

## Visual C# installeren

Voordat we aan de slag kunnen met het maken van C# (spreek uit C Sharp) programma's moet de IDE (Integrated Development Environment) geïnstalleerd worden. We gebruiken hiervoor de gratis versie van Visual C# 2008.

Het downloaden en installeren gaat als volgt:

- Open de internet browser en ga naar [www.microsoft.com/express](http://www.microsoft.com/express)
- Kik op 'Downloads'
- Kik op 'Visual C# 2008 Express Edition'
- Selecteer 'English'. (Er is helaas geen nederlandstalige versie.)
- Klik op 'Free Download'
- Sla het installatiebestand 'vcsetup.exe' ergens op.
- Start de installatie door 'vcsetup.exe' te openen.
- Volg de instructie op het scherm. Dat wil zeggen: aangeven dat je de licentie voorwaarden hebt gelezen en een aantal keren op 'next' klikken.
- De installatie doorloopt nu 5 stappen. Dit duurt even. De installer haalt bepaalde onderdelen van internet.
- Klik op 'exit' als de installatie is voltooid.

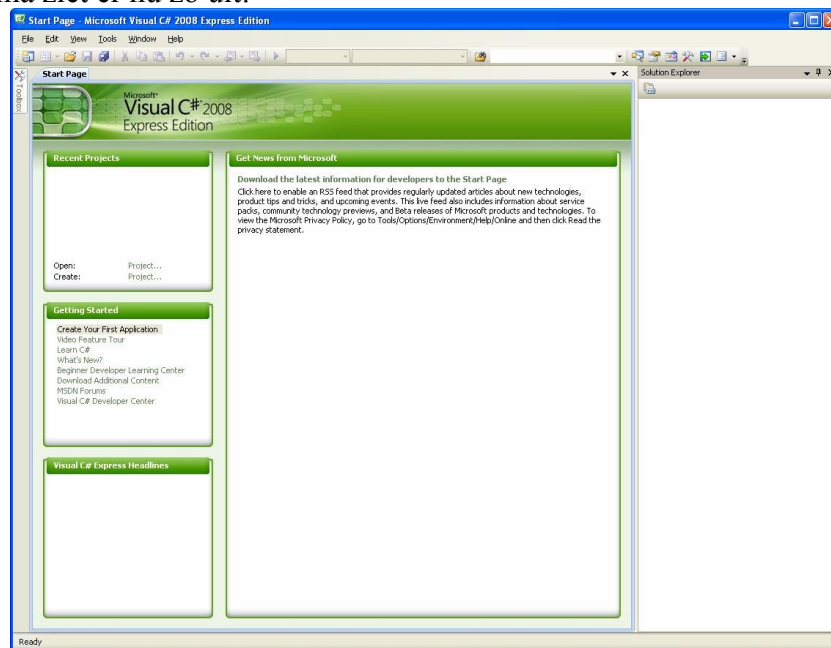
## Een nieuw project maken

We gaan een programma maken die een amplitude gemoduleerde sinus genereert. Deze sinus wordt op het scherm getekend en de frequentie en amplitude van zowel de gemoduleerde en modulerende sinus zijn door de gebruiker in te stellen.

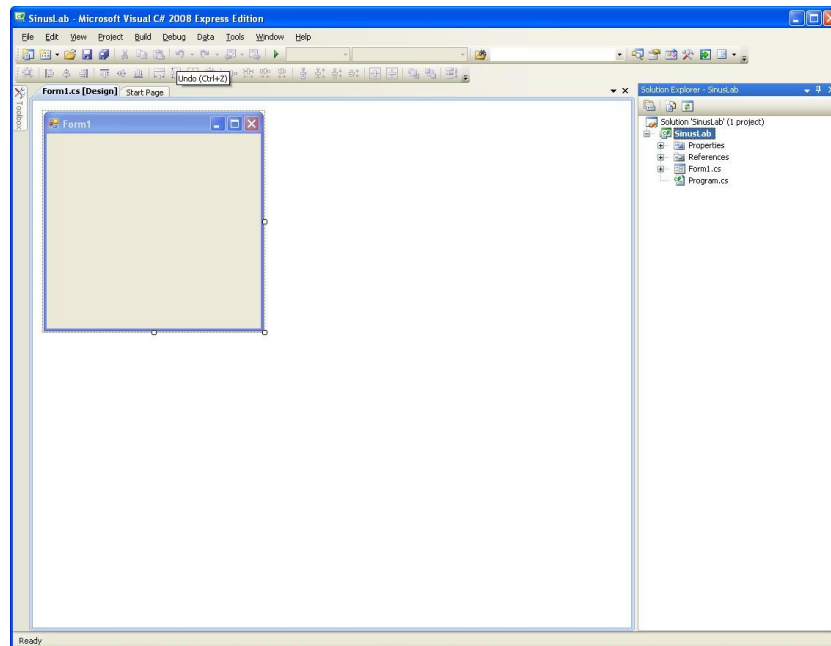
De eerste stap van het maken van programma is het creëren van het project waarbinnen het programma wordt gemaakt. Dit gaat als volgt:

- Start 'Visual C# 2008' (Als dit de eerste keer na installatie is duurt het even, want het e.e.a. moet nog geconfigureerd worden)

Het programma ziet er nu zo uit:



- Klik op 'File -> New Project'
- Nu moeten we kiezen voor het type project dat we willen maken. We willen een Windows programma schrijven met één of misschien meerdere vensters (forms). Selecteer daarom 'Windows Forms Application'. (Deze is eigenlijk al geselecteerd.)
- Nu moet het project en het programma wat we willen maken nog een naam krijgen. Type bij 'Name' de naam 'SinusLab'.
- Klik op 'OK' en er gebeurt wat hocuspokus en het resultaat ziet er als volgt uit:



We hebben nu al een werkend programma. Het doet echter nog lang niet alles wat we willen, maar je kunt hem wel opstarten, maximaliseren, minimaliseren en afsluiten, zoals we dat van de meeste Windows programma's gewend zijn. Het scherm met de naam 'Form1' is het hoofdscherm dat ook te zien is als het programma is opgestart. Klik maar eens op het groene pijltje.



Het scherm van Visual C# verandert, de IDE gaat in debug mode. Ook zien we het scherm met de naam 'Form1' nogmaals in beeld verschijnen. Dit scherm kunnen we minimaliseren, maximaliseren en afsluiten.

Sluit Form1 af door op het rode kruis te klikken, zodat het scherm van Visual C# weer terug verandert.

We gaan nu hetgeen we tot nu toe gemaakt hebben opslaan:

- Klik op 'File -> Save All'.
- We kunnen nu een map aangeven waar we het project willen opslaan en of een project-map moet worden aangemaakt. Laat alles staan zoals het staat, onthoud de map waar het weggeschreven wordt en klik op 'Save'. Die map zal zoiets zijn als:  
 "C:\Documents and Settings\PietjePuk\Mijn documenten\Visual Studio 2008\Projects"
- Sluit nu 'Visual C# 2008' af. In het volgende hoofdstuk gaan we weer verder waar we gebleven zijn.

We hebben het zelfgemaakte programma 'SinusLap' al een keer gedraaid in debug mode. We kunnen het programma ook starten door de executable 'SinusLap.exe' te starten. 'SinusLap.exe' is door de IDE gecompileerd en weggeschreven in de project map.

Navigeer met de Windows verkenners naar

Mijn documenten\Visual Studio 2008\Projects\SinusLab\SinusLab\bin\Debug en

dubbel klik op 'SinusLap.exe'. Form1 verschijnt weer in beeld. (Vergeet deze niet af te sluiten alvorens met het volgende hoofdstuk verder te gaan.)

## Solution Explorer en Properties

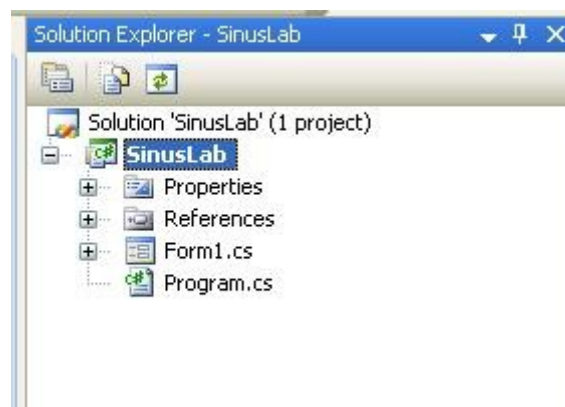
In het vorige hoofdstuk hebben we het project of solution gecreëerd waarbinnen we bouwen aan het programma SinusLab. We hebben als het ware de bouwplaats gemaakt. We kunnen het programma al draaien die enkel met zijn funderingen in het scherm verschijnt.

We gaan nu het project heropenen en Form1 een andere naam en titel geven. We gaan ook het bestand waar Form1 in zit een andere naam geven.

Start 'Visual C# 2008'. We zien nu bij 'Recent Projects' het SinusLab project staan.

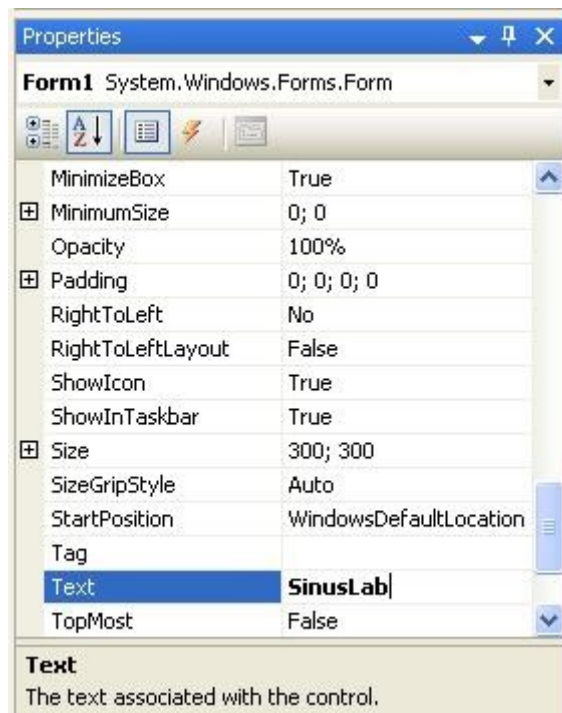


Klik op SinusLab om het project weer te openen.  
Rechts in de 'Solution Explorer' staat 'Form1.cs'.



Dubbelklik op 'Form1.cs'. Form1 wordt nu zichtbaar.

Klik met de rechter muisknop op Form1 en dan op 'Properties'. Rechts onder de Solution Explorer verschijnt nu het Properties scherm. Hier kunnen we de eigenschappen van van Form1 aanpassen.



Verander in het Properties scherm bij Text de tekst 'Form1' in 'SinusLab'. De titel van Form1 veranderd nu van 'Form1' naar 'SinusLab'.

Elke keer als een nieuwe form wordt aangemaakt is de standaard naam 'Form1', 'Form2', 'Form3', enz. Het is aan te bevelen om ook de naam van het form te veranderen in iets wat meer de functie van het form beschrijft. Scroll in het Properties scherm naar boven naar '(Name)' en verander 'Form1' in 'frmSinusLabMain'.

Nu zit het form nog in een bestand genaamd 'Form1.cs', ook deze kan aangepast worden. Klik in de 'Solution Explorer' met de rechter muisknop op 'Form1.cs' en vervolgens op 'rename'. Verander de bestandsnaam in 'guiSinusLabMain.cs'.

Klik op 'File -> Save All'

We hebben nu een project genaamd SinusLab waarbinnen het programma SinusLab.exe gebouwd wordt. Binnen het project SinusLab zit een bestand guiSinusLabMain.cs deze bevat het ontwerp en code van de form frmSinusLabMain, het hoofdscherm van het programma. frmSinusLabMain heeft in de text-propertie de tekst SinusLab staan. Deze tekst verschijnt als titel bovenin het hoofdscherm.

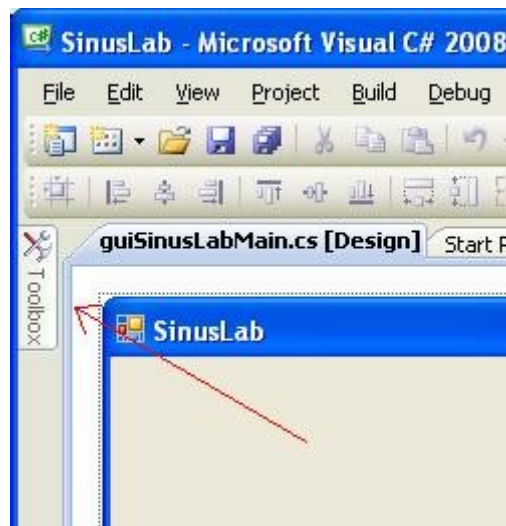
# Containers

We gaan nu frmSinusLabMain verder aankleden, zodat het een bruikbare Grafische User Interface wordt. Er zullen een aantal componenten op het scherm worden gezet waarmee de gebruiker de gewenste frequenties, amplitudes en dergelijke kan instellen. Deze componenten kunnen rechtstreeks op het form gezet worden, of ze kunnen in zogenaamde containers op het form gezet worden. Containers bieden de mogelijkheid het scherm een overzichtelijke lay-out te geven en de componenten te groeperen.

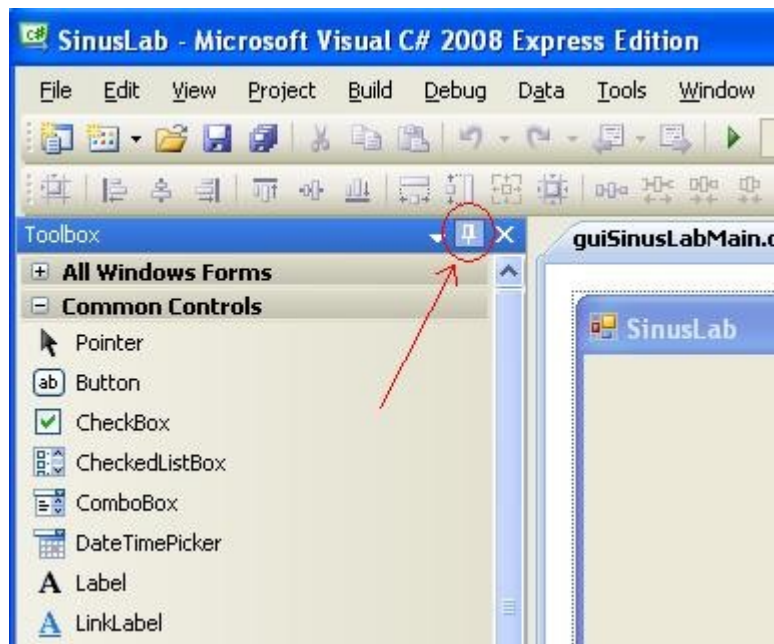
In SinusLab gaan we twee containers gebruiken:

1. Een panel waar we het oscilloscoop beeld in gaan tekenen.
2. Een panel waar we componenten in gaan zetten voor het instellen van frequenties, amplitudes, time/div etc.

De panels zitten net als buttons, invoervelden, vinkboxen e.d. in de de toolbox. Dat is een soort onuitputtelijke voorraadkast waar al dit soort componenten te vinden zijn. Onderstaande afbeelding laat zien waar de Toolbox te vinden is. Zet de muisaanwijzer boven 'Toolbox' en hij schuift het beeld in.



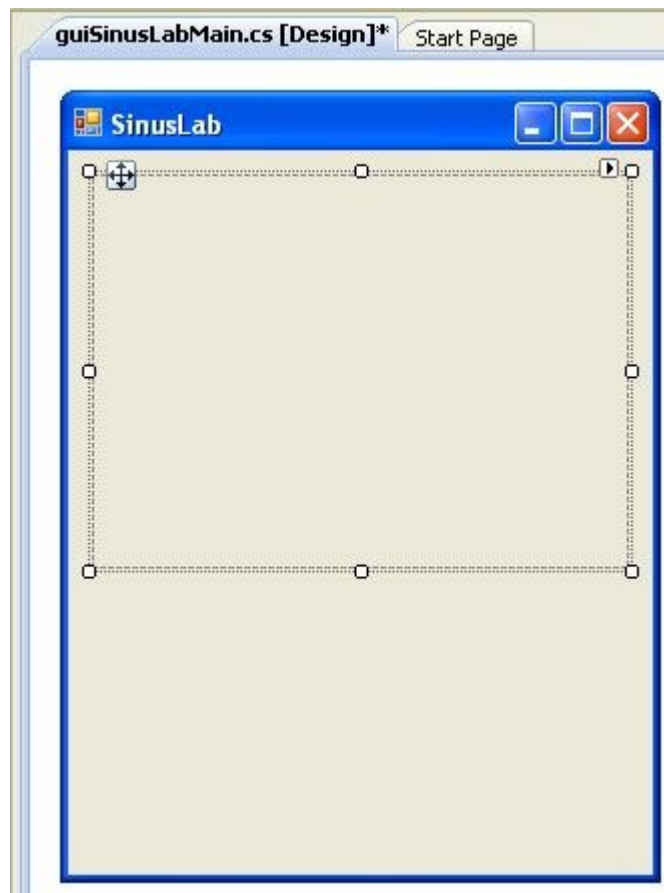
Al de muisaanwijzer de toolbox verlaat, zal deze weer verdwijnen. Dit kan verholpen worden door op de prikker te klikken die in onderstaande afbeelding staat omcirkelt.



Scrol in de toolbox naar 'Containers' en klik op 'Panel'. Klik vervolgens op ergens op het frmSinusLabMain. De panel wordt nu in zichtbaar op het form frmSinusLabMain.

Op de rand van de panel staan blokjes. Door deze blokjes te verschuiven kunnen de plaats en afmetingen van de panel veranderd worden. Klik buiten het panel op het form zodat nu het form blokjes op zijn rand krijgt, waardoor ook hiervan de afmetingen zijn aan te passen. Door weer op het panel te klikken kunnen weer de afmetingen van het panel worden aangepast.

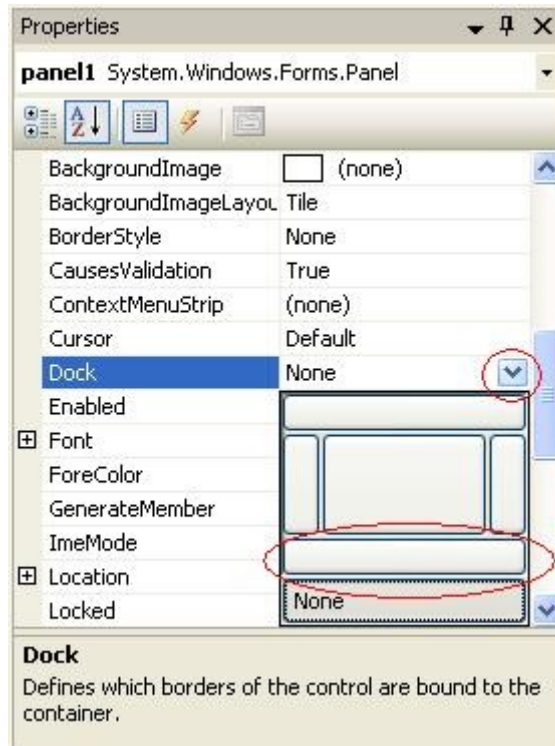
Klik in schuif wat met het form en de panel, zodat het form er als volgt uit ziet:



Net zoals we het form de naam 'frmSinusLabMain' gegeven hebben, gaan we ook het panel een naam geven. Klik op het panel en scrol vervolgens in het properties-scherm naar Name. Verander 'panel1' in 'pnlScope'.

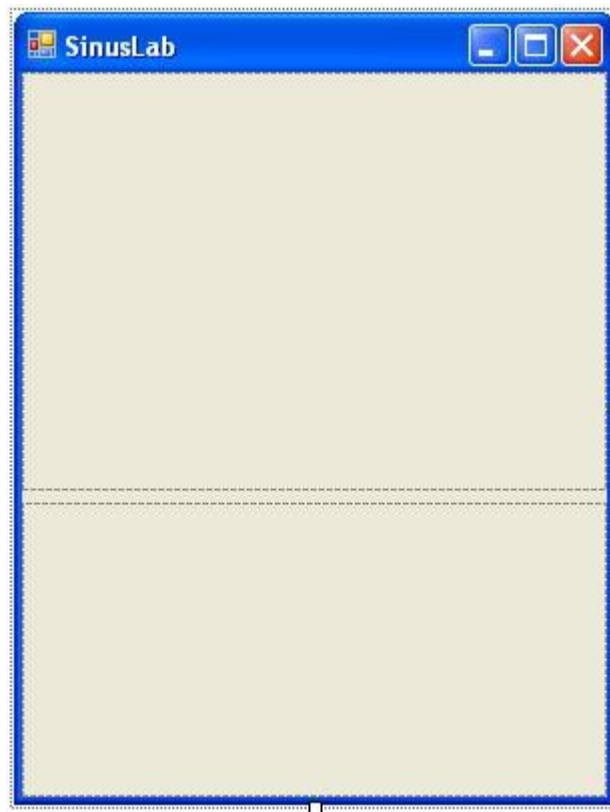
In dit panel gaan we later de sinus tekenen. De invoervelden voor de frequenties en amplitudes komen in een tweede panel. Klik in de toolbox nogmaals op panel en klik vervolgens op het form, maar onder de panel die al geplaatst is. Een tweede panel verschijnt nu op het form.

Ook van dit panel gaan we de name veranderen. Verander in het properties-scherm de Name-property naar 'pnlBottom'. Scrol in het properties-scherm iets naar beneden naar de doc-property. Klik op het pijltje zodat er rechthoeken in beeld komen. Klik vervolgens op de omcirkelde rechthoek. (Zie afbeelding)



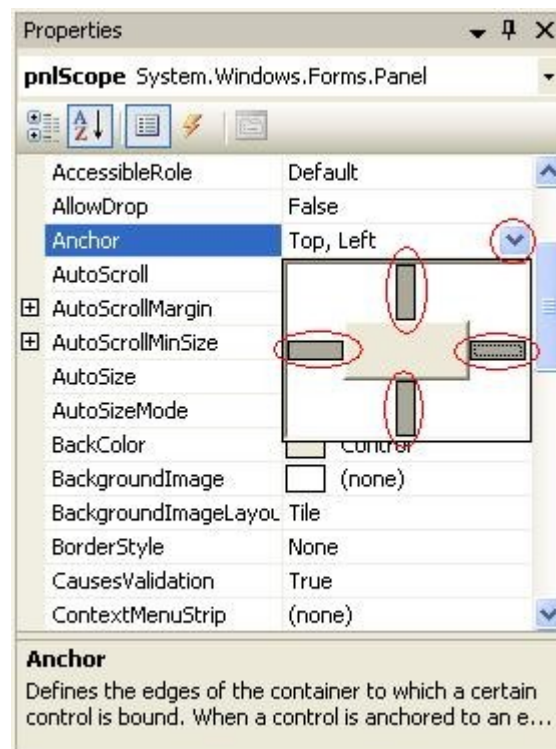
De panel gaat nu als het ware vastzitten aan de onderkant van het form.

Verschuif de randen van pnlBottom en pnlScope zodanig, dat het form eruit ziet als de volgende afbeelding:



Nu moet alleen van `pnlScope` de `Anchor`-property ingesteld worden. Deze property zorgt voor de verankering van het panel op het form. Het is de bedoeling dat de onderrand tegen het onderste panel blijft en de andere randen tegen de randen van het form.

Klik op `pnlScope` om deze te selecteren. Scrol in de properties-scherm naar `Anchor`. Klik op het pijltje zodat er rechthoeken in beeld komen. Zorg dat alle omcirkelde rechthoeken donkergrijs worden door er op te klikken. (Zie afbeelding)



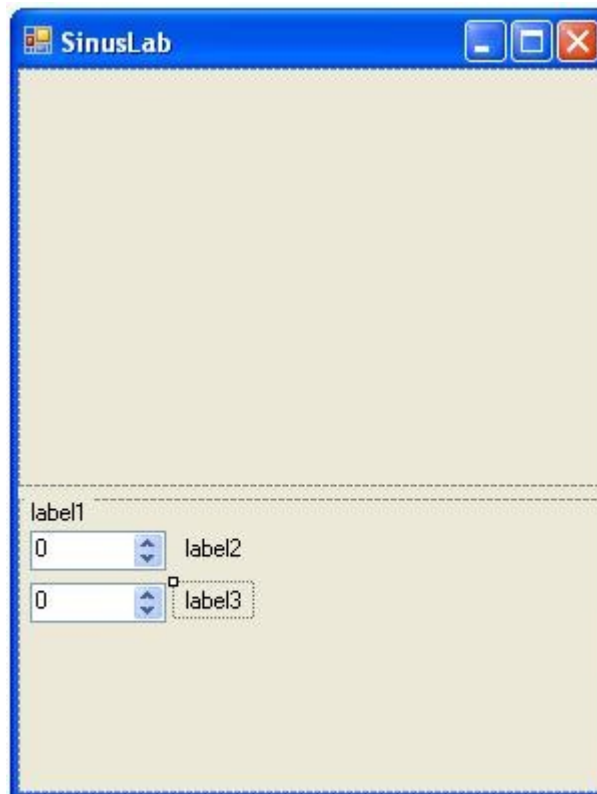
## Invoervelden

Nu kunnen de invoervelden op de onderste panel (pnlBottom) geplaatst worden.

We beginnen met de invoer van frequentie en amplitude van de draaggolf.

Scrol in de toolbox naar 'Common Controls' en klik op 'Label' in klik vervolgens op pnlBottom, zodat een label aan het onderste panel wordt toegevoegd. Verschuif de label naar de linker bovenhoek van pnlBottom.

Onder Common Controls is ook NumericUpDown te vinden plaats hiervan twee onder de label. Plaats achter elke NumericUpDown weer een label. Pas de afmetingen en en posities van de labels en NumericUpDown-controls zodanig aan dat het overeenkomt met onderstaande afbeelding.



Wijzig nu de properties van elk control volgens onderstaand tabel. Zorg dat het juiste control geselecteerd staat (door er op te klikken), alvorens de properties in de Properties Window aan te passen.

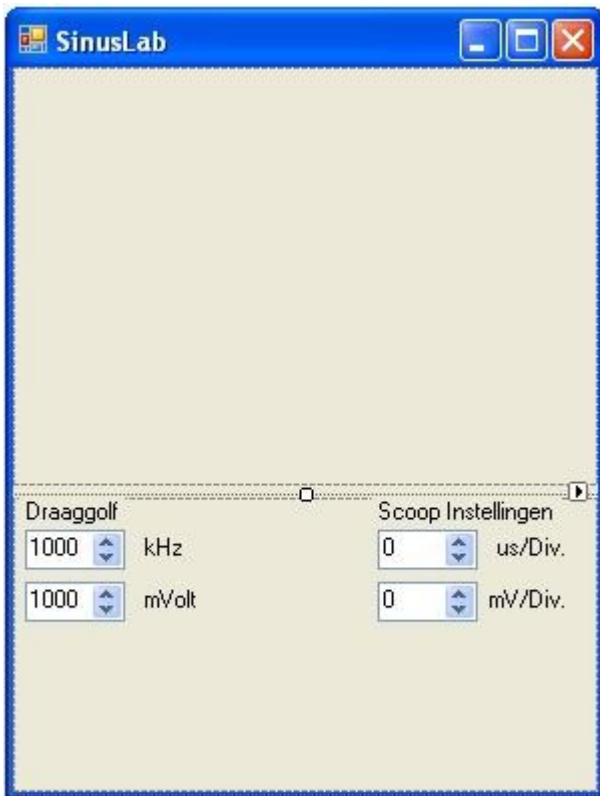
Control	property	waarde
label1	Name	lblDraaggolf
	Text	Draaggolf
label2	Name	lblFrequentie
	Text	kHz
label3	Name	lblAmplitude
	Text	mVolt
numericUpDown1	Name	updfrequentie
	Maximum	10000

	Minimum	1
	Value	1000
numericUpDown2	Name	updAmplitude
	Maximum	10000
	Minimum	1
	Value	1000

De controls zien er nu zo uit:



Op dezelfde wijze gaan we nu de controls voor de scope toevoegen en aanpassen volgens onderstaande afbeelding en tabel.



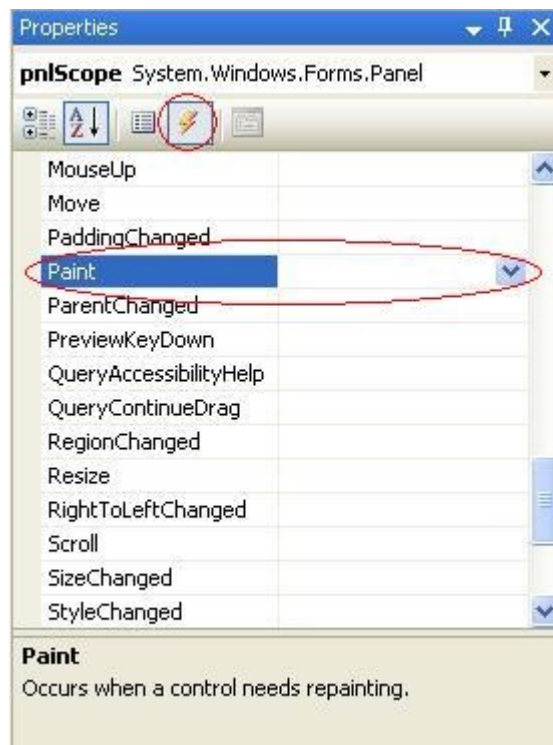
Control	property	waarde
lblScoop	Name	lblScoop
	Text	Scoop Instellingen
lblTimeDiv	Name	lblTimeDiv
	Text	us/Div.
lblVoltDiv	Name	lblVoltDiv
	Text	mV/Div.
updTimeDiv	Name	updTimeDiv
	Maximum	10000
	Minimum	1
	Value	1000
updVoltDiv	Name	updVoltDiv
	Maximum	10000
	Minimum	1
	Value	1000

## Lijnen tekenen

Op de scoop moeten een aantal lijnen zichtbaar zijn die de horizontale en verticale schaal vormen. Panels zijn van zichzelf een vlak met één kleur. We moeten er zelf voor zorgen dat de lijnen getekend worden door een stuk source code te schrijven die wat teken-opdrachten uitvoert. Dat stuk source code moet uitgevoerd worden op het moment dat het panel (pnlScope) zichzelf tekent. Elke control tekent zichzelf op het moment dat het form tevoorschijn komt of een bovenliggend scherm verdwijnt.

In het geval van onze pnlScoop zal dit Panel zijn grijze vlak tekenen. Na dit (her)teken zal het Panel een event geven. Met dit event (gebeurtenis) geeft het Panel aan dat deze getekend wordt. Wij als programmeur kunnen dit event afvangen en in een event handler de source code plaatsen waarvan wij willen dat die op dat moment wordt uitgevoerd.

De events van een control zijn te vinden in de Properties window. Selecteer pnlScoop en klik vervolgens in de Properties window op de bliksemschicht. Nu is een lijst met events te zien. Achter elke event staat een veld waar een event handler kan staan. Deze velden zijn nu nog allemaal leeg.



Zoek in de lijst naar het event met de naam 'Paint'. Dubbelklik op het veld achter Paint. Na wat korte hokuspokus wordt een nieuw scherm geopend met de naam guiSinusLabMain.cs en wat tekst dat er als volgt uit ziet:

```
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;
```

```
namespace SinusLab  
{  
    public partial class frmSinusLabMain : Form
```

```

{
    public frmSinusLabMain()
    {
        InitializeComponent();
    }

    private void pnlScope_Paint(object sender, PaintEventArgs e)
    {

    }
}

```

De `pnlScope_Paint` is de event handler waar we de teken opdrachten aan gaan toevoegen. Om het eenvoudig te houden gaan we eerst één lijn tekenen. Type daarvoor de volgende code in de event handler:

```

private void pnlScope_Paint(object sender, PaintEventArgs e)
{
    Graphics gfx = e.Graphics;
    Pen myPen = new Pen(Color.Red);
    int ZeroLine = pnlScope.Height / 2;
    gfx.DrawLine(myPen, 0, ZeroLine, pnlScope.Width, ZeroLine);
}

```

Klik op het groene pijltje, zodat het programma gecompileerd en gestart wordt. Er moet nu een rode rode horizontale lijn zichtbaar zijn. Dit is de nul-lijn van de oscilloscoop.

De source code werkt als volgt:

De event handler krijgt een variabele `e` mee. Deze variabele is van het type `PaintEventArgs`. `e` bevat `Graphics` van het type `Graphics`. In de eerste regel pakken we deze `Graphics` er uit en stoppen deze in de variabele `gfx`. Deze `graphics` kan namelijk gebruikt worden om lijnen, rechthoeken, cirkel etc. te tekenen. In ons geval om lijnen te tekenen.

Vervolgens hebben we een `Pen` nodig. Zoals de naamgeving al zegt is dit een pen waarmee getekend wordt. Zo'n pen kan verschillende eigenschappen hebben zoals dikte en kleur. In de tweede regel van de event handler wordt `myPen` van het type `Pen` gecreëerd. Hierbij wordt tussen haakjes aangegeven wat de kleur moet zijn.

In de derde regel wordt bepaald waar de lijn getekend moet worden. We willen een nul lijn tekenen die horizontaal loopt en in het midden. Dus op de halve hoogte van het Panel. We delen daarom `pnlScope.Height` door twee en stoppen deze in de variabele `ZeroLine` die op zijn beurt weer van het type `int` is.

We zien dat de source code bestaat uit verschillende variabelen die een naam hebben en van een bepaalde type zijn. Wat de verschillende types inhouden laten we nu nog achterwege, het gaat er om wat we er mee kunnen. Ook zien we dat we met de variabelen bewerkingen kunnen uitvoeren zoals berekeningen.

De vierde regel van de event handler laat uiteindelijk `Graphics gfx` de lijn tekenen. De functie `DrawLine` doet het tekenwerk. Tussen de haakjes geven we aan deze functie het gereedschap en materiaal mee waarmee de klus geklaard moet worden. Als eerste de `Pen myPen` waarmee getekend moet worden en vervolgens waar de lijn moet beginnen en eindigen. De `0` geeft aan dat de lijn helemaal links moet beginnen en de `ZeroLine` die daarop volgt geeft aan dat dit beginpunt op de halve hoogte ligt. De `pnlScope.Width` geeft aan dat de lijn helemaal rechts moet eindigen en tenslotte nogmaals de `ZeroLine` dat aangeeft dat ook het eindpunt op de halve hoogte ligt.

Nu gaan we nog meer horizontale lijnen tekenen. We doen dit met wederom met `gfx.DrawLine` en

wel met de volgende regel:

```
gfx.DrawLine(myPen, 0, y, pnlScope.Width, y);
```

Deze regel gaan we meerdere keren uitvoeren in een lus, want we willen meerdere lijnen hebben. Deze lus ziet er als volgt uit:

```
for (int y = ZeroLine - 20; y > 0; y = y - 20)
{
    gfx.DrawLine(myPen, 0, y, pnlScope.Width, y);
}
```

Deze for loop voert DrawLine een aantal keren uit, beginnend met een waarde y gelijk aan ZeroLine-20 en vervolgens met een y die telkens 20 kleiner wordt. Deze herhaling van DrawLine gaat net zo vaak door tot y bij nul aangekomen is.

Voordat we deze loop uitvoeren gaan we eerst de pen van een andere kleur inktpatroon voorzien met de code:

```
myPen.Color = Color.Black;
```

Voeg nu de bovenstaande regel en de lus toe aan de event handler. De eventhandler ziet er nu als volgt uit:

```
private void pnlScope_Paint(object sender, PaintEventArgs e)
{
    Graphics gfx = e.Graphics;
    Pen myPen = new Pen(Color.Red);
    int ZeroLine = pnlScope.Height / 2;
    gfx.DrawLine(myPen, 0, ZeroLine, pnlScope.Width, ZeroLine);
    myPen.Color = Color.Black;
    for (int y = ZeroLine - 20; y > 0; y = y - 20)
    {
        gfx.DrawLine(myPen, 0, y, pnlScope.Width, y);
    }
}
```

Start het programma. Als het goed is worden boven de rode lijn zwarte lijnen getekend.

Nu willen we ook lijnen onder de rode lijn. Hiervoor maken we een vergelijkbare lus:

```
for (int y = ZeroLine + 20; y < pnlScope.Height; y = y + 20)
```

Een oscilloscoop heeft ook verticale rasterlijnen. Deze gaan we met één lus tekenen:

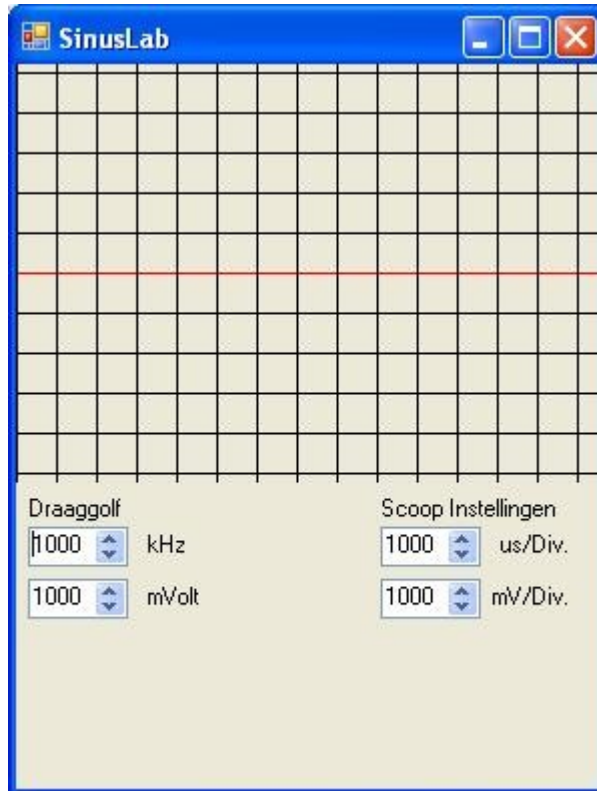
```
for (int x = 0; x < pnlScope.Width; x = x + 20)
{
    gfx.DrawLine(myPen, x, 0, x, pnlScope.Height);
}
```

Voeg de laatste lussen toe aan de event handler. De code ziet er dan als volgt uit:

```
private void pnlScope_Paint(object sender, PaintEventArgs e)
{
    Graphics gfx = e.Graphics;
    Pen myPen = new Pen(Color.Red);
    int ZeroLine = pnlScope.Height / 2;
    gfx.DrawLine(myPen, 0, ZeroLine, pnlScope.Width, ZeroLine);
    myPen.Color = Color.Black;
    for (int y = ZeroLine - 20; y > 0; y = y - 20)
    {
        gfx.DrawLine(myPen, 0, y, pnlScope.Width, y);
    }
    for (int y = ZeroLine + 20; y < pnlScope.Height; y = y + 20)
    {
        gfx.DrawLine(myPen, 0, y, pnlScope.Width, y);
    }
    for (int x = 0; x < pnlScope.Width; x = x + 20)
    {
        gfx.DrawLine(myPen, x, 0, x, pnlScope.Height);
    }
}
```

}  
}

Start het programma en het volgende is nu zichtbaar:



TO BE CONTINUED