Appendix

Toevoegen van Undo mvb Stack<>

We gebruiken een Stack<> collectie[7] om een beperkte geschiedenis van de veranderingen op de kaart bij te houden.

Eerst definiëren we een klasse waarin we telkens de vorige waarde van een aangepast blokje kunnen bewaren:

```
class Undo
{
    public int X { get; set; }
    public int Y { get; set; }
    public int OriginalValue { get; set; }
}
```

Vervolgens maken we een Stack<Undo> aan. Ter demonstratie geven we een begincapaciteit mee, zodat de gebruiker niet z'n volledige verloop kan terugdraaien, maar enkel de laatste 5 (in dit geval) aanpassingen:

```
Stack<Undo> UndoHistory = new Stack<Undo>(5);
```

We voegen een Undo knop toe aan onze UI en zullen vervolgens volgende code steeds uitvoeren:

```
if (UndoHistory.Count > 0)
{
    Undo lastaction = UndoHistory.Pop();
    currentMap.SetElement(lastaction.X, lastaction.Y, lastaction.OriginalValue);
    LoadMapOnView();
}
```

We controleren dus eerst of er nog een actie op de undo-stack staat. Zo ja, dan halen we deze eraf en passen we het aangepaste element aan naar z'n originele waarde.

Wat ons nu nog rest is in alle code waar een currentMap.SetElement(…) doorgaat, vlak ervoor steeds de originele van het element in de stack te bewaren (geel). In voorgaande demonstratie is dat dus enkel in de `mapCanvas_MouseLeftButtonUp` eventhandler:

```
if (cmbBrush.SelectedIndex > -1)
{
    Point click = e.MouseDevice.GetPosition(mapCanvas);
    int x = (int)((click.X / blokscale)) - 1;
    int y = (int)((click.Y / blokscale)) - 1;
    var t = (cmbBrush.SelectedItem as ComboBoxItem).Content.ToString();
    Undo newAction = new Undo() { X = x, Y = y, OriginalValue = (int)currentMap.GetElement(x, y) };
    UndoHistory.Push(newAction);

    currentMap.SetElement(x, y, Convert.ToInt32(t));

    LoadMapOnView();
}
```

[7] Meer info https://codevan1001nacht.wordpress.com/2013/11/04/collections-en-generic-collections/