IT-Projekt: Pong

von Lukas Schäf

Bei dem, von mir erstellten Projekt, handelt es sich um das altbekannte Spiel Pong. Wie mit ihnen abgesprochen, werde ich im Folgenden, wegen dem zu umfangreichem Code, nur die wichtigsten Programmteile erläutern. Eine kurze Beschreibung zur Funktion des Programms, ist im Code in Kommentarform zu jedem Schritt vorhanden.

Anmerkung:
Da das Programm in Visual Studio 2010 verfasst wurde und Objekte der Klasse Microsoft.VisualStudio.Powerpacks verwendet, ist es nicht kompitabel mit der Visual Studio Version von 2008!

Gliederung:

1. Beschreibung des Aufbaus:

Die Programmoberfläche besteht aus fünf Formen: MainForm, Anleitung, Highscores, Welcome und Credits. Davon sind die ersten drei Formen selbst erstellt. Welcome und Credits sind von Visual Studio angebotene Standartformen, die von mir leicht abgeändert wurden.
Die MainForm stellt das Spielfeld, mit den zwei Bumpern und dem Spielball dar. Im oberen Teil der Form findet sich eine Menüleiste, durch die man auf Einstellungen, spielbezogene Aktionen und die anderen Formen zugreifen kann.
Über die näheren Funktionen der Menüleiste, kann man unter Hilfe -> Anleitung mehr erfahren.

Zum Aufbau des Codes ist anzumerken, dass ich vier verschiedene Timer verwende, die zum Teil die selben Taktzeiten haben. Ich habe sie jedoch zwecks der Übersichtlichkeit aufgeteilt. Drei Timer sind den bewegten Objekten zugeordnet und ein Timer ist für die Zeitzählung und den Geschwindigkeitsanstieg zuständig.

2. Erklärung der wichtigsten Programmteile:

- Steuerung des Spielerbumper:

Die Steuerung des Spielerbumpers befindet sich in der Sub "playerTimer_Tick(...)". welche, wenn "playerTimer" gestartet ist, alle 1 ms aufgerufen wird.

```
playerBumper.Top = MousePosition.Y - 120
```

Mit diesem Befehl verknüpfe ich die Y-Koordinaten des Punktes der linken oberen Ecke des Spielerbumpers mit der Y-Koordinate der Mausposition minus 120 Pixel. Da diese Sub alle 1 ms aufgerufen wird, kann man den Bumper mit der Maus bewegen. Die 120 Pixel ziehe ich ab, dass die Maus annähernd auf einem Level mit der Mitte des Spielerbumpers auf dem Bildschirm ist.

```
If playerBumper.Top <= MenuStrip.Height Then
     playerBumper.Top = MenuStrip.Height
ElseIf playerBumper.Bottom >= höheSpielfeld Then
     playerBumper.Top = höheSpielfeld - playerBumper.Height
End If
```

Mit dieser If-Anweisung lege ich fest, dass der Spielerbumper sich nicht außerhalb des Spielfeldes bewegen kann, egal wie groß bzw. klein die Y-Koordinate der Maus ist.
Die If-Bedinung beschreibt den Fall, dass der Bumper sich nach oben weg bewegen würde, die ElseIf-Bedingung den Fall, dass er sich nach unten weg bewegt.

- AI des Computers:

Die Steuerung des Computerbumpers (AI) befindet sich in der Sub "computerTimer_Tick(...)", welche, wenn "computerTimer" gestartet ist, alle 1 ms aufgerufen wird.

```
If gameBall.Top <= computerBumper.Top + _
                                        (computerBumper.Height/2) Then
      If Sekunden < 30 Then
            computerBumper.Top -= vyBall
      Else
            computerBumper.Top -= 20
End If
```

Diese If-Anweisung beschreibt den Fall, dass die Y-Koordinate des Spielballs über der Mitte des Computerbumpers liegt. Mitte deswegen, da zu der Y-Koordinate der oberen Grenze des Bumpers, die Höhe des Bumpers geteilt durch zwei hinzugezählt wird. Man geht hier also von der Y-Koordinate eines Punktes in der Mitte des Bumpers aus!

Nun wird in 2 Fällen unterschieden:
1. Bevor 30 Sekunden vergangen sind, soll sich der Computerbumper mit der gleichen Geschwindigkeit wie der Ball nach oben bewegen.
2. Nachdem 30 Sekunden vergangen sind, soll sich der Bumper mit einer konstanten Geschwindigkeit von 20 nach oben bewegen.

```
ElseIf gameBall.Top >= computerBumper.Top + _
                                    (computerBumper.Height / 2) Then
      If Sekunden < 30 Then
            computerBumper.Top += vyBall
      Else
            computerBumper.Top += 20
      End If
End If
```

In dieser If-Anweisung, die im Code direkt unter der obigen Anweisung steht, wird nun der Fall behandelt, dass die Position des Spielballs unterhalb der Y-Koordinate eines Punktes in der Mitte des Computerbumpers liegt.

Es wird wie im obigen in 2 Fällen unterschieden:
1. Bevor 30 Sekunden vergangen sind, soll sich der Computerbumper mit der gleichen Geschwindigkeit wie der Ball nach unten bewegen.
2. Nachdem 30 Sekunden vergangen sind, soll sich der Bumper mit einer konstanten Geschwindigkeit von 20 nach unten bewegen.

Wie nun unschwer zu erkennen ist, erwischt unser Computergegner bis zur 30. Sekunde jeden Ball, egal wie schnell er ist. Doch nach dieser Zeitspanne, besitzt er eine nicht vom Ball abhängige Geschwindigkeit, mit der Folge, dass er wohl nach kurzer Zeit zu langsam ist um einen Ball zu erreichen

- Reflexion des Balles:

Die Steuerung und die Reflexion des Balles befindet sich in der Sub "BallTimer_Tick(...)", welche, wenn der Timer "BallTimer" gestartet ist, alle 1ms ausgeführt wird.

Anmerkung zur Steuerung des Balls:

```
gameBall.Top -= vyBall
gameBall.Left -= vxBall
```

vxBall und vyBall sind, zu jedem Neustart des Spiels, zufällig zwischen 5 und 10 gewählte Ganzzahlen.
Hier ziehe ich der X- und Y-Koordinate des Balls jeweils die vxBall- und vyBall-Geschwindigkeit ab. (vxBall und vyBall sind zum Start immer positiv)
-> Der Ball bewegt sich zum Start des Spiels pro 1 ms die vxBall-Geschwindikeit nach links und die vyBall-Geschwindigkeit nach oben.

- Reflexion an den oberen bzw. unteren Spielfeldgrenzen:

```
If gameBall.Top <= MenuStrip.Height _
                          Or gameBall.Bottom >= höheSpielfeld Then
     vyBall = -vyBall
End If
```

Mit dieser If-Anweisung wird geprüft, ob die Y-Koordinate des Punkts auf der Oberseite des Balls kleiner oder gleich der Höhe der Menüleiste, also der oberen Spielfeldgrenze, ODER ob die Y-Koordinate des Punkts auf der Unterseite des Balls größer oder gleich der Höhe des Spielfelds ist.
Wenn einer dieser beiden Fälle eintritt, wird die Y-Geschwindigkeit (vyBall) invertiert.

An einem Beispiel erläutert: Wenn der Ball vorher nach oben geflogen ist, also eine negative Y-Geschwindkeit besessen hat, und an die obere Spielfeldgrenze stößt, wird seine Y-Geschwindigkeit invertiert. Er besitzt nun eine positive vyBall und fliegt damit nach unten.

- Reflexion am Spieler- und Computerbumper

```
If gameBall.Bounds.IntersectsWith(playerBumper.Bounds) _
     Or gameBall.Bounds.IntersectsWith(computerBumper.Bounds) Then
     vxBall = -vxBall
End If
```

Allgemein:

Die Methode "Objekt1.Bounds.IntersectsWith(Objekt2.Bounds)" prüft ob das Objekt1 eine Schnittmenge mit Objekt2 hat. Ist dies der Fall ist die Aussage TRUE, ansonsten FALSE.

Anhand unseres Codes kann man also sagen:

Wenn der Spielball mit dem Playerbumper, oder dem Computerbumper eine Schnittmenge bildet, also sie sich berühren, soll die X-Geschwindigkeit (vxBall) invertiert werden.

An einem Beispiel erläutert:
Wenn der Ball von rechts nach links fliegt (vxBall negativ) und man stellt seinen Bumper so, dass der Ball auf ihn trifft, wird die X-Geschwindigkeit des Balles invertiert, also fliegt er von links nach rechts (vxBall positiv) weg.

- Geschwindigkeitsregelung des Balles:

Die Geschwindigkeitserhöhung des Balles befindet sich in der Sub "SpielzeitTimer_Tick(..)", welche, wenn der SpielzeitTimer gestartet ist, jede Sekunde aufgerufen wird.

```
For i As Integer = 0 To 50 Step 10
(1.)  If Sekunden = i Then
      (2.)  If vxBall < 0 Then
                  vxBall -= 5
            Else
                  vxBall += 5
            End If
      (3.)  If vyBall < 0 Then
                  vyBall -= 5
            Else
                  vyBall += 5
            End If
      End If
Next
```

Die For-Schleife bewirkt, dass die Hilfsvariable 'i' die Werte 0,10,20,30,40 und 50 annehmen kann. 'i' = 0 hat im ersten Durchlauf keine Bedeutung, da "Sekunden" oben im Code um 1 erhöht wurde. Die 0 spielt eine Rolle beim Minutenumsprung, wenn "Minuten" = 1 und "Sekunden" = 0 ist.
Der Wert 'i' muss ebenfalls nicht die 60 abdecken, da durch den Minutenumsprung "Sekunden" von 60 auf 0 und dafür "Minuten" von 0 auf 1 gesetzt wird.

Die 1. If-Anweisung vergleicht nun den Wert von 'i' mit dem Wert von "Sekunden". Das heißt die 1. If-Bedingung (Sekunden = i) wird TRUE bei den Sekunden 0,10,20,30,40 und 50, also alle 10 Sekunden ab dem Start wird die 1. If-Anweisung durchlaufen.

Die zweite bzw. dritte If-Anweisung prüft zuerst ob die X- (vxBall) bzw. Y- (vyBall) Geschwindigkeit negativ, also kleiner als Null, sind. Ist dies der Fall wird von der X- bzw. Y-Geschwindigkeit 5 abgezogen. Ist dies jedoch nicht der Fall, also ist die X- bzw. Y-Geschwindigkeit positiv, wird ihnen 5 hinzugezählt.

Das obige nochmal im Klartext zusammengefasst heißt das:

Ab der 10. Sekunde wird alle 10 Sekunden der Betrag der X- und Y-Geschwindkeit um 5 erhöht.

- Zusammenfassung:

- Man kann mit seinem Bumper, den man mit der Maus steuert, den Ball in die andere Richtung schlagen. Der Computer wird das selbe versuchen.

- Der Ball wird von der oberen und unteren Spielfeldgrenze, sowie von den beiden Bumpern abprallen.

- Die Geschwindigkeit des Balles erhöht sich alle 10 Sekunden.

- Der Computer ist bis zu einer bestimmten Zeit genauso schnell wie der Ball, was das Gewinnen etwas schwieriger, aber nicht unmöglich macht :).