Beispielprojekt: Pong

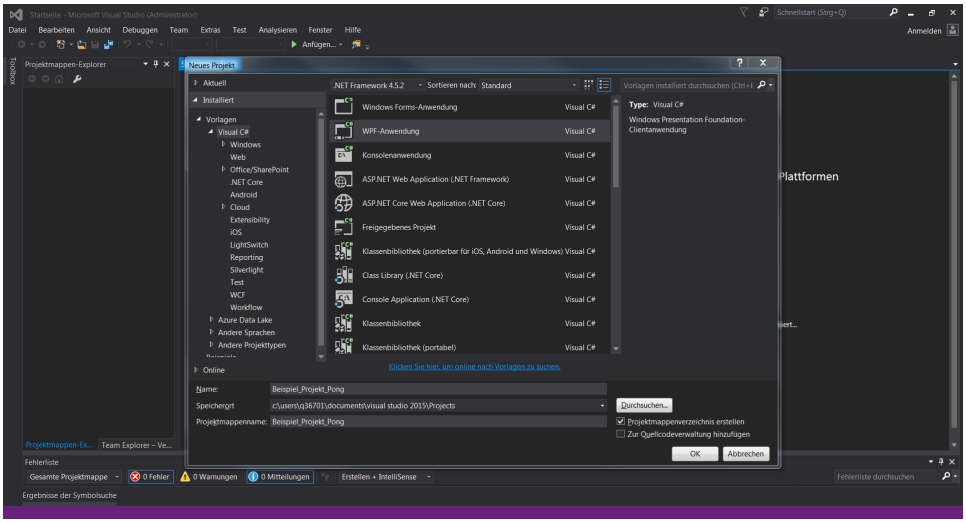
Visual-Studio-WPF-Projekt für das Retrogame "Pong"

Inhaltsverzeichnis / list of contents

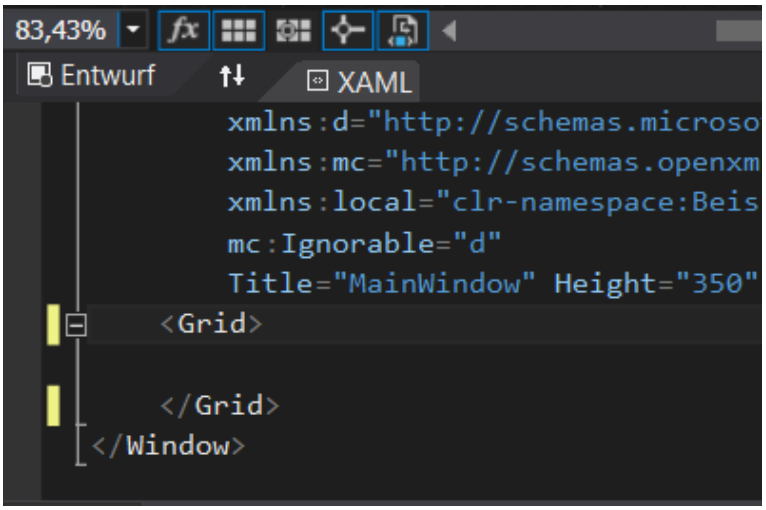
SVN: https://zznt14v.intern.zollner.de/svn/ima_praktikum

Schritt-für-Schritt-Anleitung

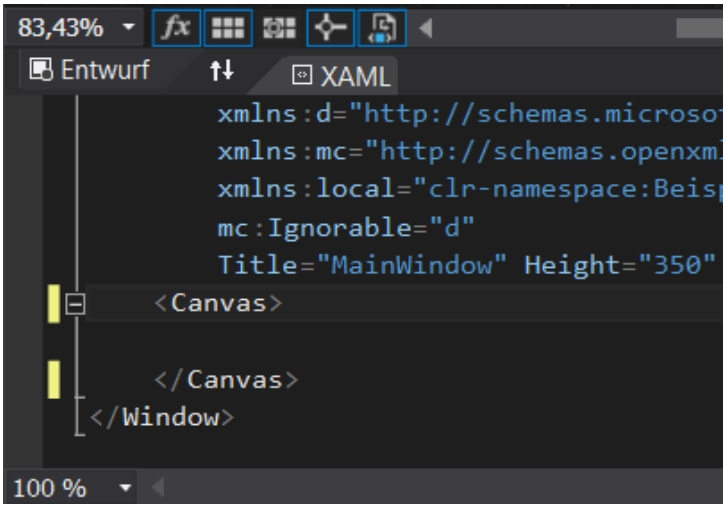
- Neues WPF-Anwendung in Visual Studio erstellen



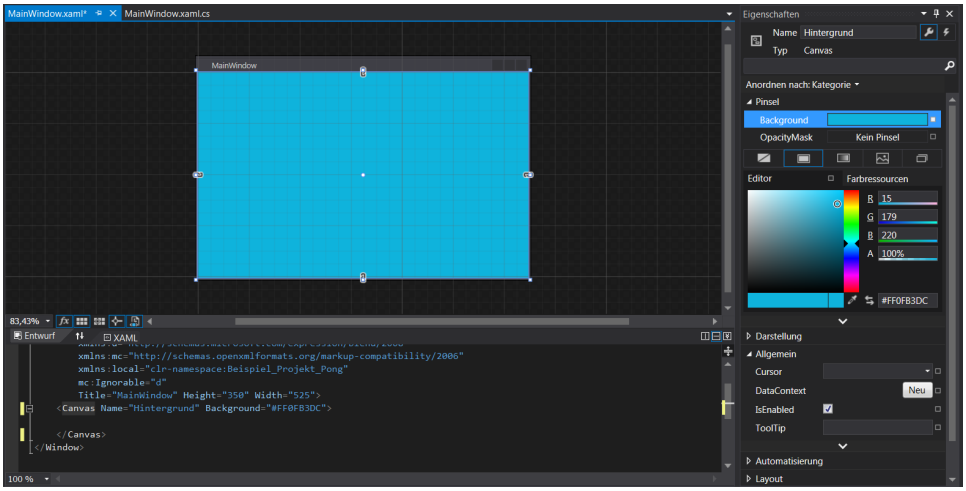
- Im XAML-Editor "Grid" in "Canvas" ändern



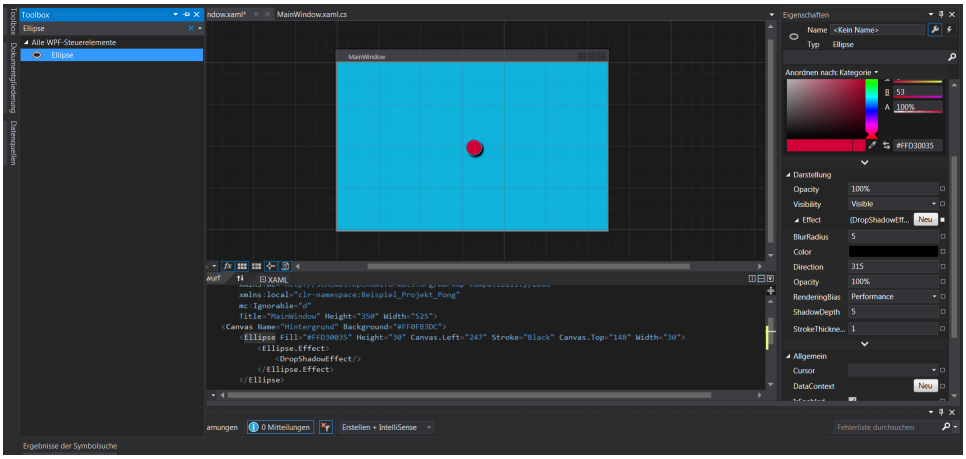
blocked URL



- Hintergrundfarbe und **Namen** für das Canvas festlegen

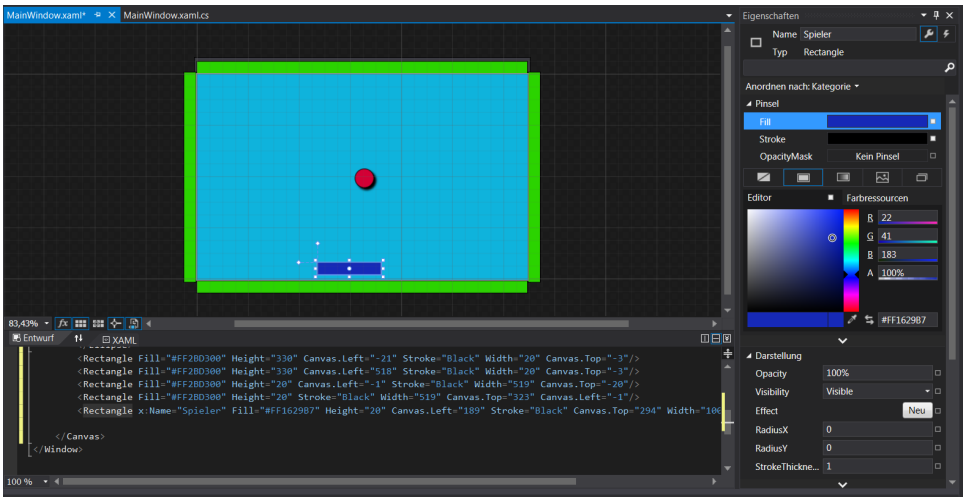


- Ellipse aus der Toolbox hinzufügen, Farbe, **Name** und eventuell Größe anpassen. In den Eigenschaften unter "*Darstellung*" " *Effect*" "*Neu*" "*DropShadowEffect*" hinzufügen.

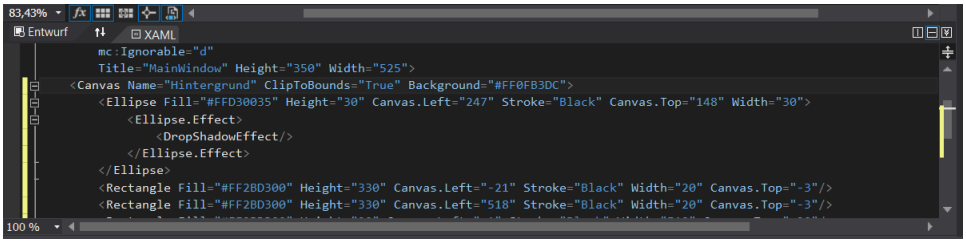


- Rectangles aus der Toolbox für die Begrenzungen und den Spieler hinzufügen. **Name**, **Farbe** und eventuell Größe für den Spieler Balken und für die **Ränder** festlegen.

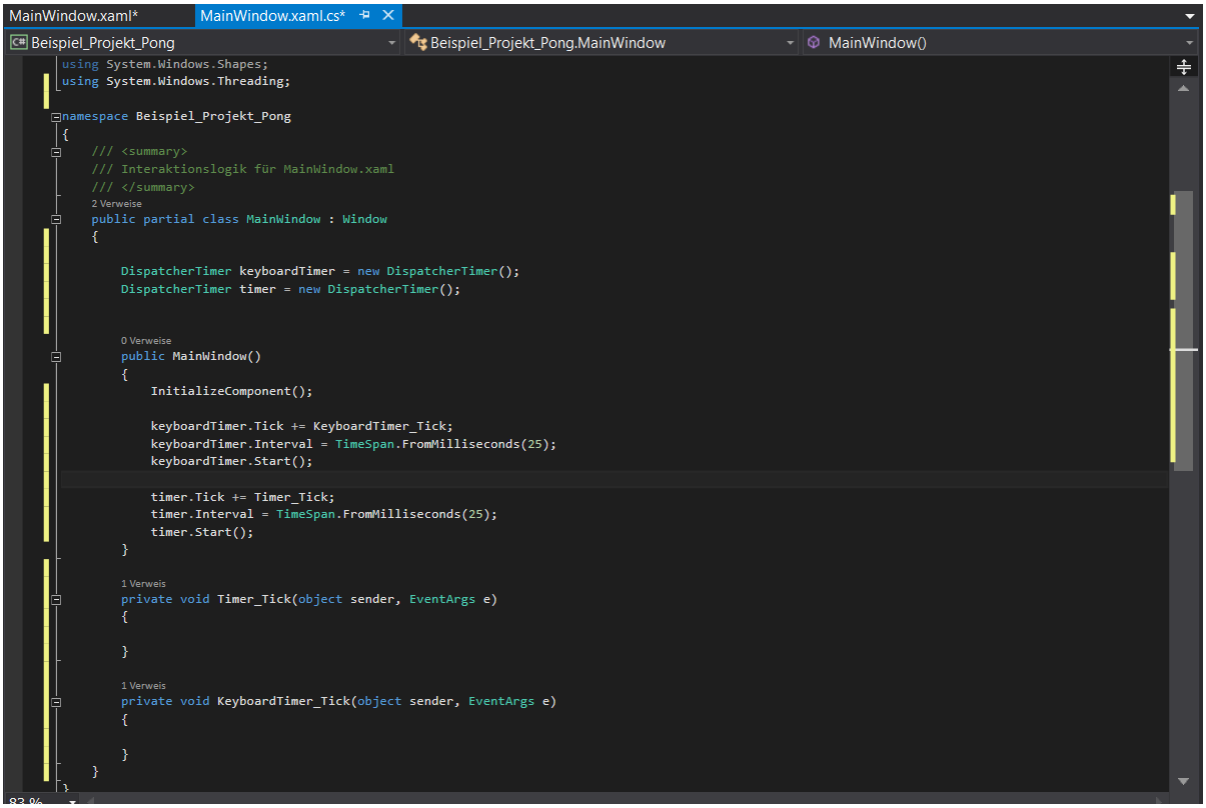
- Es ist zu beachten, dass für jedes Element, welches später mit einem anderen kollidieren kann (**Ball, Spieler-Balken, Ränder**), die Eigenschaften ***"Canvas.Top"*** und ***"Canvas.Left"*** gesetzt sind.



- "***ClipToBounds***" für das Hintergrund-Canvas auf "***True***" setzen. Dadurch werden die Balken für die Abgrenzungen nicht mehr sichtbar.



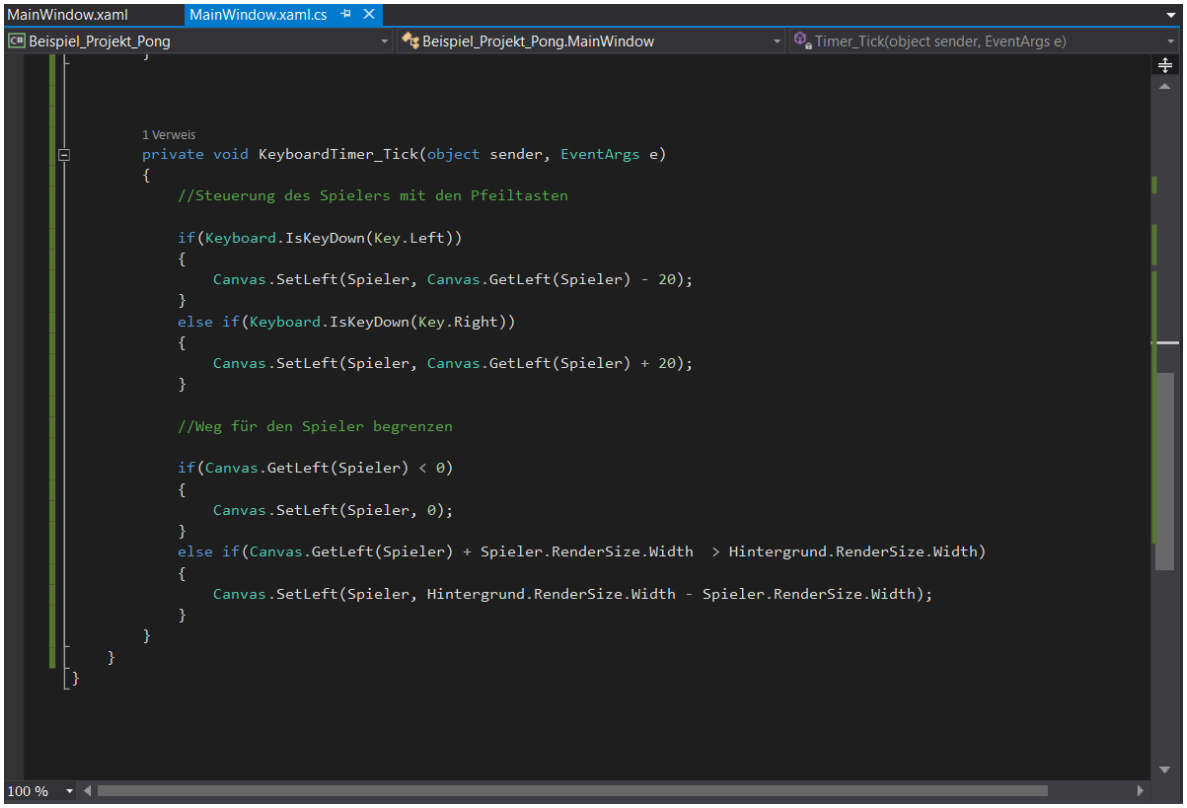
- Mit der Taste "F7" zum CodeBehind springen. Dort zwei Objekte der Klasse *DispatcherTimer* im Konstruktor instanziieren. Der Namespace "**System.Windows.Threading**" wird dazu benötigt.
Für die beiden Timer ein Tick-Event anlegen und ein Intervall (~25) festlegen.



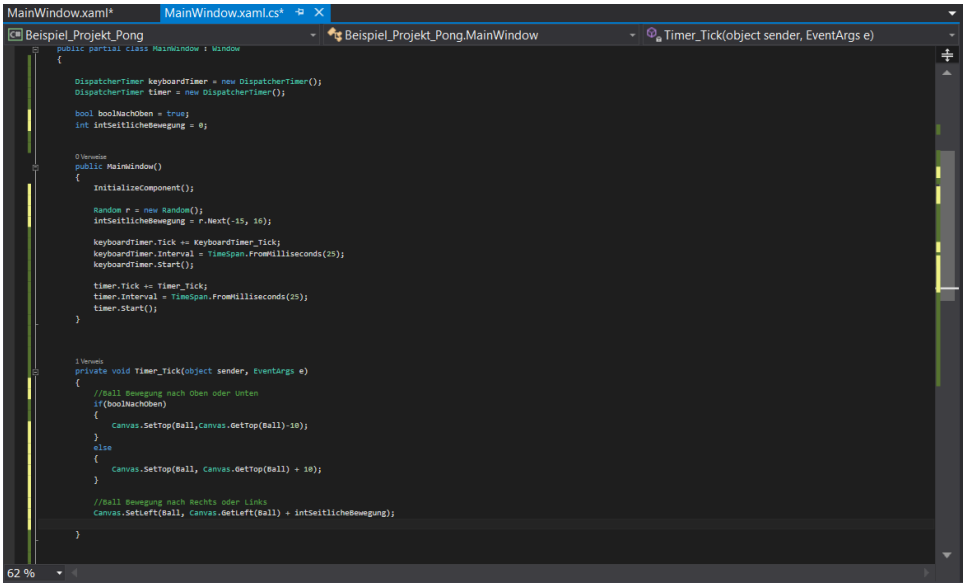
- Steuerung des Spielers mit den Pfeiltasten im *keyboardTimer Tick-Event*. Ebenfalls eine Begrenzung für den Weg des Spielers.

Ein Timer ist in diesem Fall besser als das KeyDown-Event, da schneller gesteuert werden kann.

```
        //Steuerung des Spielers mit den Pfeiltasten

        if(Keyboard.IsKeyDown(Key.Left))
        {
            Canvas.SetLeft(Spieler, Canvas.GetLeft(Spieler) - 20);
        }
        else if(Keyboard.IsKeyDown(Key.Right))
        {
            Canvas.SetLeft(Spieler, Canvas.GetLeft(Spieler) + 20);
        }

        //Weg für den Spieler begrenzen

        if(Canvas.GetLeft(Spieler) < 0)
        {
            Canvas.SetLeft(Spieler, 0);
        }
        else if(Canvas.GetLeft(Spieler) + Spieler.RenderSize.Width  > Hintergrund.RenderSize.Width)
        {
            Canvas.SetLeft(Spieler, Hintergrund.RenderSize.Width - Spieler.RenderSize.Width);
        }
```



- Steuerung des Balls durch das *Timer_Tick-Event*. Zuerst eine *Bool-Variable* für Oben/Unten und eine *Int-Variable* für Links /Rechts deklarieren.
  Im Konstruktor ein neues Objekt der Klasse "**Random**" erzeugen und eine zufällige Zahl für die Int-Variable vergeben.



- Neue Funktion für die Kollisions-Abfrage zwischen Ball und den Rändern erstellen.

```
 private bool Kollision(UIElement element1, UIElement element2)
        {
            Rect rect1 = new Rect(Canvas.GetLeft(element1), Canvas.GetTop(element1), element1.
RenderSize.Width, element1.RenderSize.Height);
            Rect rect2 = new Rect(Canvas.GetLeft(element2), Canvas.GetTop(element2), element2.
```

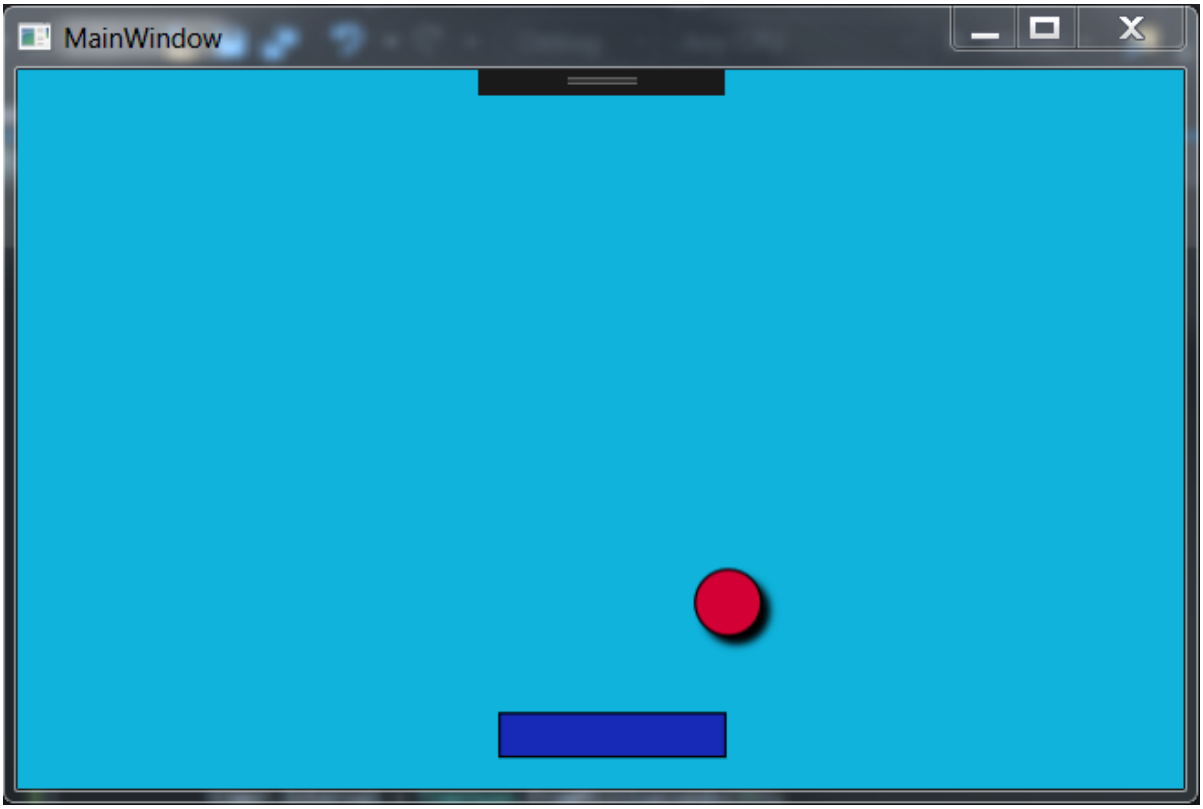
```
RenderSize.Width, element2.RenderSize.Height);

            return rect1.IntersectsWith(rect2);
        }
```

- Kollisionsabfrage im *Timer_Tick-Event.* Es wird überprüft ob der Ball einen Rand berührt. Wenn der Ball eine Begrenzung berührt, wird die Richtung des Balls umgedreht.
  Wird der Ball nicht mit dem Spieler abgefangen, ist das Spiel verloren und eine Meldung wird ausgegeben.

```
private void Timer_Tick(object sender, EventArgs e)
        {
            //Ball Bewegung nach Oben oder Unten
            if(boolNachOben)
            {
                Canvas.SetTop(Ball,Canvas.GetTop(Ball)-10);
            }
            else
            {
                Canvas.SetTop(Ball, Canvas.GetTop(Ball) + 10);
            }

            //Ball Bewegung nach Rechts oder Links
            Canvas.SetLeft(Ball, Canvas.GetLeft(Ball) + intSeitlicheBewegung);

            if (Kollision(Ball, balkenOben))
            {
                boolNachOben = false;
            }
            else if (Kollision(Ball, balkenLinks) || Kollision(Ball, balkenRechts))
            {
                intSeitlicheBewegung *= -1;
            }
            else if (Kollision(Ball, Spieler))
            {
                boolNachOben = true;
            }
            else if (Kollision(Ball, balkenUnten))
            {
                //Spiel verloren
                timer.Stop();
                MessageBox.Show("Game Over!", "Pong", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }
```

```
Beispiel_Projekt_Pong | Beispiel_Projekt_Pong.MainWindow | Timer_Tick(object sender, EventArgs e)

1 Verweis
private void Timer_Tick(object sender, EventArgs e)
{
    //Ball Bewegung nach Oben oder Unten
    if(boolNachOben)
    {
        Canvas.SetTop(Ball,Canvas.GetTop(Ball)-10);
    }
    else
    {
        Canvas.SetTop(Ball, Canvas.GetTop(Ball) + 10);
    }

    //Ball Bewegung nach Rechts oder Links
    Canvas.SetLeft(Ball, Canvas.GetLeft(Ball) + intSeitlicheBewegung);

    if (Kollision(Ball, balkenOben))
    {
        boolNachOben = false;
    }
    else if (Kollision(Ball, balkenLinks) || Kollision(Ball, balkenRechts))
    {
        intSeitlicheBewegung *= -1;
    }
    else if (Kollision(Ball, Spieler))
    {
        boolNachOben = true;
    }
    else if (Kollision(Ball, balkenUnten))
    {
        //Spiel verloren
        timer.Stop();
        MessageBox.Show("Game Over!", "Pong", MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}

75 %
```

Fertig ist das Spiel



Mögliche Erweiterungen:

- Spielneustart ohne das Programm zu beenden
- Offline Multiplayer
- Zähler für die Anzahl der Berührungen von Spieler und Ball
- Schwierigkeit nach einer gewissen Zeit erhöhen
- Highscore
- Popups / Boosts
    - Längerer Balken
    - Kürzerer Balken
    - Schnellerer Pong
    - Langsamerer Pong
    - Unsichtbar
    - Mehrere Pong
    - Kurven Pong
    - Hindernisse
- Ballsteuerung
- KI (Computergegner)
- Pongeffekte / Effekte
- Online Multiplayer
- Level / Steigende Schwierigkeit
- Maussteuerung