spiel handelt, bei letzterem wird bei Gleichstand nach jeweils 5 Angriffen eine Verlängerung gespielt (jeweils zwei Angriffe pro Team). Wenn danach noch kein Sieger ermittelt werden konnte, wird ein Elfmeterschießen durchgeführt: 5 Angriffe pro Team mit jeweils 80% Trefferwahrscheinlichkeit, da es beim Elferschießen nicht mehr auf Können sondern nur noch auf Glück ankommt.

### Dokumentation:

Für das Schätzen eines Spielergebnisses ist die Klasse *CSpiel* zuständig, insbesondere ihre Methoden *float getP(int pkt\_diff)* und *Spielen()*. *getP()* errechnet die Trefferwahrscheinlichkeit aus der Punktdifferenz nach Weltrangliste, *Spielen()* führt das Bernoulli-Experiment durch.

### Programmablaufprotokoll:

Hier werden beispielhaft einige Spiele gezeigt, die die Verlängerungs- und Elfmeterfunktion des Programms verdeutlichen:

```
Portugal - Costa Rica 1:1 -> 2:1 n.V.
Portugal - Tschechien 1:1 -> 2:2 n.V. -> 6:7 n.E.
Tschechien - Brasilien 0:0 -> 0:0 n.V. -> 13:12 n.E.
```

Nun einige Beispiele, wie die Ergebnisse ausfallen, wenn ein Team deutlich besser ist, man sieht, dass dieses Team meist gewinnt:

Tschechien - Ghana 1:1  
 Tschechien - Ghana 2:0  
 Tschechien - Ghana 2:0  
 Tschechien - Ghana 3:1  
 Tschechien - Ghana 1:1

Bei ungefähr gleichstarken Teams fallen die Ergebnisse nicht so einseitig aus:

Deutschland und Polen liegen nach Weltrangliste nur 14 Punkte auseinander und es gewinnt mal das eine Team, mal das andere, Teilweise gehen die Spiele unentschieden aus:

Deutschland - Polen 3:3  
 Deutschland - Polen 0:1  
 Deutschland - Polen 1:1  
 Deutschland - Polen 2:0  
 Deutschland - Polen 1:0

### Anmerkung:

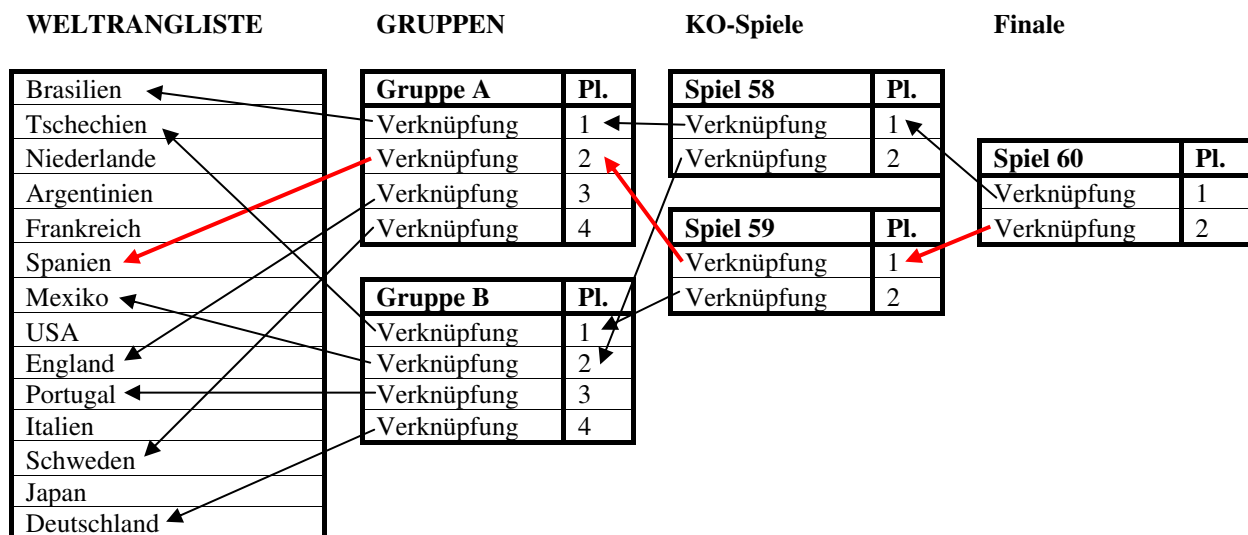
Meine Schätzfunktion ist zwar keine Erweiterung, da sie ja von der Aufgabenstellung gefordert wird, aber ich finde, dass sie trotzdem einen Pluspunkt verdient hat, da die Überlegungen, die der Bildung des Modells für die Schätzfunktion zugrunde lagen, anhand von realen Spielergebnissen erhärtet und ausführlich erörtert wurden.

## 2. Das Simulationssystem muss außer der Weltrangliste mit den Angaben für die am Turnier beteiligten Teams auch den Turnierplan einlesen können.

### Lösungsidee:

Eine Spielpaarung besteht aus Verknüpfungen zu Teilnehmern von vorherigen Spielen oder Gruppen. Die Verknüpfungen werden beim Simulieren des Spiels ausgewertet und dabei festgestellt, welche Teams gegeneinander antreten. Verknüpfungen können auf vorherige Spiele (Erster oder Zweiter), auf Gruppen (mit Platzierung) oder einfach auf die Weltrangliste zeigen, wenn das Team direkt definiert wurde. So enthält nur die Weltrangliste die Informationen über die Teams, alles andere wird nur über Zeiger auf vorherige Spiele abgewickelt. Zur Dokumentation der Funktionalität wird das Programm die WM einmal simulieren und diese Simulation ausgeben.

Prinzipiell kann der Aufbau des Turniers so dargestellt werden (Nur als Beispiel und Auszugsweise):



Wenn nun zur Berechnung des Finales herausgefunden werden soll, welches Team dort antritt, zeigt die Verknüpfung für den zweiten Finalteilnehmer auf den Sieger aus Spiel 59, dieser war der zweite aus Gruppe A, welcher das Team Spanien aus der Weltrangliste war.

## Dokumentation:

Zur Umsetzung der Lösungsidee wurden folgende Klassen und Datenstrukturen verwendet:

-*STeam*: Datenstruktur, speichert zu einem Teamname seine Stärke in der Weltrangliste und hält noch eine Variable für Sortieraufgaben frei.

-*CSpielplan*: Hauptklasse, die die Weltrangliste(ein Feld aus *STeams*) sowie alle Gruppen und KO-Spiele speichert, den Turnierplan anzeigen und die WM simulieren wird.

-*CGruppe*: Stellt eine Gruppe dar, deren Mitglieder als Verknüpfungen( $\rightarrow$ *CLink*) gespeichert werden. Der Spezialfall einer *CGruppe* mit 2 Mitgliedern wird als KO-Spiel interpretiert, dessen Sieger als erstplatzierter der ‚Gruppe‘ abgefragt werden kann

Die Klasse erstellt den Spielplan(als Feld aus *CSpielen*) für die Gruppe (jeder spielt genau einmal gegen jeden). Nachdem die Spiele simuliert wurden( $\rightarrow$ *CSpiel*) sortiert die Klasse die Mannschaften nach den angegebenen Kriterien: Die Funktion *Sortiere(sortkat,start,ende)* sammelt je nach übergebenem Parameter *sortkat* (=Sortierkategorie) für die an den Positionen *start* bis *ende* befindlichen Teams die relevanten Informationen (Punkte insgesamt, Punkte aus den Direktbegegnungen,...) und sortiert sie anschließend nach diesen Werten.

-*CLink*: Verknüpfungen werden in der Klasse *CLink* realisiert: Ein *CLink* ist entweder ein Zeiger auf einen Platz in einer Gruppe, oder, falls dieser Zeiger nicht existiert auf ein Team aus der Weltrangliste. Das Abfragen der *CLinks* erfolgt rekursiv, bis ein Team der Weltrangliste zurückgegeben werden kann.

-*CSpiel*: Stellt Funktionen zum Simulieren eines Spiels zwischen zwei Mannschaften dar ( $\rightarrow$ Aufgabe 1)

Beim Einlesen der Weltrangliste wird einfach das Feld *WRListe* des Spielplan-Objekts *spielplan* gefüllt.

Die Syntax der Datei „wm-spielplan.txt“ wurde verändert, ohne dass Informationen verloren gehen: „<gruppen phase=1>“ wurde beispielsweise gekürzt als „<gruppen 1“, und die abschließenden Tags(„</spiel>“) wurden komplett entfernt. Zum Einlesen des Spielplans werden je nach geöffnetem „Tag“ Eigenschaften festgesetzt und dann *CGruppen* mit diesen Eigenschaften erstellt und dem Spielplan-Objekt hinzugefügt.

Das Programm liest zunächst die Datei „rangliste.txt“ ein und zeigt diese auf Anforderung an. Danach kommt eine Abfrage Aufgabe 3 betreffend, anschließend wird „spielplan.txt“ eingelesen und kann ebenfalls angezeigt werden. Schlussendlich kann man die Weltmeisterschaft einmal durchspielen lassen, dabei wird auch die Platzierungsfindung in der Gruppenphase begründet.

## Programmablaufprotokoll:

Zunächst ein Programmdurchlauf mit dem Beispielplan. Besondere Ereignisse wurden hervorgehoben und kommentiert. Unwichtigere Stellen wurden gekürzt. Danach wird das Programm mit einem beispielhaften Spielplan für die Champions League der Vereinsmannschaften getestet.

```
Rangliste 'rangliste.txt' auslesen...
Rangliste anzeigen? (J/N) j
Brasilien: 841 Pkt.
Tschechien: 796 Pkt.
Niederlande: 791 Pkt.
Argentinien: 774 Pkt.
...
Togo: 582 Pkt.
Angola: 570 Pkt.
```

Anzeige der Weltrangliste

```
...
Spielplan 'spielplan.txt' auslesen...
Spielplan anzeigen? (J/N) j
```

```
S P I E L P L A N:
~~~~~
Gruppenphase 1:
```

Anzeige des Spielplans mit allen wichtigen Informationen

```
Gruppe A:
~~~~~
Deutschland
Costa Rica
Polen
Ecuador
...

Gruppe H:
~~~~~
Spanien
Ukraine
Tunesien
Saudi-Arabien
```

Achtelfinale:  
~~~~~  
49: Erster A - Zweiter B  
50: Erster C - Zweiter D  
...  
56: Erster H - Zweiter G

Viertelfinale:  
~~~~~  
57: Erster 49 - Erster 50  
...  
60: Erster 55 - Erster 56

Halbfinale:  
~~~~~  
61: Erster 57 - Erster 58  
62: Erster 59 - Erster 60

Spiel um Platz Drei:  
~~~~~  
63: Zweiter 61 - Zweiter 62

Finale:  
~~~~~  
64: Erster 61 - Erster 62

Einmalige Simulation der WM  
~~~~~  
Gruppenphase 1:  
  
...  
Gruppe B  
~~~~~  
England - Paraguay 1:0  
England - Trinidad und Tobago 1:1  
England - Schweden 1:2  
Paraguay - Trinidad und Tobago 0:1  
Paraguay - Schweden 0:1  
Trinidad und Tobago - Schweden 0:1

Simulation der WM

Rangliste nach Punkte aus allen Spielen:  
Schweden 9  
England 4  
Trinidad und Tobago 4  
Paraguay 0

England und Trinidad punktgleich

Rangliste nach Punkte aus Direktbegegnungen:  
Schweden 9  
England 1  
Trinidad und Tobago 1  
Paraguay 0

Gegeneinander spielten sie unentschieden

Rangliste nach Tordiff. aus Dir.-Beg.:  
Schweden 9  
England 0  
Trinidad und Tobago 0  
Paraguay 0

...  
Rangliste nach Tordiff. gesamt:  
Schweden 9  
England 0  
Trinidad und Tobago 0  
Paraguay 0

Rangliste nach erz. Tore gesamt:  
Schweden 9  
England 3  
Trinidad und Tobago 2  
Paraguay 0

England erzielte insgesamt mehr Tore, ist deshalb vor Trinidad

Rangliste:  
1. Schweden  
2. England  
3. Trinidad und Tobago  
4. Paraguay

## Gruppe C

~~~~~

Argentinien - Elfenbeinküste 0:0  
Argentinien - Serbien-Montenegro 2:0  
Argentinien - Niederlande 2:0  
Elfenbeinküste - Serbien-Montenegro 0:1  
Elfenbeinküste - Niederlande 0:3  
Serbien-Montenegro - Niederlande 0:1

## Rangliste nach Punkte aus allen Spielen:

Argentinien 7  
Niederlande 6  
Serbien-Montenegro 3  
Elfenbeinküste 1

## Rangliste:

1. Argentinien
2. Niederlande
3. Serbien-Montenegro
4. Elfenbeinküste

...

## Gruppe H

~~~~~

Spanien - Ukraine 0:1  
Spanien - Tunesien 3:1  
Spanien - Saudi-Arabien 0:1  
Ukraine - Tunesien 0:1  
Ukraine - Saudi-Arabien 1:1  
Tunesien - Saudi-Arabien 0:1

## Rangliste nach Punkte aus allen Spielen:

Saudi-Arabien 7  
Ukraine 4  
Spanien 3  
Tunesien 3

## Rangliste nach Punkte aus Direktbegegnungen:

Saudi-Arabien 7  
Ukraine 4  
Spanien 3  
Tunesien 0

Beispiel für die Sortierfunktion: Spanien hat gegen Tunesien gewonnen

## Rangliste:

1. Saudi-Arabien
2. Ukraine
3. Spanien
4. Tunesien

## Achtelfinale:

~~~~~

Ecuador - England 1:1 -> 1:1 n.V. -> 4:6 n.E.  
Argentinien - Iran 2:2 -> 3:2 n.V.  
Schweden - Deutschland 0:1  
Mexiko - Niederlande 1:1 -> 1:3 n.V.  
Tschechien - Australien 3:1  
Frankreich - Ukraine 3:4  
Brasilien - USA 1:1 -> 2:1 n.V.  
Saudi-Arabien - Schweiz 1:0

## Viertelfinale:

~~~~~

England - Argentinien 1:2  
Tschechien - Ukraine 2:1  
Deutschland - Niederlande 0:2  
Brasilien - Saudi-Arabien 1:1 -> 2:2 n.V. -> 6:7 n.E.

## Halbfinale:

~~~~~

Argentinien - Tschechien 0:3  
Niederlande - Saudi-Arabien 3:0

## Spiel um Platz Drei:

~~~~~

Argentinien - Saudi-Arabien 3:1

## Finale:

~~~~~

Tschechien - Niederlande 1:0

**Erweiterung:**

Da das Programm sehr allgemein ausgelegt ist, kann es auch mit anderen Spielplänen umgehen, so zum Beispiel mit dem der Champions League (CL) der Vereinsmannschaften. Wichtig ist hierbei nur eine passende Rangliste. Ich habe als Beispiel die CL ein wenig verkürzt: Es werden die Ligen der vier Länder Deutschland, Spanien, Italien und England simuliert, wobei jeweils die drei besten aus jedem Land für die Gruppenphase der CL qualifiziert sind. Die Gruppenphase ‚meiner‘ CL besteht aus 4 Gruppen á 4 Mannschaften. Die übrigen 4 Plätze werden über eine Qualifikationsrunde ausgespielt, in der die viert- und fünftplatzierten jedes Landes im KO-System um die CL-Plätze kämpfen. Der Spielplan sieht dann(ohne die Nationalligen) folgendermaßen aus:

```

S P I E L P L A N:
~~~~~
Qualifikationsrunde:
~~~~~
1: Vierter Deutschland - Fuenfter England
2: Vierter Italien - Fuenfter Deutschland
3: Vierter England - Fuenfter Spanien
4: Vierter Spanien - Fuenfter Italien
Gruppenphase 2:

Gruppe A:
~~~~~
Erster Deutschland
Zweiter England
Dritter Spanien
Erster 2

Gruppe B:
~~~~~
Zweiter Deutschland
Dritter England
Erster Italien
Erster 4

Gruppe C:
~~~~~
Dritter Deutschland
Zweiter Spanien
Zweiter Italien
Erster 3

Gruppe D:
~~~~~
Erster England
Dritter Italien
Erster Spanien
Erster 1

Viertelfinale:
~~~~~
5: Erster A - Zweiter B
6: Erster C - Zweiter D
7: Erster B - Zweiter A
8: Erster D - Zweiter C

Halbfinale:
~~~~~
9: Erster 5 - Erster 6
10: Erster 8 - Erster 7

Spiel um Platz Drei:
~~~~~
11: Zweiter 9 - Zweiter 10

Finale:
~~~~~
12: Erster 9 - Erster 10

```

**Hier der Programmablauf:**

```

Gruppe Deutschland
~~~~~
...
Rangliste:
  1. Schalke
  2. Bayern Muenchen

```



3. Hamburg  
4. Bremen  
5. Berlin

...  
18. Koeln

Gruppe England  
~~~~~

Rangliste:

1. Chelsea London  
2. Manchester United  
3. FC Liverpool  
4. Bolton Wanderers  
5. Tottenham

...  
20. Sunderland

Gruppe Italien  
~~~~~

Rangliste:

1. Juventus Turin  
2. Inter Mailand  
3. AC Mailand  
4. Florenz  
5. AS Rom

...  
20. Treviso

Gruppe Spanien  
~~~~~

Rangliste:

1. Real Madrid  
2. FC Barcelona  
3. Osasuna  
4. CF Valencia  
5. Sevilla

...  
20. Malaga

Qualifikation:  
~~~~~

Bremen - Tottenham 1:2  
Florenz - Berlin 5:1  
Bolton Wanderers - Sevilla 1:0  
CF Valencia - AS Rom 1:1 -> 1:1 n.V. -> 5:6 n.E.

Gruppenphase 2:

Gruppe A

~~~~~

Florenz - Manchester United 1:1  
Florenz - Osasuna 3:0  
Florenz - Schalke 1:0  
Manchester United - Osasuna 2:0  
Manchester United - Schalke 2:1  
Osasuna - Schalke 1:1

Rangliste:

1. Florenz  
2. Manchester United  
3. Schalke  
4. Osasuna

Gruppe B

~~~~~

Juventus Turin - Bayern Muenchen 2:2  
Juventus Turin - AS Rom 3:1  
Juventus Turin - FC Liverpool 3:2  
Bayern Muenchen - AS Rom 1:0  
Bayern Muenchen - FC Liverpool 2:1  
AS Rom - FC Liverpool 1:2

Rangliste:

1. Juventus Turin  
2. Bayern Muenchen  
3. FC Liverpool  
4. AS Rom

## Gruppe C

~~~~~

Hamburg - FC Barcelona 1:2  
 Hamburg - Inter Mailand 1:1  
 Hamburg - Boloton Wanderers 2:2  
 FC Barcelona - Inter Mailand 1:1  
 FC Barcelona - Boloton Wanderers 2:0  
 Inter Mailand - Boloton Wanderers 2:1

## Rangliste:

1. FC Barcelona
2. Inter Mailand
3. Hamburg
4. Boloton Wanderers

## Gruppe D

~~~~~

Chelsea London - Tottenham 2:0  
 Chelsea London - Real Madrid 3:0  
 Chelsea London - AC Mailand 1:1  
 Tottenham - Real Madrid 0:2  
 Tottenham - AC Mailand 1:1  
 Real Madrid - AC Mailand 0:1

## Rangliste:

1. Chelsea London
2. AC Mailand
3. Real Madrid
4. Tottenham

## Viertelfinale:

~~~~~

Florenz - Bayern Muenchen 2:2 -> 3:2 n.V.  
 FC Barcelona - AC Mailand 1:2  
 Juventus Turin - Manchester United 3:1  
 Chelsea London - Inter Mailand 3:0

## Halbfinale:

~~~~~

AC Mailand - Florenz 2:1  
 Juventus Turin - Chelsea London 2:2 -> 2:2 n.V. -> 12:11 n.E.

## Spiel um Platz Drei:

~~~~~

Chelsea London - Florenz 5:0

## Finale:

~~~~~

Juventus Turin - AC Mailand 3:2

### 3. Gestalte dein Simulationssystem so flexibel, dass auch andere Faktoren [...] bei der Schätzung von Spielergebnissen berücksichtigt werden können.

#### Lösungsidee:

Die Schätzung der Spielergebnisse hängt nur von der Torerfolgswahrscheinlichkeit  $p$  (siehe Aufgabe 1) ab. Diese ist nur von der Differenz der Weltranglistenpunkte abhängig. Also kann zur Berücksichtigung weiterer Faktoren einfach die Punktzahl in der Weltrangliste „manipuliert“ werden.

#### Dokumentation:

Im Programm wird die Manipulation durch Eingabe von Prozentwerten realisiert: Eingabe von „Deutschland 115“ verbessert Deutschland um 15%, während „Brasilien 80“ die Form der Brasilianer um 20% sinken lässt. Zudem sind vorgefertigte Faktoren mit ihren zugehörigen Prozentzahlen ins Programm eingebaut, die ausgewählt werden können.

#### Programmablaufprotokoll:

##### F L E X I B I L I T Ä T

~~~~~

Bitte Faktoren auswählen, die die Stärken der Mannschaften beeinflussen.

SYNTAX: <Mannschaftsname> <Änderung in Prozent>

Bsp.: 'Brasilien 90' verschlechtert Brasilien um 10 Prozent.

Es sind auch mehrere Faktoren für eine Mannschaft zulässig.

Durch Eingabe der Ziffern von 0-9 koennen vorgefertigte Faktoren verwendet werden:

|                                                       |             |
|-------------------------------------------------------|-------------|
| 0 - Heimvorteil Deutschland:                          | +5 Prozent  |
| 1 - Deutschlands Benachteiligung bei Weltrangliste:   | +10 Prozent |
| 2 - Grosse Motivation des 'Fussballzwerchs' Ghana:    | +15 Prozent |
| 3 - Grosse Motivation des 'Fussballzwerchs' Trinidad: | +15 Prozent |
| 4 - Grosse Motivation des 'Fussballzwerchs' Togo:     | +15 Prozent |
| 5 - Grosse Motivation des 'Fussballzwerchs' Angola:   | +15 Prozent |
| 6 -                                                   |             |
| 7 -                                                   |             |
| 8 - Arroganz von Brasilien:                           | -10 Prozent |
| 9 - Holland schlechter:                               | -10 Prozent |

```
>>0
```

```
Deutschland: 710 -> 745
```

```
>>1
```

```
Deutschland: 745 -> 819
```

```
>>8
```

```
Brasilien: 841 -> 756
```

```
>>Spanien 115
```

```
Spanien: 771 -> 886
```

Flexibilität durch  
vorgefertigte Faktoren

Flexibilität durch eigene  
Eingabe

#### 4. Beantworte Fragen nach Spielpaarungen und die Weltmeisterfrage[...]. Gib mögliche Antworten an sowie die Wahrscheinlichkeiten, die dein System für diese Antworten berechnet hat.

##### Lösungsidee:

Es werden zwei Arten von Wahrscheinlichkeiten betrachtet:

-Einerseits die Wahrscheinlichkeit, mit der ein Team eine bestimmte Platzierung in einem Spiel oder einer Gruppe erreicht. Dazu werden zunächst rekursiv alle Vorgängerspiele markiert, die zur Berechnung dieses Spiels notwendig sind. Die markierten Spiele werden simuliert und es wird gezählt, wie oft welche Teams im gesuchten Spiel die gesuchte Platzierung erreichen.

-Zum anderen kann auch die Wahrscheinlichkeit angegeben werden, mit der ein Team bei einem Spiel dabei ist. Dazu werden die Wahrscheinlichkeiten für die Platzierungen, aus denen sich die Spielpaarung zusammensetzt berechnet. Ausgegeben wird dann die Wahrscheinlichste Spielpaarung, sowie die Wahrscheinlichkeiten mit denen andere Teams in diesem Spiel antreten.

##### Dokumentation:

Im Programm muss zunächst die Spielnummer oder der Gruppenname eingegeben werden. Dann kann man die Wahrscheinlichkeit für einen bestimmten Platz erfragen, indem die gesuchte Platzierung eingegeben wird (z.B. „A 2“ für den Zweitplatzierten aus Gruppe A oder „64 1“ für den Sieger von Spiel 64, den Weltmeister).

Alternativ kann nach der Spielnummer auch eine ‚0‘ eingegeben werden, worauf das Programm die wahrscheinlichste Spielpaarung ausgibt.

Zur Realisierung der Lösungsidee sind die Funktionen *Simuliere(string Name, int Platz, int anzahl)* und *Simuliere(string Name, int anzahl)* zuständig.

*Simuliere(string Name, int Platz, int anzahl)* durchsucht zunächst rekursiv den Spielplan und markiert dabei alle Spiele und Gruppen, indem es das öffentliche Attribut *bool mitberechnen* der Gruppen **true** setzt. Diese Gruppen Werden dann in einer Schleife immer wieder simuliert, wobei für jedes Team gezählt wird, wie oft es den gesuchten Platz erreicht hat, indem die *Daten* Eigenschaft jedes Teams inkrementiert wird. Die Teams werden dann absteigend sortiert und die Wahrscheinlichkeiten ausgegeben. **Wahrscheinlichkeit=Häufigkeit/Anzahl der Simulationen**

*Simuliere(string Name, int anzahl)* berechnet die Wahrscheinlichkeiten für eine Spielpaarung: Dazu werden die Vorgängerspiele simuliert (mit obiger Funktion). Die Wahrscheinlichste Spielpaarung ist dann die Begegnung zwischen den Teams, welche am wahrscheinlichsten die Vorgängerplatzierung erreichen. Weiterhin wird noch angegeben, wie wahrscheinlich andere Teams in dieser Begegnung antreten.

##### Programmablaufprotokoll:

Ich werde hier einige Fragen nach Spielpaarungen und Spielergebnissen beantworten, dabei einige scheinbare Paradoxien aufzeigen und die Gründe dafür erklären. Grundlage für die Schätzungen sind dabei immer mind.

50000 Simulationen, bei dieser Versuchszahl schwanken die berechneten Wahrscheinlichkeiten in der Regel um weniger als 1%.

1.) Da ich im schönen Allgäu wohne und das örtlich nahesten WM-Stadion in München steht, möchte ich wissen, wer im Achtelfinale in der Allianz-Arena spielt! Hierzu lasse ich die Wahrscheinlichkeiten für das Spiel 49 berechnen:

```
Fuer welches Spiel willst du die Wahrscheinlichkeiten wissen?
>>49
Platz eingeben oder 0 fuer Auflistung der Teilnehmer
>>0
Wieviele Berechnungen?
>>50000
Wahrscheinlichste Zusammensetzung:
~~~~~
Erster A:Deutschland mit 32.714 Prozent
Zweiter B:Schweden mit 35.764 Prozent
Wahrscheinlichkeiten fuer alle Teams:
~~~~~
Schweden mit 35.764 Prozent
Deutschland mit 32.714 Prozent
England mit 30.464 Prozent
Costa Rica mit 28.32 Prozent
Polen mit 26.89 Prozent
Paraguay mit 23.94 Prozent
Ecuador mit 12.076 Prozent
Trinidad und Tobago mit 9.832 Prozent
```

Somit sehe ich am wahrscheinlichsten das Spiel Deutschland gegen Schweden, obwohl eine Begegnung zwischen Deutschland und England auch nicht wesentlich unwahrscheinlicher ist. England ist nur

2.) Die Wahrscheinlichkeiten für das Achtelfinalspiel 50 erscheinen zunächst paradox:

```
Wahrscheinlichste Zusammensetzung:
~~~~~
Erster C:Niederlande mit 53.402 Prozent
Zweiter D:Portugal mit 35.652 Prozent
Wahrscheinlichkeiten fuer alle Teams:
~~~~~
Niederlande mit 53.402 Prozent
Argentinien mit 42.206 Prozent
Portugal mit 35.652 Prozent
Mexiko mit 33.248 Prozent
Iran mit 27.74 Prozent
Angola mit 3.36 Prozent
Elfenbeinkueste mit 2.638 Prozent
Serbien-Montenegro mit 1.754 Prozent
```

Der Sieger aus Gruppe C ist am wahrscheinlichsten Niederlande, das zweite Team der Begegnung am wahrscheinlichsten Portugal. Die größten Wahrscheinlichkeiten sind jedoch für Niederlande und Argentinien angegeben. Diese zwei Teams stehen zwar sehr oft in diesem Achtelfinale, die Annahme, dass sich das Spiel am häufigsten aus diesen beiden Teams zusammensetzt ist aber ein Trugschluss, da beide Teams aus Gruppe C kommen und so nie gegeneinander spielen können. Niederlande und Argentinien haben in ihrer Gruppe leichtes Spiel und machen die ersten beiden Plätze nur unter sich aus, während Gruppe D viel ausgeglichener ist, und mit Portugal, Mexiko und dem Iran gleich drei Teams um die Achtelfinalplätze kämpfen.

3.) Nun die Frage aller Fragen: Wer wird Weltmeister??? Dazu lasse ich mein System die Wahrscheinlichkeiten für einen Sieg im Spiel 64 (dem Finale) zur Sicherheit zweimal ausrechnen. Die Anzahl der Simulationen ist so gewählt, dass die Schwankungen i. d. R. nicht mehr als 0,1% betragen:

|                             |                             |
|-----------------------------|-----------------------------|
| Brasilien: 26.438 Prozent   | Brasilien: 26.462 Prozent   |
| Niederlande: 11.266 Prozent | Niederlande: 11.308 Prozent |
| Tschechien: 10.152 Prozent  | Tschechien: 10.224 Prozent  |
| Frankreich: 7.96 Prozent    | Frankreich: 7.96 Prozent    |
| Argentinien: 7.906 Prozent  | Argentinien: 7.854 Prozent  |
| Spanien: 6.99 Prozent       | Spanien: 6.908 Prozent      |
| Mexiko: 5.62 Prozent        | Mexiko: 5.44 Prozent        |
| England: 5.2 Prozent        | England: 5.04 Prozent       |
| USA: 4.47 Prozent           | USA: 4.724 Prozent          |
| Portugal: 3.968 Prozent     | Portugal: 3.874 Prozent     |
| Schweden: 2.344 Prozent     | Schweden: 2.402 Prozent     |
| Italien: 1.874 Prozent      | Italien: 2.016 Prozent      |
| Sonstige: 1.46 Prozent      | Sonstige: 1.404 Prozent     |
| Deutschland: 1.06 Prozent   | Deutschland: 1.114 Prozent  |

Somit wird – welche Überraschung – der Weltranglistenerste Brasilien am wahrscheinlichsten (in einem von vier Fällen) Weltmeister. Dies deckt sich auch gut mit der WM-Geschichte, wenn man annimmt, dass Brasilien immer die vergleichsweise beste Mannschaft war. In bisher 17 Weltmeisterschaften wurde Brasilien 4 mal Weltmeister.

Auffallend ist auch, dass Tschechien weniger wahrscheinlich Weltmeister wird die Niederlande, obwohl Tschechien in der Weltrangliste geringfügig besser als Holland ist. Dies liegt an der durch den Turnierplan vorgegebenen schlechteren Ausgangsposition für Tschechien, die im Viertelfinale eventuell schon auf Brasilien treffen und dort eher schlechte Chancen haben.

Die Wahrscheinlichkeit für Deutschland ist allerdings für meinen Geschmack mit 1% zu niedrig bemessen.

4.) Aus diesem Grund werde ich die Weltmeisterschaft noch einmal simulieren, und zwar unter Berücksichtigung der Tatsache, dass Deutschland für Freundschaftsspiele weniger Weltranglistenpunkte erhielt und zudem noch den nicht zu unterschätzenden Heimvorteil besitzt. Dies wird über das Flexibilitätssystem aus Aufgabe 3 realisiert.

```
Brasilien: 24.18 Prozent
Deutschland: 16.138 Prozent
Tschechien: 9.152 Prozent
Niederlande: 9.072 Prozent
Frankreich: 6.708 Prozent
Argentinien: 6.264 Prozent
Spanien: 6.002 Prozent
Mexiko: 4.752 Prozent
USA: 4.056 Prozent
England: 3.772 Prozent
Sonstige: 2.96 Prozent
Portugal: 2.946 Prozent
Italien: 1.674 Prozent
Schweden: 1.582 Prozent
Japan: 0.742 Prozent
```

Dieses Ergebnis stimmt schon zuversichtlicher für die kommende Weltmeisterschaft!!

5.) Zusammensetzung einer CL-Gruppe: Das System berechnet natürlich auch für meine kleine CL die Wahrscheinlichkeiten, hier z.B. wie eine Gruppe zusammengesetzt wird:

```
Wahrscheinlichste Zusammensetzung:
~~~~~
Erster Deutschland:Bayern Muenchen mit 73.76 Prozent
Zweiter England:Manchester United mit 73.62 Prozent
Dritter Spanien:CF Valencia mit 29.4 Prozent
Erster 2:Florenz mit 35.76 Prozent
Wahrscheinlichkeiten fuer alle Teams:
~~~~~
Bayern Muenchen mit 73.76 Prozent
Manchester United mit 73.62 Prozent
Florenz mit 35.76 Prozent
CF Valencia mit 29.4 Prozent
AS Rom mit 28.7 Prozent
Real Madrid mit 28.66 Prozent
Osasuna mit 22.6 Prozent
FC Liverpool mit 20.16 Prozent
Hamburg mit 18.98 Prozent
Inter Mailand mit 17.94 Prozent
AC Mailand mit 10.66 Prozent
Celta Vigo mit 7.2 Prozent
Schalke mit 4.78 Prozent
Sevilla mit 4.48 Prozent
Bremen mit 3.74 Prozent
FC Barcelona mit 3.08 Prozent
Villareal mit 2.88 Prozent
Sonstige mit 2.86 Prozent
```

So qualifiziert sich als Deutscher Meister Bayern München am wahrscheinlichsten für Gruppe A, und Florenz schafft es wahrscheinlich über die Qualifikation in die CL-Gruppe A.

6.) Nun wäre noch interessant, wer denn CL-Sieger wird... Die Antwort darauf lautet folgendermaßen:

```
Chelsea London: 44.3 Prozent
Juventus Turin: 35.9 Prozent
AC Mailand: 6.65 Prozent
Inter Mailand: 3.65 Prozent
```

```

Bayern Muenchen: 3.3 Prozent
FC Barcelona: 3 Prozent
Manchester United: 1.5 Prozent
Sonstige: 0.65 Prozent
Florenz: 0.55 Prozent
Hamburg: 0.5 Prozent

```

Das Ergebnis fällt zwar recht einseitig für Chelsea und Juve aus, der Grund dafür liegt in der nicht ganz ausgereiften Rangliste...

### Programmtext:

#### Main.cpp (Auszüge)

```

int main(int argc, char *argv[])
{
    //Spielplan Objekt erzeugen, dient zum Speichern aller aus Dateien gehalten informationen
    CSpielplan plan;

    //Variable zum Speichern aller Eingaben
    string Input;
    //Teamdaten laden
    ifstream Rangliste;//Datei-Objekt
    Rangliste.open("rangliste.txt",ios::binary|ios::in);
    //Fehler beim öffnen??
    if(!Rangliste){
        cerr<<"Rangliste konnte nicht geoeffnet werden";
    }
    cout<<"Rangliste 'rangliste.txt' auslesen..."<<endl;
    //Datei auslesen
    Rangliste >> Input;
    while(!Rangliste.eof()){
        STeam team;//Struktur zum Speichern der Teamdaten
        Rangliste>>Input;//Mannschaftsname einlesen
        team.Name=Input;
        Rangliste>>Input;
        //WR-Punkte bzw. weitere Teile des Namens einlesen
        while(!atoi(Input.c_str())){//solange keine Zahl kommt
            team.Name.append(" ");
            team.Name.append(Input);//Zweiter Teil
            Rangliste>>Input;
        }
        team.Staerke=atoi(Input.c_str());
        plan.AddTeam(team);
        Rangliste >> Input;//Platz in Weltrangliste; irrelevant
    }
    Rangliste.close();
    //Anzeigen
    cout<<"Rangliste anzeigen? (J/N) ";
    cin>>Input;
    if(Input=="J"||Input=="j"){
        plan.WRListeAnzeigen();
    }

    //Verbesserungen, Verschlechterungen auswaehlen
    cout<<endl<<"F L E X I B I L I T A E T"<<endl<<"~~~~~"<<endl;
    cout<<"Bitte Faktoren auswaehlen, die die Staerken der Mannschaften beeinflussen."<<endl
    <<"SYNTAX: <Mannschaftsname> <Aenderung in Prozent>"<<endl
    <<"Bsp.: 'Brasilien 90' verschlechtert Brasilien um 10 Prozent."<<endl
    <<"Es sind auch mehrere Faktoren fuer eine Mannschaft zulaessig."<<endl<<endl
    <<"Durch Eingabe der Ziffern von 0-9 koennen vorgefertigte Faktoren"<<endl
    <<"verwendet werden:"<<endl
    <<"0 - Heimvorteil Deutschland:          +5 Prozent"<<endl
    <<"1 - Deutschlands Benachteiligung bei Weltrangliste:  +10 Prozent"<<endl
    <<"2 - Grosse Motivation des 'Fussballzwerchs' Ghana:  +15 Prozent"<<endl
    <<"3 - Grosse Motivation des 'Fussballzwerchs' Trinidad: +15 Prozent"<<endl
    <<"4 - Grosse Motivation des 'Fussballzwerchs' Togo:    +15 Prozent"<<endl
    <<"5 - Grosse Motivation des 'Fussballzwerchs' Angola:  +15 Prozent"<<endl
    <<"6 - "<<endl
    <<"7 - "<<endl
    <<"8 - Arroganz von Brasilien:              -10 Prozent"<<endl
    <<"9 - Holland schlechter:                 -10 Prozent"<<endl
    <<endl;
    cout<<">>";
    cin>>Input;
    while(!(Input=="e"||Input=="E")){
        if(Input[0]>='0'&&Input[0]<='9'){

```

```

        int aenderung;
        string Teamname;
        switch(atoi(Input.c_str())){
            case 0: aenderung=105;Teamname="Deutschland";break;
            case 1: aenderung=110;Teamname="Deutschland";break;
            case 2: aenderung=115;Teamname="Ghana";break;
            case 3: aenderung=115;Teamname="Trinidad und Tobago";break;
            case 4: aenderung=115;Teamname="Togo";break;
            case 5: aenderung=115;Teamname="Angola";break;
            case 6: aenderung=100;Teamname="";break;
            case 7: aenderung=100;Teamname="";break;
            case 8: aenderung=90;Teamname="Brasilien";break;
            case 9: aenderung=90;Teamname="Niederlande";break;
        }
        //Team ändern
        plan.AendereTeam(Teamname,aenderung);
    }else{
        string Teamname=Input;
        cin>>Input;
        while(!atoi(Input.c_str())){//solange keine Zahl kommt
            Teamname.append(" ");
            Teamname.append(Input);//Zweiter Teil
            cin>>Input;
        }
        int aenderung=atoi(Input.c_str());
        //Team ändern
        plan.AendereTeam(Teamname,aenderung);
    }
    cout<<">>";
    cin>>Input;
}

//Spielplan laden
ifstream Spielplan;//Datei-Objekt
Spielplan.open("spielplan.txt",ios::binary|ios::in);
//Fehler beim öffnen??
if(!Spielplan){
    cerr<<"Rangliste konnte nicht geöffnet werden";
}
//Datei auslesen
Spielplan >> Input;
if(Input!="<spielplan")cerr<<"Fehler in Spielplan-Datei!";
cout<<endl<<"Spielplan 'spielplan.txt' auslesen..."<<endl;
//Einleseschleife
Spielplan >> Input;
//Statusvariablen
int gruppenphase=0;//Befinden wir uns beim einlesen von Gruppen? 0=Keine Gruppenphase
int kograd=0;//In welchen KO-Spielen befinden wir uns? 0=Kein KO
while(!Spielplan.eof()){
    if(Input=="<gruppen"){
        Spielplan>>Input;
        gruppenphase=atoi(Input.c_str());
    }else if(Input=="<gruppe"){
        //Gruppe erzeugen
        CGruppe gruppe;
        Spielplan>>Input;//Gruppenname einlesen
        gruppe.SetName(Input.c_str(),gruppenphase);
        //Teams der gruppe einlesen
        Spielplan>>Input;
        CLink link;
        while(Input!="<gruppen"&&Input!="<gruppe"&&
            Input!="<ko"&&Input!="<finale"&&!Spielplan.eof()){//Möglichkeit,wie
            //Deklaration der Teams in der gruppe
            //abgebrochen wird

            if(Input[0]=='<'){//Team aus tag
                if(Input=="<gp"||Input=="<se
                    Spielplan>>Input;//ID einlesen
                    string GrName=Input;
                    Spielplan>>Input;
                    int Platz=atoi(Input.c_str());
                    link.Init(plan.GetGruppe(GrName),Platz);
                    //Team zur Gruppe Hinzufügen
                    gruppe.AddLink(link);
                }
            }else{//team einfach Deklariert
                string teamname=Input;
                while(!plan.GetTeam(teamname)){

```

```

        Spielplan>>Input;
        teamname=teamname+" "+Input;
    }
    //Füge Team hinzu
    link.Init(plan.GetTeam(teamname));
    //Team zur Gruppe Hinzufügen
    gruppe.AddLink(link);
}
Spielplan>>Input;//Nächstes Team einlesen
}
plan.AddGruppe(gruppe);
continue;//Gruppe ist gerade im Lesebuffer...->Nicht löschen!
}else if(Input=="<ko"||Input=="<finale"){
    bool finalspiele=Input=="<finale";
    Spielplan>>Input;//KO-Grad einlesen
    kograd=atoi(Input.c_str());
    int anzahl_spiele=finalspiele?1:kograd;//Bei Finalspielen ist Anzahl der Spiele
                                           //nicht gleich dem Grad

    //KO-Spiele einlesen(als Gruppe)
    for(int i=0;i<anzahl_spiele;i++){
        //Spielbeginn
        Spielplan>>Input;
        if(Input=="<spiel")
        {
            CGruppe spiel;//temp. Speicherort für eigenschaften von Spiel
            //Spielnummer
            Spielplan>>Input;
            spiel.SetName(Input,kograd);
            //Beide Teams auslesen
            CLink verkn[2];
            for(int t=0;t<2;t++){
                //von Gruppe oder Spiel?
                Spielplan>>Input;
                if(Input=="<gp"||Input=="<se"){
                    Spielplan>>Input;//ID einlesen
                    string GrName=Input;
                    Spielplan>>Input;
                    int Platz=atoi(Input.c_str());
                    verkn[t].Init(plan.GetGruppe(GrName),Platz);
                }
                //Link hinzufügen
                spiel.AddLink(verkn[t]);
            }
            plan.AddGruppe(spiel);
        }else{//Spiel einlesen
            cout<<"SYNTAX ERROR!!! Achtung: Keine Entsprechung bei "<<Input<<endl;
        }
    }
    // FOR: kograd Spiele einlesen
}else{
    cout<<"SYNTAX ERROR!!! Achtung: Keine Entsprechung bei "<<Input<<endl;
}
    Spielplan>>Input;//nächsten Teil einlesen
} //Einleseschleife Spielplan
Spielplan.close();
//Anzeigen
cout<<"Spielplan anzeigen? (J/N) ";
cin>>Input;
if(Input=="J"||Input=="j"){
    plan.Anzeigen();
}
Input="";
while(Input!="E"&&Input!="e"){
    //Wahrscheinlichkeiten abfragen
    cout<<endl<<endl<<"Fuer welches Spiel willst du die
        Wahrscheinlichkeiten wissen?"<<endl
        <<"Gib einfach die Spielnummer ein!"<<endl<<endl
        <<"Alternativ kannst du auch 'S' eingeben,"<<endl
        <<"um die WM einmal zu simulieren."<<endl<<endl
        <<"Sonst 'E' zum beenden!"<<endl
        <<">>";
    cin>>Input;
    if(Input=="S"||Input=="s"){
        //Simulieren
        plan.Simulieren();
    }else if(Input!="E"&&Input!="e"){
        //Soll ein Platz abgefragt werden, oder nur welche Teams dort spielen?
        cout<<endl<<"Platz eingeben oder 0 fuer Auflistung der Teilnehmer"<<endl<<">>";
        int platz;
    }
}

```



```

        cin>>platz;
        if(platz<0||platz>9)platz=0;//0 steht für Auflistung der Teilnehmer
        //Wie oft soll die WM simuliert werden?
        cout<<endl<<"Wieviele Berechnungen?"<<endl<<">>";
        string anz;
        cin>>anz;
        cout<<"Bitte Warten..."<<endl;
        if(platz)plan.Simulieren(Input,platz,atoi(anz.c_str()));
        else plan.Simulieren(Input,atoi(anz.c_str()));
    }
} //while(Input!="E")
return EXIT_SUCCESS;
}

```

### Gruppe.h

```

//Klasse zum Verwalten einer Gruppe
class CGruppe{
public:
    //Gruppe initialisieren
    void SetName(string,int);//Name/SpielNr, Gruppenphase/KOPhase
    void AddLink(CLink link){Links.push_back(link);}//Info über Team
    //Gruppe abfragen
    int GetPhase(){return Phase;}
    string GetName(){return Name;}
    int AnzahlTeams(void){return Links.size();}
    CLink GetLink(int platz){return Links[platz-1];}
    //Gruppe spielen lassen
    void ErstelleSpielplan(void);//Erstellt die Spiele in der Gruppe
    void Simulieren(bool ausgabe=true);
    //Öffentliche Eigenschaften
    bool mitberechnen; //wird das Spiel zum Berechnen der gefr. Wahrsch. benötigt
private:
    //Methoden
    void isort(int,int);//Sortiert das CLinks-Feld von start-ende nach Teamdaten
    void Sortiere(int,int,int,bool=false);//Erstellt die Rangliste
    int Phase;//aktuelle Gruppenphase/KOPhase
    string Name;//Name der Gruppe/Nummer des Spiels
    vector<CLink>Links;// Infos über die Teams in der Gruppe,sortiert nach Platzierungen
    vector<CSpiel> Spiele;//Die Spiele die in der Gruppe ausgetragen werden
};

```

### Gruppe.cpp

```

//Gruppe spielen lassen
void CGruppe::Simulieren(bool ausgabe){
    for(int spielnr=0;spielnr<Spiele.size();spielnr++){
        Spiele[spielnr].Simulieren(ausgabe);
    }
    //Gruppe sortieren
    Sortiere(0,Links.size()-1,0,ausgabe);
}

void CGruppe::ErstelleSpielplan(void){
    Spiele.clear();
    for(int teamnr=0;teamnr<Links.size();teamnr++){//alle Spielpaarungen erstellen
        for(int gegner=teamnr+1;gegner<Links.size();gegner++){
            CSpiel spiel;
            spiel.SetLinks(Links[teamnr],Links[gegner],Links.size()==2?true:false);
            Spiele.push_back(spiel);
        }
    }
}

//Private Methoden
void CGruppe::Sortiere(int start, int ende, int sortkat, bool anzeigen){//Sortiert die Teams nacheinander
    //Zahlenfeld mit Daten füllen
    //Zuerst alle Daten löschen
    for(int i=start;i<=ende;i++)Links[i].GetTeam()->Daten=0;

    for(int tn=start;tn<=ende;tn++){//Für jedes Team die Daten sammeln
        for(int sn=0;sn<Spiele.size();sn++){//Jedes Spiel untersuchen
            int anzahl=0;// anzahl der relevanten Teams im Spiel
            //Wie werden die Daten geholt?
            switch(sortkat){
                case 0://0. die Anzahl Punkte aus allen Gruppenspielen;
                    Links[tn].GetTeam()->Daten+=Spiele[sn].Punkte(Links[tn]);

```

```

        break;
    case 1:
        //Waren 2 der punktgleichen Mannschaften beim Spiel dabei?
        for(int k=start;k<=ende;k++)if(Spiele[sn].Dabei(Links[k]))anzahl++;
        if(anzahl>=2)Links[tn].GetTeam()->Daten+=Spiele[sn].Punkte(Links[tn]);
        break;

    case 2:
        //Waren 2 der punktgleichen Mannschaften beim Spiel dabei?
        for(int k=start;k<=ende;k++)if(Spiele[sn].Dabei(Links[k]))anzahl++;
        if(anzahl>=2&&Spiele[sn].Dabei(Links[tn]))Links[tn].GetTeam()->Daten+=
            (Spiele[sn].Tore(Links[tn])-Spiele[sn].GGTore(Links[tn]));
        break;

    case 3:
        //Waren 2 der punktgleichen Mannschaften beim Spiel dabei?
        for(int k=start;k<=ende;k++)if(Spiele[sn].Dabei(Links[k]))anzahl++;
        if(anzahl>=2&&Spiele[sn].Dabei(Links[tn]))Links[tn].GetTeam()->Daten+=
            Spiele[sn].Tore(Links[tn]);
        break;

    case 4://4. die Tordifferenz aus allen Gruppenspielen;
        Links[tn].GetTeam()->Daten+=(Spiele[sn].Tore(Links[tn])-
            Spiele[sn].GGTore(Links[tn]));
        break;

    case 5://5. die Anzahl der in allen Gruppenspielen erzielten Tore.
        Links[tn].GetTeam()->Daten+=Spiele[sn].Tore(Links[tn]);
        break;

    default://6. Losverfahren
        if(tn>=start&&tn<=ende)Links[tn].GetTeam()->Daten=rand()*3/RAND_MAX;
        break;
    } //switch(sortkat)
} //jedes spiel
} //jedes Team

//Zahlenfeld+Teams sortieren
isort(start,ende);

//Sind mehrere Gleichrangig?==> neue Sortierkategorie
int is=start,ie=start;//Indizes der gleichbewerteten Teams
for(int tn=start;tn<=ende-1;tn++){
    if(Links[tn].GetTeam()->Daten!=Links[tn+1].GetTeam()->Daten){
        if(is!=ie)Sortiere(is,ie,sortkat+1,anzeigen);
        is=tn+1;
        ie=tn+1;
    }
    else{
        ie=tn+1;
    }
}
if(is!=ie)Sortiere(is,ie,sortkat+1,anzeigen);
}

```

**CLink.h**

```

//Klasse zum Verwalten von Verknüpfungen auf Spiele bzw. Gruppen
class CLink{
public:
    CLink(){ptrGruppe=NULL;ptrTeam=NULL;} //Link noch nicht gesetzt
    CLink(CGruppe* ptrDieGruppe,int platzierung); //erstellt Link auf einen Gruppenplatz
    CLink(STeam* ptrDasTeam); //erstellt Link auf Team von Weltrangliste
    void Init(CGruppe* ptrDieGruppe,int platzierung); //wie Konstr.
    void Init(STeam* ptrDasTeam);
    string Beschreibung(); //Gibt zurück, auf was der Link zeigt
    STeam* GetTeam(); //Gibt Zeiger auf das Team zurück, auf das der Link aktuell zeigt.
    CGruppe* GetGruppe(){return ptrGruppe;} //Gibt Zeiger auf die Gruppe zurück, auf die
        //der Link aktuell zeigt.

    int GetPlatz(void){return Platz;}

private:
    //Eine Verknüpfung kann entweder auf eine Gruppenplatzierung verweisen
    CGruppe* ptrGruppe; //Zeiger auf die Gruppe, auf die der Link verweist
    int Platz; //Die Platzierung der gesuchten Mannschaft in der Gruppe
    //Oder auf einen Namen in der Weltrangliste
    STeam* ptrTeam;
};
#endif //link_h

```

**Team.h**

```
//Struktur zum Verwalten vom einzelnen Mannschaften
struct STeam{
    string Name;
    int Staerke;
    int Daten;//Zum berechnen von Tabellenpositionen, Punkten,...
};
```

**Spiel.h**

```
//Klasse zum Verwalten von Spielen
class CSpiel{
public:
    //Initialisieren des Spiels
    void SetLinks(CLink,CLink,bool);//Die Verknüpfungen zu den spielenden Teams
    void Simulieren(bool ausgabe=true);//simuliert das
    //Abfragen zum berechnen der Tabellenergebnisse
    bool Dabei(CLink);//War eine Mannschaft bei diesem Spiel dabei?
    int Punkte(CLink);//gibt zurück, wieviel Punkte das Team STeam aus gewonnen hat...
    int Tore(CLink);//dasselbe in grün
    int GGTore(CLink);//Gegentore
private:
    inline float GetP(int);//Berechnet die
    CLink Link1;
    CLink Link2;
    int tore1;
    int tore2;
    bool ko;//Wird bei Unentschieden aufgehört oder Verl./Elferschießen??
};
```

**Spiel.cpp**

```
inline float CSpiel::GetP(int diff){
    return (1.1015*exp(.0037*diff)/5.); //Formel für Torwahrscheinlichkeit...
}

void CSpiel::Simulieren(bool ausgabe){
    float p1,p2;//Wahrscheinlichkeiten für ein Tor
    p1=GetP(Link1.GetTeam()->Staerke-Link2.GetTeam()->Staerke);
    p2=GetP(Link2.GetTeam()->Staerke-Link1.GetTeam()->Staerke);
    tore1=tore2=0;
    if(ausgabe)cout<<Link1.GetTeam()->Name<<" - "<<Link2.GetTeam()->Name<<" ";
    //5 Angriffe von beiden Teams simulieren:
    for(int angriff=0;angriff<5;angriff++){
        if(rand()<p1*RAND_MAX)tore1++;
        if(rand()<p2*RAND_MAX)tore2++;
    }
    if(ausgabe)cout<<tore1<<": "<<tore2;
    //Bei KO-Spielen:Verlängerung... weitere 2 Angriffe pro Team
    if(ko&&tore1==tore2){
        for(int angriff=0;angriff<2;angriff++){
            if(rand()<p1*RAND_MAX)tore1++;
            if(rand()<p2*RAND_MAX)tore2++;
        }
        if(ausgabe)cout<<" -> "<<tore1<<": "<<tore2<<" n.V.";
    }
    bool elfer=false;
    //Bei KO-Spielen:Elferschießen... 5 Elfer pro Team(p=0,8)
    if(ko&&tore1==tore2){
        elfer=true;
        for(int elfer=0;elfer<5;elfer++){
            if(rand()<.8*RAND_MAX)tore1++;
            if(rand()<.8*RAND_MAX)tore2++;
        }
    }
    //Bei KO-Spielen:Elferschießen... je ein elfer bis eine mannschaft besser ist
    while(ko&&tore1==tore2){
        if(rand()<.8*RAND_MAX)tore1++;
        if(rand()<.8*RAND_MAX)tore2++;
    }
    if(ausgabe&&elfer)cout<<" -> "<<tore1<<": "<<tore2<<" n.E.";
    if(ausgabe)cout<<endl;
}
```

**Spielplan.h**

```
//Klasse zum Verwalten des Spielplans
class CSpielplan{
public:
    CSpielplan();
    //Spielplan initialisieren
    void AddTeam(STeam team){WRListe.push_back(team);}//Der WR-Liste ein Team hinzufügen
    void AendereTeam(const string&,int);//Ändert die Stärke des eines Teams
    void AddGruppe(CGruppe gruppe){Gruppen.push_back(gruppe);}
    //Spielplan abfragen
    STeam* GetTeam(const string&);//Liefert Zeiger auf Team der WR mit demselben
    CGruppe* GetGruppe(const string&);//Liefert Zeiger auf die Gruppe KO-Spiel
    void WRListeAnzeigen(void);
    void Anzeigen(void);
    //WM simulieren
    void Simulieren(void);//Einmalige Simulation der WM
    void Simulieren(string&,unsigned int anzahl);//anzahl-malige Simulation aller not-
        //wendigen Spiele, um die Wahrscheinlichkeiten des angegebenen Spiels zu berechnen
    vector<STeam> Simulieren(string&,int,unsigned int anzahl, bool anzeigen=true);
        //anzahl-malige Simulation aller notwendigen
//Spiele, um die Wahrscheinlichkeiten der angegebenen Platzierung zu berechnen; Gibt Liste
//mit allen Teams+Wahrsch. zurück
private:
    void markiereVorgaenger(CGruppe*);//Sucht die Zur berechnung eines Spiels notwendigen
    vorherigen Spiele
    string GetWortVonGrad(int);

    vector<STeam> WRListe;//Feld zum Speichern aller Teams(=> Die Weltrangliste)
    vector<CGruppe> Gruppen;
};
```

**Spielplan.cpp**

```
//Implementation der Klasse zum Verwalten des Spielplans
void CSpielplan::Simulieren(void){
    cout <<endl<<"Einmalige Simulation der WM"<<endl<<"~~~~~"<<endl;
    //Gruppen Simulieren
    int phase=0;
    for(int index=0;index<Gruppen.size();index++){
        //KO-Spiel oder Gruppe?
        if(Gruppen[index].AnzahlTeams(>2){//Gruppe
            if(Gruppen[index].GetPhase()!=phase){
                phase=Gruppen[index].GetPhase();
                cout <<"Gruppenphase " <<phase<<":"<<endl;
            }
            //Gruppenspiele erstellen
            cout<<endl<<"Gruppe " <<Gruppen[index].GetName()<<endl;
            cout<<"~~~~~"<<endl;
            Gruppen[index].ErstelleSpielplan();
            Gruppen[index].Simulieren();
            //Rangliste:
            cout<<endl<<"Rangliste:"<<endl;
            for(int platz=1;platz<=Gruppen[index].AnzahlTeams();platz++){
                cout<<" " <<platz<<". " <<Gruppen[index].GetLink(platz).GetTeam()->Name;
            }
        }else{//KO-Spiel
            if(Gruppen[index].GetPhase()!=phase){
                phase=Gruppen[index].GetPhase();
                cout<<endl<<GetWortVonGrad(phase)<<":"<<endl;
                for(int z=0;z<=GetWortVonGrad(phase).length();z++)cout<<"~";
                cout<<endl;
            }
            //Spiel Simulieren, Anzeigen übernimmt schon die Klasse Gruppe
            Gruppen[index].ErstelleSpielplan();
            Gruppen[index].Simulieren();
        }
    }
}

void CSpielplan::markiereVorgaenger(CGruppe* vonGruppe){//Die für die Gruppe *vonGruppe
    //wichtigen Spiele markieren
    vonGruppe->mitberechnen=true;//Zum Mitberechnen markieren
    //Vorgaengerspiele finden
    for(int tnr=0;tnr<vonGruppe->AnzahlTeams();tnr
        CGruppe* naechsteGruppe=vonGruppe->GetLink(tnr+1).GetGruppe();
        if(naechsteGruppe){
```

```

        markiereVorgaenger(naechsteGruppe);
    }
}

void CSpielplan::Simulieren(string& id,unsigned int anzahl){
    vector<STeam> Teamliste[20]; //Feld aus Teamlisten
    CGruppe* untersuchteGruppe=GetGruppe(id);
    cout<<"Wahrscheinlichste Zusammensetzung:"<<endl
    <<"~~~~~"<<endl;
    for(int index=1;index<=untersuchteGruppe->AnzahlTeams();index++){
        if(untersuchteGruppe->GetLink(index).GetGruppe()){
            string grName=untersuchteGruppe->GetLink(index).GetGruppe()->GetName();
            int platz=untersuchteGruppe->GetLink(index).GetPlatz();
            Teamliste[index-1]=Simulieren(grName,platz,anzahl,false);
            cout<<untersuchteGruppe->GetLink(index).Beschreibung()<<": "
            <<Teamliste[index-1].front().Name<<" mit "
            <<Teamliste[index-1].front().Daten/(float)anzahl*100.<<" Prozent"<<endl;
        }
    }
    cout<<"Wahrscheinlichkeiten fuer alle Teams:"<<endl
    <<"~~~~~"<<endl;
    //alle Daten in Liste[0] zusammenfassen
    for(int liste=1;liste<untersuchteGruppe->AnzahlTeams();liste++){
        for(int team=0;team<Teamliste[listete].size();team++){
            //Teamliste[team] zu Teamliste[0] hinzufügen
            int gefunden=Teamliste[0].size();
            for(int t0=0;t0<Teamliste[0].size();t0++){
                if(Teamliste[0][t0].Name==Teamliste[listete][team].Name){
                    gefunden=t0;
                }
            }
            if(gefunden==Teamliste[0].size())Teamliste[0].push_back(Teamliste[listete][team]);
            else Teamliste[0][gefunden].Daten+=Teamliste[listete][team].Daten;
        }
    }
    //Liste[0] sortieren
    vector<STeam> Teams_t; //temporär für Teams
    //Temp. Feld füllen
    for(int index=0;index<Teamliste[0].size();index++){
        Teams_t.push_back(Teamliste[0][index]);
    }
    //Sortieren durch auswählen
    for(int i=0;i<Teamliste[0].size();i++){ //Jeden Platz des urspr. Feldes
        int jmax=0;
        for(int j=0;j<Teams_t.size();j++){if(Teams_t[j].Daten > Teams_t[jmax].Daten) jmax=j;
        Teamliste[0][i]=Teams_t[jmax];
        Teams_t[jmax]=Teams_t.back();Teams_t.pop_back();
    }
    //Ausgeben:
    for(int i=0;i<Teamliste[0].size();i++)
        cout<<Teamliste[0][i].Name<<" mit "
        <<Teamliste[0][i].Daten*100./(anzahl*untersuchteGruppe->AnzahlTeams())
        <<" Prozent"<<endl;
}

vector<STeam> CSpielplan::Simulieren(string& id, int platz, unsigned int anzahl, bool
anzeigen){
    //WK für einen Platz ausrechnen und ausgeben
    //zuerst alle markierungen löschen
    for(int i=0;i<Gruppen.size();i++)Gruppen[i].mitberechnen=false;
    //Spiele bzw. Gruppen rekursiv markieren, die Simuliert werden sollen
    CGruppe* untersuchteGruppe=GetGruppe(id);
    markiereVorgaenger(untersuchteGruppe);

    //WM Simulieren
    vector<STeam> Teams; //Alle Teams die in diesem Spiel vorkommen;Daten:Anzahl der Vorkommen

    //Gruppenspielpläne nur einmal erstellen-->Zeitersparnis
    for(int index=0;index<Gruppen.size();index++){
        if(Gruppen[index].mitberechnen)Gruppen[index].ErstelleSpielplan();
    }

    for(int sim=0;sim<anzahl;sim++){
        //Gruppen Simulieren
        for(int index=0;index<Gruppen.size();index++){
            if(Gruppen[index].mitberechnen)Gruppen[index].Simulieren(false);

```

```
}

//Untersuchen, welches Teams im Spiel mit der ges. id den ges. Platz belegt hat
STeam team_t;
team_t.Name=untersuchteGruppe->GetLink(platz).GetTeam()->Name;
team_t.Daten=1;
//Ist das Team schon im Feld?
int tn=0;
for(tn=0;tn<Teams.size();tn++){
    if(Teams[tn].Name==team_t.Name){
        Teams[tn].Daten++;
        break;
    }
}
if(tn==Teams.size())Teams.push_back(team_t);
} //anzahl simulationen durchführen

//Ergebnisse formatieren
//Zu kleine Ergebnisse zu Sonstige zusammenfassen
STeam sonstige;
sonstige.Name="Sonstige";
sonstige.Daten=0;
for(int index=0;index<Teams.size();index++){
    //0.5-Prozent-Hürde
    if(100.*Teams[index].Daten/(float)anzahl<.5){
        sonstige.Daten+=Teams[index].Daten;
        //Team löschen
        Teams[index]=Teams.back();Teams.pop_back();
        index=0;//Schleife neu starten
    }
}
if(sonstige.Daten)Teams.push_back(sonstige);

//Ergebnisse sortieren
vector<STeam> Teams_t;//temporär für Teams
//Temp. Feld füllen
for(int index=0;index<Teams.size();index++){
    Teams_t.push_back(Teams[index]);
}
//Sortieren durch auswählen
for(int i=0;i<Teams.size();i++){//Jeden Platz des urspr. Feldes
    int jmax=0;
    for(int j=0;j<Teams_t.size();j++){//Größtes Element finden
        if(Teams_t[j].Daten > Teams_t[jmax].Daten) jmax=j;
    }
    //Größtes Element einfügen
    Teams[i]=Teams_t[jmax];
    //eingefügtes Element aus Temp.Liste löschen
    Teams_t[jmax]=Teams_t.back();
    Teams_t.pop_back();
}

//Ergebnisse ausgeben

if(anzeigen)for(int index=0;index<Teams.size();index++){
    cout<<"    "<<Teams[index].Name<<": "
        <<100.*Teams[index].Daten/(float)anzahl<<" Prozent"<<endl;
}
return Teams;
}
```

Hiermit erkläre ich, die Einsendung zum 24. Bundeswettbewerb Informatik ohne fremde Hilfe angefertigt zu haben.

Lindenberg im Allgäu, den 19.4.2006, .....