

Deel GFX2 - Overzicht en oefeningen

Uitbreiding

Nu we weten hoe we met een Graphics object kunnen tekenen, zouden we graag ons Playground project uitbreiden met verschillende soorten figuren. Ook willen we extra functionaliteit toevoegen (bv. figuren kunnen aanklikken, animaties, enz.).

Het concept 'een figuur' (nml. wat dit precies voorstelt en welke variaties er op dit thema zijn), staat nu centraal in onze oplossing en we willen dit dan ook weerspiegeld zien in de structuur van ons programma. We doen dit met klassen en objecten, dit zijn immers de “kennisbrokjes” waarmee we onze programma’s vormgeven.

In de oplossingen van de vorige opdrachten was dit niet het geval! Er waren geen Cirkel of Rechthoek klassen, laat staan een 'Diagram' concept waar we extra functionaliteit konden aanhangen (bv. figuren toevoegen of verwijderen, nagaan op welke figuur geklikt werd, figuren verplaatsen, enz).

De figuren waren slechts impliciet aanwezig in de zin dat ze wel op het scherm te zien waren, maar het enige dat we ervan terug vinden in onze code zijn flarden method calls zoals DrawLine en FillRectangle.

In de volgende oefening zullen we deze stap zetten en onze figuren expliciet maken in de code a.d.h.v. klassen (en bijgevolg, objecten tijdens de uitvoering).

Oefening DGFX2.01

Maak een nieuw "Windows Forms App" project (zie deel GFX1) met naam 'Diagram'.

Verwijder de Form1.cs en Program.cs files, we zullen ze niet nodig hebben.

Open het bestand les 'deel-gfx2-oefeningen-code.zip' en stop de inhoud in dit project.

Dit project volgt grotendeels de structuur van het Playground project en bevat drie klassen

Domain.Diagram

Dit is een verzameling van figuren, een tekening als het ware

Domain.Rectangle

Dit stelt een rechthoek voor die op een Diagram kan geplaatst worden en zichzelf kan tekenen

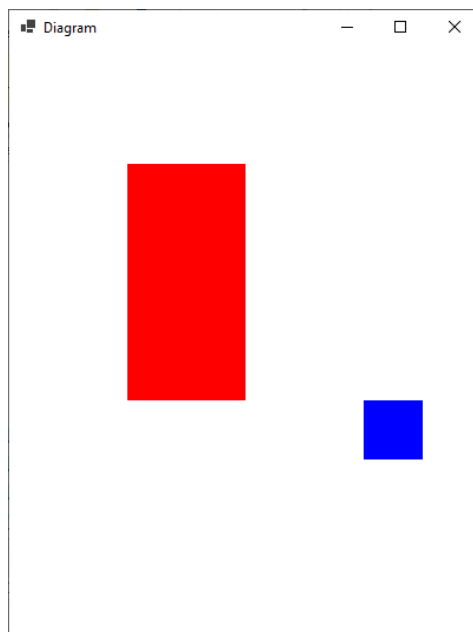
UI.DiagramControl

Dit is een de UI-component die een Diagram kan visualiseren. Dit is analoog aan de ViewControl uit het Playground project

Waarom hebben we een Diagram klasse nodig? Zou een List (of array) niet al voldoende zijn om onze figuren te verzamelen? Voorlopig wel, maar we gaan op dit Diagram niveau later nog wat functionaliteit bijvoegen en dat zou niet lukken met List (of array). Vandaar dat we het meteen goed zullen doen.

Kijk eens naar DiagramControl.OnPaint alsook naar de code van de klasse Rectangle. Je zult zien dat een rechthoek ook een kleur heeft en al dan niet gevuld kan zijn.

Implementeer de methods `Diagram.AddRectangle` en `Diagram.DrawAll`. Als je het programma dan runt krijg je volgende te zien :



Oefening DGFX2.02

In de Program.Main method staat o.a. de volgende regel :

```
Rectangle r1 = new Rectangle(100, 100, 100, 200, System.Drawing.Color.Red, true);
```

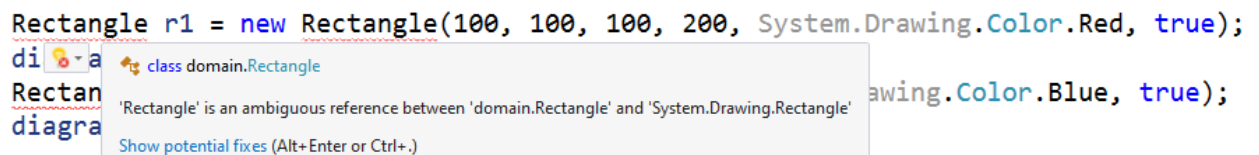
Zoals je kunt zien gebruiken we hier de volledige naam van de Color klasse. Je zou toch verwachten dat we bovenaan deze file een using statement zouden zetten :

```
using System.Drawing;
```

zodat die regel korter kan geschreven worden als

```
Rectangle r1 = new Rectangle(100, 100, 100, 200, Color.Red, true);
```

Probeer dit eens uit en let op de foutmelding die je krijgt :



```
Rectangle r1 = new Rectangle(100, 100, 100, 200, System.Drawing.Color.Red, true);  
diagram.AddRectangle(r1);  
Rectangle r2 = new Rectangle(300, 300, 50, 50, System.Drawing.Color.Blue, true);  
diagram.AddRectangle(r2);
```

'Rectangle' is an ambiguous reference between 'domain.Rectangle' and 'System.Drawing.Rectangle'

Show potential fixes (Alt+Enter or Ctrl+.)

Wat is er hier precies aan de hand?

Oefening DGFX2.03

Neem de oplossing van de vorige opdracht en geef een schematische voorstelling van de objecten in het geheugen (i.e. een object diagram) op het moment dat de DiagramControl.OnPaint() method wordt opgeroepen. Je kunt natuurlijk enkel de objecten tekenen van klassen die je kent.

Oefening DGFX2.04

Breid de vorige oplossing uit zodat er ook cirkels kunnen toegevoegd worden aan een diagram.

De initialisatie van het diagram in de Program.Main method wordt dan :

```
Rectangle r1=new Rectangle(100,100,100,200, System.Drawing.Color.Red, true);  
diagram.AddRectangle(r1);  
Circle c1=new Circle(150,150,60, System.Drawing.Color.Green, true);  
diagram.AddCircle(c1);  
Circle c2=new Circle(325,285,30, System.Drawing.Color.Yellow, true);  
diagram.AddCircle(c2);  
Rectangle r2=new Rectangle(300,300,50,50, System.Drawing.Color.Blue, true);  
diagram.AddRectangle(r2);
```

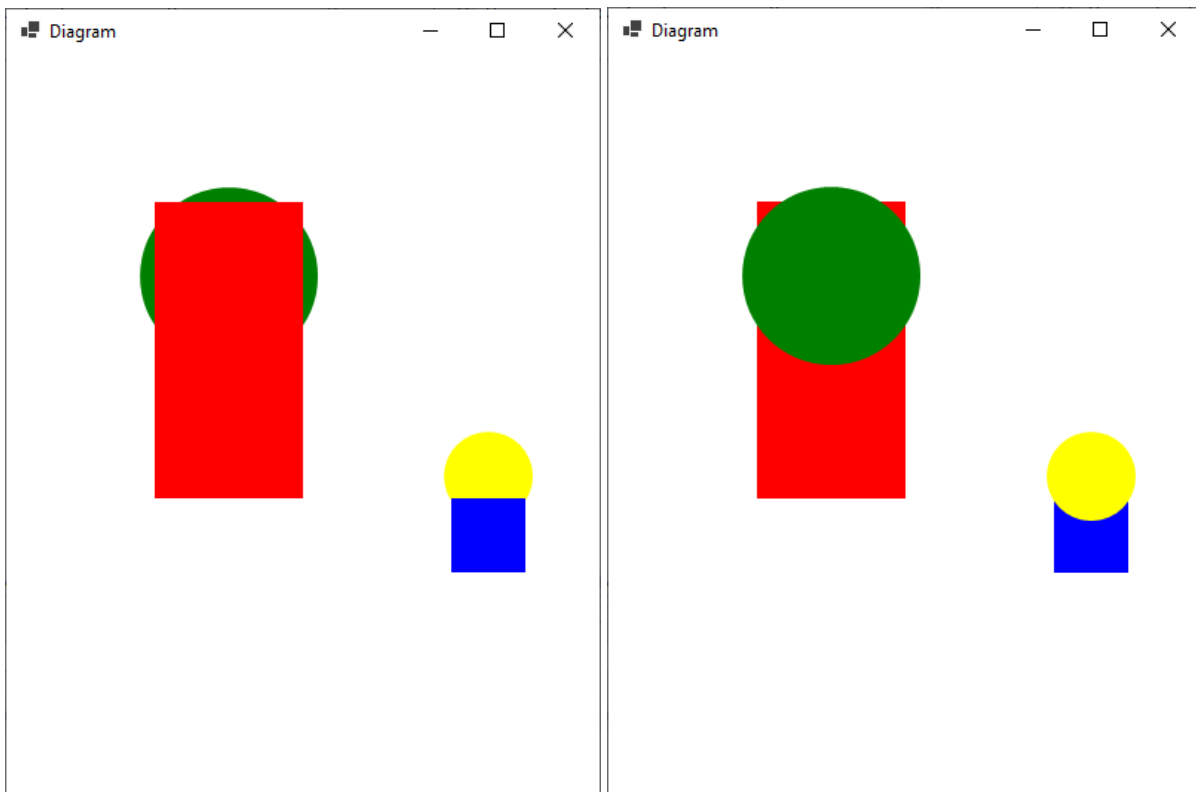
Schrijf dus een klasse Circle met de volgende properties :

CenterX, CenterY, Radius, Color en IsFilled

die zichzelf kan tekenen en een constructor heeft om al deze waarden in 1 keer in te stellen (analoog met klasse Rectangle dus).

Je zult ook code moeten toevoegen in Diagram om Diagram.AddCircle aan de praat te krijgen, en ervoor te zorgen dat de cirkels daadwerkelijk getekend worden als het diagram gevraagd wordt om alles te hertekenen (in DrawAll dus).

Als alles goed is krijg je dan een van volgende mogelijkheden te zien :



Waarvan zou het afhangen of je de linkse of de rechtse situatie te zien krijgt?

Denkoefening

Stel je nu eens voor dat we een grafische editor of een spelletje willen bouwen waarbij meer dan 2 soorten figuren gebruikt worden. Bijvoorbeeld driehoeken, sterren, afbeeldingen, etc.

Welke soorten klassen komen er dan bij?

Wat voor wijzigingen verwacht je in de Diagram klasse te moeten doorvoeren?