Practicum Informatica Werktuigen
2048 in Visual Basic .NET

22 oktober 2014

Om jullie extra te motiveren voor het practicum van Informatica Werktuigen, hebben we de opdracht gesitueerd in de spelletjeswereld. Aangezien de meest recente 3D games iets te complex zijn, hebben we voor dit practicum gekozen voor een eenvoudiger spel: *2048*.

1 Het spel 2048

De originele versie van het spel kan je online vinden (`http://gabrielecirulli.github.io/2048/`), maar voor alle zekerheid zullen we hier kort de werking van *2048* uit de doeken doen.

Het spelbord Het bord is een vierkant, verdeeld in 4×4 tegels. Iedere tegel kan ofwel leeg zijn ofwel een getal van een macht van 2 bevatten (i.e. 2, 4, 8, 16, etc.) Ieder getal heeft ook zijn eigen specifieke tegel kleur.

Het spel Initieel bevat het spelbord een 2- of een 4-tegel op een willekeurige plaats. Het spel werkt met zetten. Iedere zet controleert de speler het spel met behulp van de directionele pijltjestoetsen op het toetsenbord (omhoog, omlaag, links of rechts). Wanneer een toets ingedrukt wordt, schuiven alle tegels zo ver mogelijk in die richting. Hierbij worden aangrenzende tegels met hetzelfde getal samengevoegd tot een tegel met als getal hun som. Indien dit geen nuloperatie was (er zijn dus tegels verschoven en/of samengevoegd), wordt er op een lege willekeurige plaats een 2- of een 4-tegel geplaatst. Het spel eindigt indien er niets meer verschoven of samengevoegd kan worden in de vier richtingen.

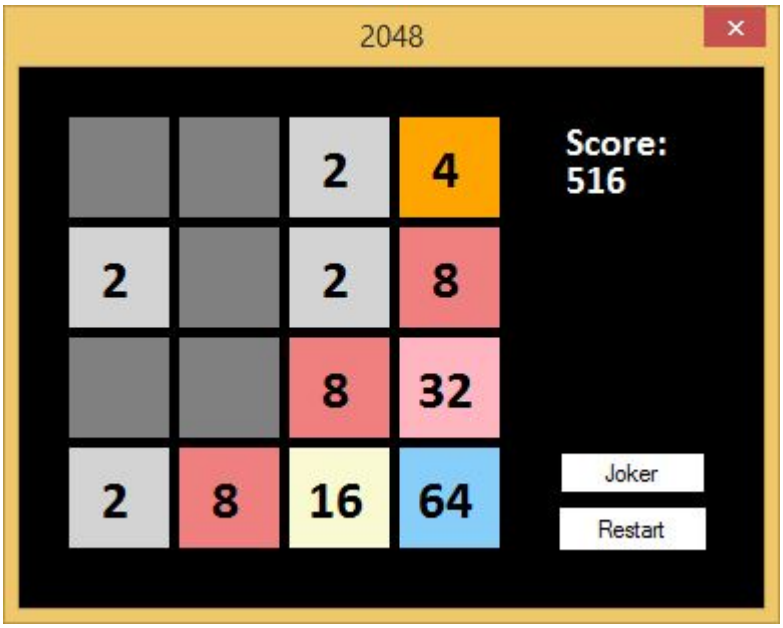
Doel van het spel Het doel van het spel is om zo lang mogelijk door te blijven spelen en op die manier een high score te halen. De score wordt berekend door iedere keer als twee tegels samengevoegd worden de vorige score op te hogen met de waarde van de nieuw bekomen tegel. Zo wordt bijvoorbeeld bij het samenvoegen van twee 4-tegels de score opgehoogd met 8.

In principe kan dit spel eeuwig doorgaan zolang de speler zichzelf niet vastwerkt. Voor dit practicum is het voldoende om bij het halen van 2048 het spel te stoppen.

2 De opgave

De opgave bestaat er dus in het *2048* spel te implementeren in Visual Basic .NET. In deze sectie vind je een opsomming terug van wat we concreet van jullie verwachten.

Zoals in de cursus en de oefenzittingen begin je eerst met het ontwerpen van de user interface. Deze dient het 4×4 rooster van tegels weer te geven. Buiten het rooster is er ook nog een vlak voorzien voor de huidige score, samen met een 'New Game' knop. De speler controleert het spel met de directionele pijltjestoetsen. Een voorbeeld van een layout vind je in Figuur 1. Je mag ook wat creatiever zijn en je eigen layout bedenken. De objecten die aanwezig moeten zijn, kan je afleiden uit de vereisten die we aan je programma stellen.



Figuur 1: Een mogelijke layout

Het programma moet minstens aan volgende criteria voldoen:

- Het speelveld moet bestaan uit een rooster van 4×4 tegels. Elke tegel kan bijvoorbeeld bestaan uit een vierkant `Label` object.
- De tegels zijn ofwel leeg, of hebben een waarde van 2^n (i.e. 2, 4, 8, 16, 32, 64, 128, etc.). Hoewel het spel in principe kan doorgaan tot het spelbord vast zit, is het voldoende om als hoogste tegel 2048 te kiezen. Iedere tegelwaarde heeft ook zijn eigen kleur, die je zelf mag kiezen of mag inspireren op het originele spel. (`http://gabrielecirulli.github.io/2048/`)
- De speler controleert het verschuiven van de tegels met de directionele pijltjestoetsen op het toetsenbord. Bij het succesvol verschuiven in een

richting worden alle tegels zo ver mogelijk naar die richting verschoven, en worden alle tegels met dezelfde waarde samengevoegd tot een tegel van hun gecombineerde waarde.

- Na een succesvolle zet (**enkel** wanneer er effectief tegels verschoven of samengevoegd zijn; het spelbord is dus veranderd), wordt er op een *willekeurige* lege tegel een waarde van 2 (90% kans) of 4 (10% kans) geplaatst.
- De score wordt als volgt bijgehouden: wanneer een nieuw spel start wordt de score op 0 gezet. Bij iedere succesvolle samenvoeging van twee tegels wordt deze score opgehoogd met de nieuw bekomen waarde. Zo wordt bijvoorbeeld bij het samenvoegen van twee 4-tegels de score opgehoogd met 8.
- Na iedere succesvolle zet van de speler dient de eindconditie van het spel gecontroleerd worden: is het nog mogelijk om in één van de richtingen tegels te verschuiven en/of samen te voegen? Indien *niet* dient er een 'Game Over' `MessageBox` te verschijnen met de optie om opnieuw te beginnen.

3 Uitbreidingen

Indien je de puntjes uit de vorige sectie aan de praat hebt gekregen kan je eens proberen om de volgende uitbreidingen te implementeren.

- Houd naast de huidige score van de speler ook de high score bij. Deze high score dient enkel bewaard te blijven tijdens het uitvoeren van het programma, dus na een herstart zal deze terug op 0 staan.
- Maak het mogelijk om bij een nieuw spel de omvang van het spelbord als speler zelf te kiezen. Het spelbord moet wel altijd vierkant zijn (b.v. 9×9, 5×5, ...).
- Introduceer de mogelijkheid van een *Joker*. In de GUI wordt een knop voorzien om deze éénmaal per spel in te zetten. Wanneer de Joker ingezet wordt, kan de speler m.b.v. een muisklik op een tegel deze tegel leegmaken. Vervolgens kan het spel gewoon verdergespeeld worden met de directionele pijltjestoetsen. De Joker mag slechts éénmaal per spel worden ingezet.

4 Tips 'n Tricks

- Zorg ervoor dat je programma makkelijk te begrijpen is. Dit doe je door voldoende commentaar te voorzien in je code, alsook door duidelijke en zinvolle namen voor je code-elementen te kiezen. Gebruik hiervoor de conventie zoals afgesproken in de oefenzitting over Visual Basic.
- Standaard gaat Visual Basic .NET veel kleine foutjes, zoals het gebruik van ongeïnitialiseerde variabelen of functies die niet altijd een waarde teruggeven, over het hoofd zien. Dit kan echter soms leiden tot fouten in het programma die moeilijk te vinden zijn. Om ervoor te zorgen dat de Visual Basic compiler beter zijn best doet om die veelgemaakte fouten eruit te halen, kan je de volgende twee lijnen helemaal bovenaan je code zetten:

```
Option Explicit On
Option Strict On
```

- De eenvoudigste manier om een tegel te visualiseren is met behulp van een `Label`. Door de `AutoSize` property van het `Label`-object op `False` te zetten, kan het `Label` vierkant gemaakt worden. Het inkleuren gebeurt door de `BackColor` property aan te passen naar de gewenste kleur.

- Om een eenvoudig en gestructureerd programma te krijgen is het belangrijk dat je de juiste datastructuren gebruikt voor het opslaan van het rooster. Bij voorkeur gebruik je een 2-dimensionale `Array`. Dit kan je beschouwen als een 4×4 matrix die al de `Label` objecten bevat die de tegels visualiseren. Specifieke objecten uit deze matrix kunnen dan opgevraagd worden door de X- en de Y-coördinaat mee te geven (v.b. `rooster(1,2)`). Deze concepten zijn deels uitgewerkt in onderstaande code:

```
'Matrix van 4 breed (0 tot 3) en 4 hoog (0 tot 3)
Dim rooster(3, 3) As Label

'Initialisatie van het rooster
For i As Integer = rooster.GetLowerBound(0) To rooster.GetUpperBound(0)
    For j As Integer = rooster.GetLowerBound(1) To rooster.GetUpperBound(1)
        rooster(i, j) = New Label
        rooster(i, j).Size = New Size(50, 50)
        rooster(i, j).TextAlign = ContentAlignment.MiddleCenter
        rooster(i, j).Font = New Font("Calibri", 20, FontStyle.Bold)
        rooster(i, j).Name = "tile_" & i & "_" & j

        'x,y locatie in het venster
        rooster(i, j).Location = New Point(... , ...)
        'kleur van de tegel
        rooster(i, j).BackColor = ...
        'text op de tegel (initieel leeg)
        rooster(i, j).Text = ""

        Me.Controls.Add(rooster(i, j))
    Next
Next
```

- Het is structureel best om de code die beslist welke tegelwaarde op welke kleur mapt te isoleren in een functie. Op deze manier wordt code-duplicatie tegengegaan. Een mogelijke functie heeft als argument de nieuwe waarde en geeft een `Color` object terug dat onmiddellijk kan toegekend worden aan de `BackColor` property van een `Label`. We geven een skelet voor deze code hieronder.

```
Private Function Tile_Color(ByVal value as String) As Color

    Select Case value
        Case ""
```

```
            'Lege tegel
            Return Color.LightGray
        Case "2"
            '2-tegel
            Return Color...
        Case "4"
            ...
            ...
    End Select
End Function

...

raster(2,3).Text = "4"
raster(2,3).BackColor = Tile_Color("4")
```

- De speler controleert het spel via de directionele pijltjestoetsen. Om op deze events te reageren moet er een event handler op `Form` niveau geschreven worden die `KeyUp` events afhandelt. Ook is het uitermate belangrijk dat de KeyPreview property van het Form object op True staat. Indien niet, zal enkel het controle object (Buttons, Labels, etc.) dat geselecteerd is events ontvangen, en niet het Form. Hieronder volgt voorbeeldcode die dit doet.

```
Private Sub Form1_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load

    ...
    Me.KeyPreview = True
    ...

End Sub

Private Sub Form1_KeyUp(ByVal sender As Object, _
    ByVal e As KeyEventArgs) Handles Me.KeyUp

    Select Case e.KeyCode
        Case Keys.Right
            'Rechts
            ...
        Case Keys.Left
            'Links
            ...
        Case Keys.Up
            'Omhoog
            ...
        Case Keys.Down
            'Omlaag
            ...
    End Select
```

```
End Sub
```

Maak zoveel mogelijk gebruik van functies om code-duplicatie tegen te gaan!

- Op twee punten in het programma zal het nodig zijn om willekeurig keuzes te maken: bij het selecteren van een lege tegel na een succesvolle zet van de speler, en bij het kiezen of er een 2 (90% kans) of een 4 (10% kans) ingevuld moet worden op deze lege tegel. Onderstaande code laat zien hoe je best te werk gaat. Let op, onderstaande code houdt geen rekening met het feit of de gekozen tegel al dan niet leeg is. Hier dien je nog een oplossing voor te vinden.

```
Dim rnd = New Random()

'Haalt een willekeurig element uit een lijst genaamd 'list'
Dim randomElement = list(rnd.Next(0, list.Count))

'If-else constructie met kansverdelingen
'Er wordt een willekeurig getal opgevraagd [0,10[
'bijgevolg is er 10 pct. kans dat dit getal 0 is.
If rnd.Next(0,10) = 0 Then
    '10 pct. kans
    ..
Else
    '90 pct. kans
    ..
End If
```

- Het kan soms noodig zijn om één enkele event handler te gebruiken voor meerdere GUI objecten. In dit practicum zal dit bij name nodig zijn om de tegels klikbaar te maken zonder 16 identieke Clicked event handlers te schrijven. Onderstaande code laat zien hoe je best te werk gaat.

```
'Gemeenschappelijke handler
Private Sub onLabelClick(ByVal sender As Label, ByVal e As EventArgs)
    'sender bevat aangeklikte Label
    ...
End Sub

'Voor ieder klikbaar Label:
AddHandler rooster(i, j).Click, AddressOf onLabelClick
```

5 Afspraken

Je practicum moet **ten laatste op maandag 3 november 2014** ingeleverd worden. Dien op die dag voor 12:00 een ZIP-bestand in met je code. Geef het ZIP-bestand een naam volgens deze conventie: `iw_<VOORNAAM>_<NAAM>_vb.zip`, waar `<VOORNAAM>` en `<NAAM>` uiteraard vervangen worden.

Voor vragen en problemen kan je altijd terecht op het forum voor dit practicum op Toledo. Dit kan je bereiken via `http://toledo.kuleuven.be`

Hou er ook rekening mee dat er een demo van dit practicum volgt. Je zal dan een aantal zaken aan je practicum moeten veranderen. De datum wanneer deze demo zal doorgaan zal later bekend gemaakt worden. De demo wordt gegeven in **Visual Basic .NET 2012**. Programma's gemaakt in oudere versies van VB.NET kunnen hier zonder problemen op uitgevoerd worden. Let op! Bij nieuwere versies kan je mogelijk wel in de problemen komen! Mensen die niet beschikken over Visual Basic kunnen het practicum maken in de pc-klassen waar de oefenzittingen over VB.NET plaatsvonden, of kunnen een gratis versie van VB.NET downloaden (zie Toledo).

Succes!

Yolande Berbers
Jasper Bogaerts
Job Noorman
Thomas Winant
Raoul Strackx
Mathy Vanhoef
Wilfried Daniels