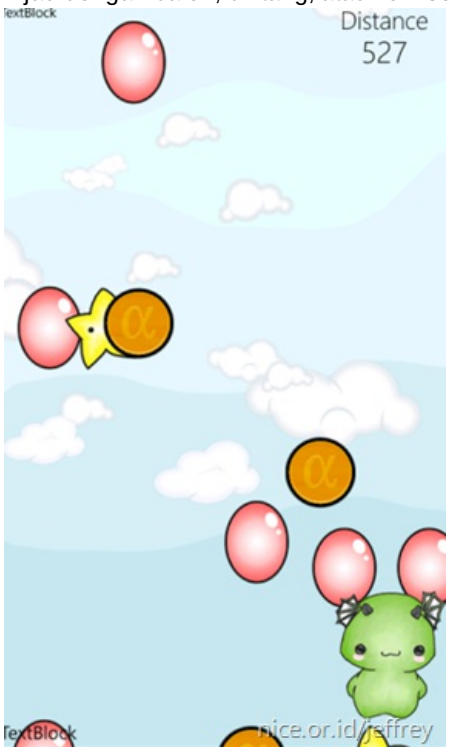
Pixel-Perfect Collision Detection pada Silverlight for Windows Phone

Artikel ini akan menjelaskan bagaimana caranya untuk melakukan *pixel-perfect collision detection* pada Silverlight for Windows Phone.

Pendahuluan

Dalam pengembangan sebuah game, kebutuhan untuk melakukan *collision detection* (pendeteksian tumbukan) antara 2 objek sangat sering sekali diperlukan. Terutama apabila *collision detection* tersebut dilakukan pada 2 objek PNG *images*, monster hijau dengan balon, bintang, atau koin seperti gambar di bawah ini:



Prosedur

Langkah-langkah untuk melakukan *pixel-perfect collision detection* pada Silverlight for Windows Phone adalah sebagai berikut:

1. Buat `CharacterUserControl` yang berisi `Image` dari monster hijau dan beri nama **img1**.

2. Buat `StarUserControl` yang berisi `Image` dari bintang dan beri nama **img1**.

3. Tambahkan instans dari `CharacterUserControl` dan `BalloonUserControl` pada `MainPage`.
4. Definisikan *method* `GetWriteableBitmap()`.

(C) Copyright Nokia 2014. All rights reserved.

```
protected WriteableBitmap GetWriteableBitmap(FrameworkElement control)
{
    WriteableBitmap wb = new WriteableBitmap((int)control.Width,
(int)control.Height); ;
    wb.Render(control, new TranslateTransform());
    wb.Invalidate();
    return wb;
}
```

5. Karena proses pembuatan `WriteableBitmap` sangatlah sering, maka `WriteableBitmap` dari `Image` tersebut akan disimpan pada *property* `Tag` dari `Image` pada saat *constructor* dipanggil.

```
Image imgChar = chrMain.FindName("img1") as Image;
WriteableBitmap wbChar = GetWriteableBitmap(chrMain);
imgChar.Tag = wbChar;

Image imgStar = str1.FindName("img1") as Image;
WriteableBitmap wbStar = GetWriteableBitmap(str1);
imgStar.Tag = wbStar;
```

6. Definisikan *method* `CheckCollision()` yang akan dipanggil untuk melakukan pengecekan. Pada *method* ini, pengecekan akan dilakukan untuk setiap 10 pixel x pada setiap 10 pixel y.

```
/**
 * Check the collision between control1 - control2 and also controlElem1 -
controlElem2
 **/
protected bool CheckCollision(FrameworkElement control1, FrameworkElement
controlElem1,
                                FrameworkElement control2, FrameworkElement
controlElem2)
{
    if (control2.Tag != null)
        return false;
    if (control2.Margin.Top + control2.Height < control1.Margin.Top)
        return false;

    if (control2.Margin.Left + control2.Width >= control1.Margin.Left &&
        control2.Margin.Left <= control1.Margin.Left + control1.Width)
    {
        bool bCollision = false;
        Point ptCheck = new Point();

        // do a more accurate pixel hit test in every 10 pixels
        for (int x = Convert.ToInt32(control1.Margin.Left);
                x < Convert.ToInt32(control1.Margin.Left + control1.Width); x +=
10)
        {
            for (int y = Convert.ToInt32(control1.Margin.Top);
                    y < Convert.ToInt32(control1.Margin.Top + control1.Height); y
+= 10)
            {
                ptCheck.X = x;
                ptCheck.Y = y;
```


(C) Copyright Nokia 2014. All rights reserved.

```
                if (CheckCollisionPoint(ptCheck, control1, controlElem1))
                    if (CheckCollisionPoint(ptCheck, control2, controlElem2))
                    {
                        bCollision = true;
                        break;
                    }
            }
            if (bCollision) break;
        }
        return bCollision;
    }
    return false;
}
```

7. Definisikan *method* `CheckCollisionPoint()` yang akan dipanggil untuk melakukan pengecekan di setiap pixel yang telah ditentukan.

```
/**
 * Check the collision between control and controlElem in the selected pt
 **/
public bool CheckCollisionPoint(Point pt, FrameworkElement control,
FrameworkElement controlElem)
{
    // NOTE that we saved the WB in the Tag object for performance
    WriteableBitmap wb = controlElem.Tag as WriteableBitmap;

    int width  = wb.PixelWidth;
    int height = wb.PixelHeight;

    double offSetX = control.Margin.Left;
    double offSetY = control.Margin.Top;

    double xCur = pt.X - offSetX;
    double yCur = pt.Y - offSetY;

    if (xCur < 0 || xCur >= width || yCur < 0 || yCur >= height)
        return false;

    int offset = (int) ((width * yCur) + xCur);

    if (offset >= wb.Pixels.Count())
        return false;

    return (wb.Pixels[offset] != 0);
}
```

8. Pada *event handler* `TimerTick` pada *game logic*, lakukan *collision detection* tersebut.

```
Image imgChar = chrMain.FindName("img1") as Image;
Image imgStar = str1.FindName("img1") as Image;
if (CheckCollision(chrMain, imgChar, str1, imgStar))
{
    // game over
    DoGameOver();
    break;
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

Kesimpulan

Tidaklah sulit untuk melakukan *collision detection* pada Silverlight for Windows Phone dan cukup hanya dengan langkah-langkah yang sederhana. Sebagai catatan, pendekatan ini memang mudah untuk diimplementasikan namun jika *collision detection* dilakukan pada banyak objek sekaligus maka performa aplikasi dapat menurun secara drastis. Sebaiknya, tidak melakukan pengecekan pada gambar PNG karena akan memerlukan waktu komputasi yang lama.

Semoga bermanfaat!

Referensi

Beaulieu, A. 2009. Improved HitTest Method for Silverlight 3.

(C) Copyright Nokia 2014. All rights reserved.