Generate Break Away Thresholds

```
/// <summary>
/// Calculate the smallest/biggest X value coordinate on screen where
/// bezier curve will fit within bounds of screen
/// </summary>
/// <param name="leftMostEnemyPos">Coordinate of the left most enemy</param>
/// <param name="rightMostEnemyPos">Coordinate of the right most enemy</param>
/// <returns>A list containing two vectors, the smallest coordinate where a bezier curve will
fit and vice versa</returns>
public List<Vector2> GenerateBreakAwayThresholds(Vector2 leftMostEnemyPos, Vector2
rightMostEnemyPos)
{
    List<Vector2> constraintCoordinates = new List<Vector2>();

    Vector2 currentSmallest = leftMostEnemyPos, currentBiggest = rightMostEnemyPos;

    while (currentBiggest.X < GraphicsDevice.Viewport.Width - 10)
    {
        GenerateCurveRight(rightMostEnemyPos, GraphicsDevice.Viewport);

        //Find the largest x coordinate in the lst of coordinates
        for (int i = 0; i < path.Count; i++)
        {
            if (currentBiggest.X < path[i].X)
            {
                currentBiggest = path[i];
            }
        }

        if (currentBiggest.X < GraphicsDevice.Viewport.Width - 10)
        {
            rightMostEnemyPos.X += 10;
        }
    }

    constraintCoordinates.Add(rightMostEnemyPos);

    while (currentSmallest.X > 10)
    {
        GenerateCurveLeft(leftMostEnemyPos, GraphicsDevice.Viewport);

        //Find the smallest x coordinate in the list of coordinates
        for (int i = 0; i < path.Count; i++)
        {
            if (currentSmallest.X > path[i].X)
            {
                currentSmallest = path[i];
            }
        }

        if (currentSmallest.X > 10)
        {
            leftMostEnemyPos.X -= 10;
        }
    }

    constraintCoordinates.Add(leftMostEnemyPos);
    return constraintCoordinates;
}
```