HOOFDSTUK **26**

Universal Apps bouwen

Inleiding

In dit extra hoofdstuk gaan we je apps leren bouwen met C#. Microsoft heeft al verschillende pogingen gedaan om ontwikkelaars de mogelijkheid te geven apps te ontwikkelen. Eerst was er Windows Phone 7, met daarop een versie van het Silverlight-platform. Dit zou je kunnen beschouwen als een eerste probeersel. Helaas liepen ontwikkelaars en consumenten er niet echt warm voor. Daarna kwam Microsoft met Windows 8 en Windows 8.1 voor desktopcomputers en tablets, en Windows Phone 8 voor smartphones. Hoe je apps voor dit platform ontwikkelt, kun je lezen in een eerdere versie van dit hoofdstuk (die nog steeds is terug te vinden op de website die bij het boek hoort). Windows Phone 8 was redelijk succesvol, maar Windows 8 was dat zeker niet. Ontwikkelaars die een bepaalde toepassing beschikbaar wilden maken op beide systemen, moesten *twee* apps schrijven: eentje voor de smartphone en eentje voor de tablet. Daarbij waren sommige technieken en bibliotheken niet geünificeerd en moest je voor sommige (identieke) taken andere code schrijven in de twee apps.

Ondertussen heeft Microsoft met de komst van Windows 10 het principe van de Universal App gelanceerd. Dat principe is eenvoudig. Windows 10 is een besturingssysteem waarvan de kern (de basis) voor elk type toestel hetzelfde is. Een laptop die Windows 10 draait of een Lumia 950: het is in essentie *hetzelfde* systeem. Daarnaast kan Windows 10 ook draaien op een Xbox-spelconsole, een Raspberry Pi-experimenteerbordje[1] of zelfs een Augmented Reality-bril (de Microsoft HoloLens[2]). Het grote voordeel van deze aanpak is dat je een app maar één keer hoeft te schrijven en maar één keer in de Store hoeft te zetten. In principe werkt je app dan op alle toestellen die zijn uitgerust met Windows 10. De apps draaien dus 'universeel' op die toestellen, vandaar de naam Universal Apps. Je app zal er natuurlijk anders uitzien naargelang zij gebruikt wordt op een telefoon, een grote desktopcomputer of een futuristische bril. Daarvoor zijn er technieken ontwikkeld die de app adapteren naar de verschillende uitvoeringsomgevingen. Toch is het belangrijk om te beseffen dat het telkens om een en dezelfde app gaat.

Voordat we van start kunnen gaan, moeten we enkele tools installeren. We geven eerst antwoord op de vraag wat je naast Visual Studio allemaal nodig hebt om apps te ontwikkelen en te debuggen. Vervolgens tonen we hoe je een eenvoudig programma opbouwt en uitvoert. We leggen uit hoe je dit programma zowel op een desktopcomputer als op een Windows Phone kunt draaien. Ten slotte bouwen we app-versies van enkele voorbeelden die in het boek de revue zijn gepasseerd. We leggen hierbij de nadruk op een goed (klassen)ontwerp en realiseren een aangepast uiterlijk (GUI) voor Windows en Windows Phone. Hopelijk heb je na het lezen van dit hoofdstuk zin in meer, want we kunnen hier niet ingaan op alle toeters en bellen. Daarom geven we aan het einde van het hoofdstuk nog enkele tips om zelf verder aan de slag te gaan.

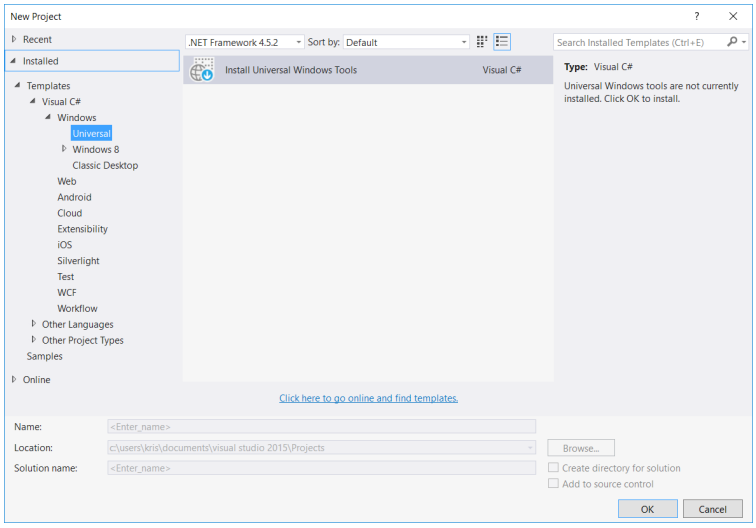
De tools die je nodig hebt

Je kunt gratis aan de slag met Visual Studio Community Edition. Dit is dezelfde versie als de versie die in de overige hoofdstukken gebruikt is. Om de voorbeelden uit

1 https://dev.windows.com/en-us/iot
2 https://www.microsoft.com/microsoft-hololens

dit hoofdstuk uit te proberen, dien je de allerlaatste versie van Visual Studio Community Edition te installeren, inclusief de laatste updates (op het moment van schrijven VS 2015 Update 1). De projecten die hier beschreven worden, kun je helaas niet uitvoeren met oudere versies van Visual Studio.

Tijdens het installeren van Visual Studio kun je aanvinken welke mogelijkheden je allemaal tot je beschikking wilt hebben. Mogelijk is app-ontwikkeling uitgeschakeld, maar dit kun je achteraf gemakkelijk toevoegen. Controleer eerst of je de juiste project templates hebt. Hiervoor ga je naar **File | New Project** en navigeer je vervolgens naar **Templates | Visual C# | Windows | Universal**. Staat daar "Install Universal Windows Tools", dan heb je nog een aantal extra's nodig om van start te kunnen gaan. Klik op de gegeven link en doorloop de stappen. Je zult Visual Studio moeten afsluiten en mogelijk de computer moeten herstarten om de installatie te voltooien.

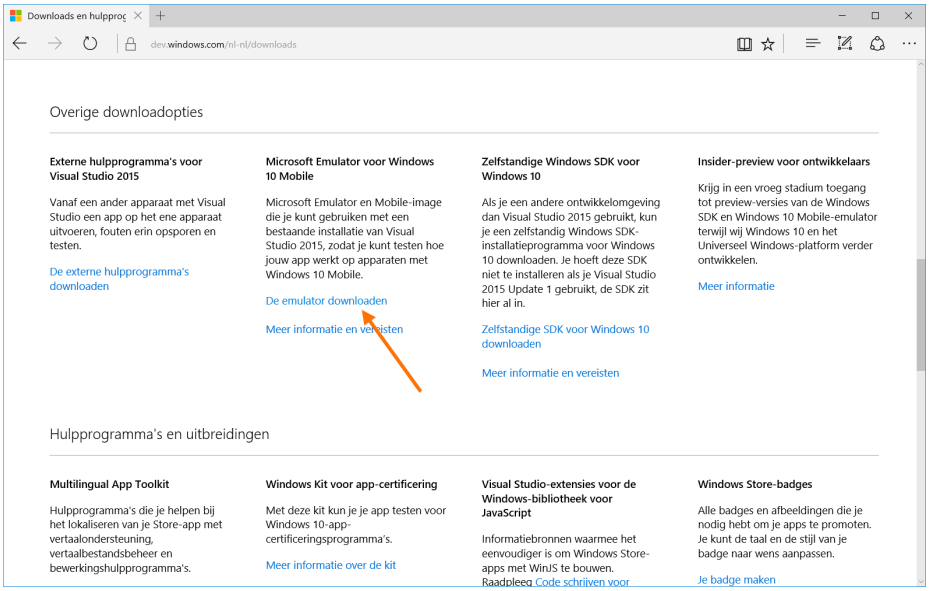


Figuur 26.1 Installeer de universal project templates

Nadat je deze templates hebt geïnstalleerd, is het belangrijk om de laatste updates van de emulators te downloaden voor Windows Phone. Een *emulator* is een virtueel apparaat dat op je ontwikkelmachine draait en waarmee je je programma's kunt testen. Ook al heb je een echt toestel, je wilt natuurlijk graag weten of alles een beetje werkt voordat je het op het echte toestel uitprobeert. De bouw- en testcyclus is vaak ook korter met een emulator. Daarnaast is één toestel er ook maar één. Er zijn meerdere emulators beschikbaar voor verschillende typen toestellen. Krijg je echt interesse in het professioneel of hobbymatig schrijven van apps, dan zou je zeker kunnen overwegen een toestel met Windows Phone aan te schaffen. Deze toestellen zijn helemaal niet zo duur. Een basistoestel (bijvoorbeeld de Lumia 550) vind je al voor rond de 130 euro. Zo'n toestel is perfect voor app-ontwikkeling. Let er wel op dat je een recent toestel aanschaft met het Windows Phone 10-besturingssysteem, en niet een toestel met de verouderde versie 8.

Ga naar de ontwikkelaarswebsite van Microsoft (http://dev.windows.com) en zoek de link Downloads. Daar vind je een rechtstreekse verwijzing naar de Windows 10-emu-

lator, waarmee je verschillende toestellen kunt simuleren. Figuur 26.2 toont deze website.



Figuur 26.2 Installeer de emulators

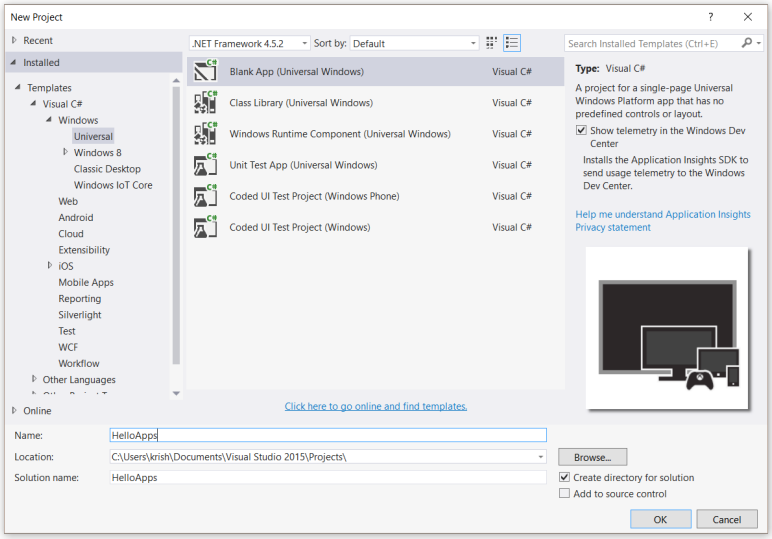
Tijdens het installeren van de tools is het mogelijk dat het installatieprogramma voortijdig afbreekt. Zorg ervoor dat het Windows 10-besturingssysteem dat op je ontwikkelmachine staat een Professional of Enterprise Edition is. De Home Edition is ontoereikend. Deze versie bevat namelijk geen Hyper-V-optie. Hyper-V is een techniek voor het draaien van emulators boven op je besturingssysteem.

Via dezelfde site kun je ook inloggen met een Microsoft-account. Dit is een gratis account waarmee je eveneens toegang hebt tot bijvoorbeeld OneDrive (cloudopslag) en Outlook.com (e-mail). Misschien heb je dus al een dergelijk account. Zo niet, dan kun je het best een nieuw account aanmaken. Het account komt van pas als je later je apps in de Store wilt plaatsen en verkopen. (Dit is trouwens gratis voor studenten met het DreamSpark-programma.) Een dergelijke registratie is *niet* nodig als je de apps alleen op je eigen toestel wilt uitproberen.

Een eerste programma: HelloApps

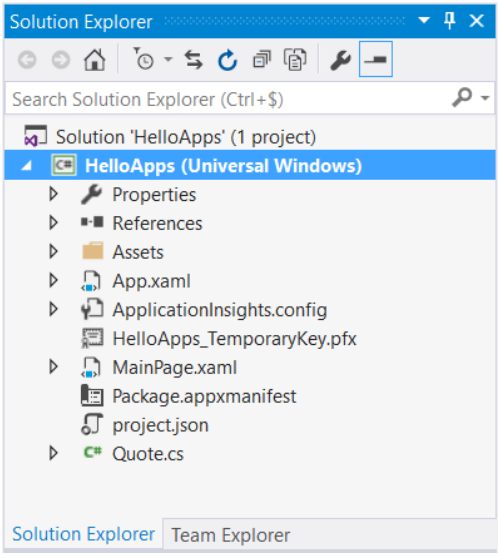
Zoals zo vaak begin je het best met een eenvoudig programma. Dit helpt je om de tools te verkennen en zo zie je ook of alles naar behoren geïnstalleerd en operationeel is. De eerste stap ken je al: het beginnen van een nieuw project. Start hiervoor Visual Studio en begin een nieuw project (**File | New Project**). Kies nu onder de verschillende templates voor een *Universal Apps*-project, namelijk een *Blank App*, en noem het "HelloApps", zoals aangeduid in figuur 26.3.

Let erop dat het vinkje "Create directory for solution" aangevinkt is. Dit zorgt ervoor dat het project netjes in één overkoepelende map terechtkomt. In meer geavanceerde toepassingen wordt vaak met meerdere projecten gewerkt, en die kun je dan



Figuur 26.3 Het project HelloApps

netjes in deze overkoepelende map onderbrengen. Werken met meerdere projecten behandelen we niet in dit (inleidende) hoofdstuk, maar het is wel een goede gewoonte om aan te houden.



Figuur 26.4 De solution voor het programma HelloApps

Voeg nu de klasse `Quote` toe aan het project zoals te zien in figuur 26.4. Dit doe je op de bekende manier: rechtsklikken op het project, kiezen voor **Add > New Item** en kiezen voor 'Class'.

De klasse zelf definieert een methode `SayHello`, die een `string` retourneert:

Solution: HelloApps | Project: HelloApps | Bestand: Quote.cs

```
public class Quote
{
    public string SayHello()
    {
        return "Hello there, I'm running as an app!";
    }
}
```

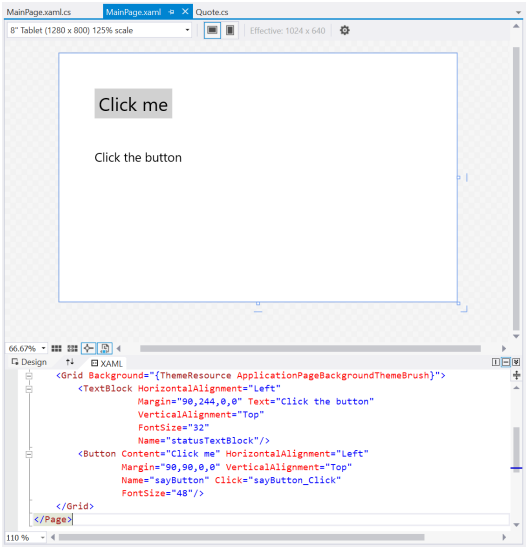
Nu gaan we de gebruikersinterfaces vervolledigen. We maken een versie die er goed zal uitzien op een tablet of desktopcomputer. Open in het project **HelloApps** het bestand `MainPage.xaml` en completeer de interface met volgende code:

Solution: HelloApps | Project: HelloApps | Bestand: MainPage.xaml

```
<Page ...>
  <Grid ...>
    <TextBlock HorizontalAlignment="Left"
               Margin="90,244,0,0" Text="Click the button"
               VerticalAlignment="Top"
               FontSize="32"
               Name="statusTextBlock"/>
    <Button Content="Click me" HorizontalAlignment="Left"
            Margin="90,90,0,0" VerticalAlignment="Top"
            Name="sayButton" Click="sayButton_Click"
            FontSize="48"/>
  </Grid>
</Page>
```

Deze code lijkt heel erg op WPF en je zult onmiddellijk begrijpen wat hier bedoeld wordt. Een Windows-app draait steeds in de context van een 'pagina', vandaar dat het rootelement nu geen `Window`-element is (zoals in WPF) maar een `Page`-element. Daaronder heb je een `Grid`-element dat je reeds kent, en daaronder kun je de controls definiëren. De gangbaarste controls ken je reeds van WPF. Die kun je dus zonder meer toepassen. Ze hebben echter een andere look gekregen die meer is toegespitst op het gebruik met een tablet en touchbediening.

In figuur 26.5 zie je hoe Visual Studio dit bestand weergeeft. In plaats van een WPF-venster zie je nu een rechthoek die het scherm van een tablet voorstelt. Daarin heb je de gebruikersinterface van het programma: een `Button` en een `TextBlock`. Windows-toestellen zijn er in alle vormen en maten. Daarom vind je aan de bovenkant van het ontwerpvenster een selectieruimte, waar je als programmeur kunt experimenteren met verschillende beeldscherminstellingen. Je kunt bijvoorbeeld de resolutie en de grootte van de tablet veranderen om te kijken welk effect dit heeft op het uiterlijk van je app. Je kunt ook testen hoe je app eruitziet op een telefoon met verschillende groottes, in verschillende resoluties en in horizontale of verticale stand.



Figuur 26.5 Het ontwerp van de interface

TESTVRAAG

26.1 *Vergelijk de volgende resoluties:*

- 8" tablet (1280 x 800), 125% scale. Dit betekent een grootte van 8 inch, met in horizontale richting 1280 pixels en in verticale richting 800 pixels, waarbij de effectieve afmetingen van de elementen 125% vergroot zijn.
- 5" Phone (1920 x 1080), 300% scale. Een grootte van 5 inch op een telefoon (standaard in portretweergave) met een resolutie van 1920 bij 1080 pixels (dit noemt men een high-definition resolutie). De elementen zijn 300% vergroot.
- 55" Surface Hub (1920 x 1080), 100% scale.

Kun je nu het uiterlijk van de app in het ontwerpvenster verklaren met deze verschillende instellingen?

Wanneer er op de knop geklikt wordt, willen we een melding laten verschijnen. Dit doen we op dezelfde manier als in WPF, namelijk door het `Click`-event af te handelen met een methode:

Solution: HelloApps | Project: HelloApps | Bestand: MainWindow.xaml.cs

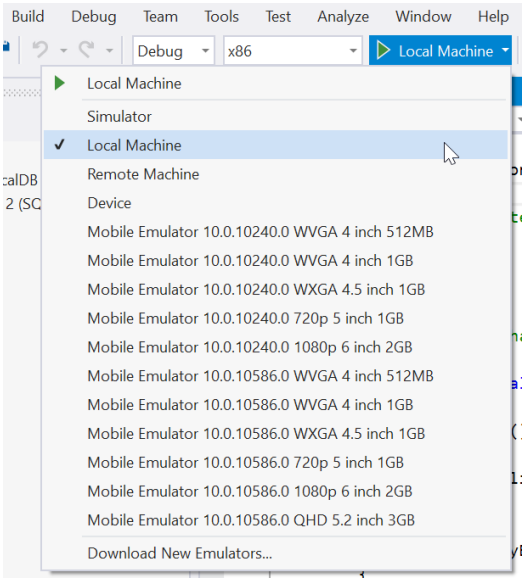
```
private void sayButton_Click(object sender, RoutedEventArgs e)
{
    Quote quote = new Quote();
    statusTextBlock.Text = quote.SayHello();
}
```

Programma's uitvoeren

Als alle programmacode is ingegeven, is het tijd om het programma te laten lopen. Dit gebeurt opnieuw op dezelfde manier als bij een WPF-applicatie, namelijk via de run-knop of de F5-toets. Afhankelijk van de instellingen zal de applicatie op je eigen systeem of in een emulator opgestart worden. Het is ook mogelijk om het programma via een echt toestel te testen. We geven een overzicht van alle mogelijkheden.

Op de lokale machine

Zorg ervoor dat de optie "Local Machine" geselecteerd is (zie figuur 26.6. Je programma zal dan uitgevoerd worden als app op je eigen machine. Dit werkt alleen als je Windows 10 geïnstalleerd hebt. Deze werkwijze lijkt op de werkwijze die je gewend was met WPF-programma's. Op het eerste gezicht is er dan ook weinig verschil met WPF-programma's. Weet echter dat deze apps ook kunnen draaien op een smartphone, iets wat helemaal niet mogelijk is met WPF!



Figuur 26.6 Het programma starten

Deze manier van werken is prima, maar niet altijd representatief voor alle mogelijke situaties.

Op een emulator

Verander daarom de opstartwijze (zie figuur 26.6) en run je programma opnieuw via een van de vele emulators voor tablets en/of smartphones. Kies bijvoorbeeld voor "Mobile Emulator 10.0.1024.0 WVGA 4 inch 512 MB". Dit stelt een (virtueel) toestel voor met een scherm van 4 inch groot en met 512 MB RAM geheugen. Je hebt een

ruime keuze aan toestellen, met een waaier aan schermgroottes en geheugen. Maar let op: hoe krachtiger het virtuele toestel dat je gebruikt, hoe zwaarder je ontwikkelmachine wordt belast, waardoor die trager zal reageren. Het is dus verstandig om op een minder krachtige machine geen al te zware emulators op te starten.

Je zult zien dat er vervolgens een soort virtueel toestel wordt opgestart waarin je app draait. De eerste keer kan dit een tijdje duren. Daarna kun je je programma volledig testen en debuggen. Een voorbeeld van de app in de emulator zie je in figuur 26.7.

Figuur 26.7 Het programma in de emulator

Deze emulator blijft draaien, ook als je het debuggen beëindigt of als je de app sluit. Je kunt de emulator eenvoudig afsluiten door rechtsboven op het kruisje te klikken (*Close*), maar het is gebruikelijk om de emulator tijdens het ontwikkelen gewoon aan te laten staan.

Je zou een melding kunnen krijgen dat de emulator niet kan worden opgestart omdat je niet over de benodigde permissies beschikt. Klik deze boodschap gewoon weg door op "Retry" te klikken. De app zou dan alsnog moeten starten.

Het dient gezegd dat het draaien van de Windows-emulator niet altijd van een leien dakje loopt. We sommen hier de problemen op die zich het vaakst voordoen, met een mogelijke oplossing.

1. Sommige antivirusprogramma's verhinderen een verbinding tussen Visual Studio en de emulator. Je merkt dit doordat de emulator opstart, maar de melding "Loading..." zichtbaar blijft zonder dat de app start. Schakel in dat geval je antivirusprogramma's even uit terwijl je aan het programmeren bent. Niet vergeten ze nadien weer in te schakelen!
2. Als je het vervelend vindt om telkens weer op "Retry" te moeten klikken om de emulator te starten, run Visual Studio dan zelf als Administrator. Dit doe je bijvoorbeeld door rechts te klikken op de snelkoppeling naar het Visual Studio-programma en te kiezen voor "Run as Administrator".

3. De emulator gebruikt een virtualisatietechniek die Hyper-V heet. De technische details zijn niet zo belangrijk. Zorg er wel voor dat Hyper-V is ingeschakeld. Dit moet je doen op twee niveaus:
   – Ten eerste moet je Hyper-V aanzetten in de BIOS. Start de computer opnieuw op en houd een functietoets ingedrukt, zodat het besturingssysteem niet opstart en je wijzigingen kunt doorvoeren. De precieze stappen kunnen we je niet meegeven, want die zijn op elke computer anders. Je zult het dus even moeten uitzoeken of navragen. Meestal is het de <F2>- of <F10>-toets of een speciale toets als <Del>. Vervolgens zoek je naar de volgende termen, afhankelijk van de fabrikant van de processor (Intel of AMD). Deze termen zet je AAN:

Eigenschap	AMD-instelling	Intel-term
SLAT (Second Level Address Translation)	NP (Nested Page Tables) RVI (Rapid Virtualization Indexing)	EPT (Extended Page Tables)
Hardware-assisted virtualization	SVM (AMD support for hardware-assisted virtualization)	VMX (Intel support for hardware-assisted virtualization)
Data Execution Prevention (DEP)	NX (No Execute)	XD (Execute Disable)

   – Vervolgens zoek je in de BIOS naar de volgende settings. Deze settings schakel je UIT:
   Intel VT-d
   Trusted Execution
   – Merk op dat de bovenstaande instellingen niet steeds op alle machines voorkomen. Zolang je maar een van bovenstaande instellingen hebt, weet je dat op je machine Hyper-V mogelijk is.
   – Op oudere machines is Hyper-V niet mogelijk. In dat geval zul je via een echt toestel moeten testen. We leggen verderop uit hoe dat moet.
   – Als Hyper-V in de BIOS aanstaat, dien je nog een laatste stap te doen in het besturingssysteem. Ga naar het "Configuratiescherm", selecteer "Programma's" en vervolgens "Windows onderdelen in- of uitschakelen". Zoek naar Hyper-V en zorg dat alle checkboxen aangevinkt staan (zie figuur 26.8).

Dit alles lijkt erg ingewikkeld, maar laten we je geruststellen: op de meeste computers dien je enkel deze laatste stap te doen (zie figuur 26.8). Probeer dit dus eerst en kijk of je de emulator kunt starten. Als dat niet lukt, kun je altijd nog proberen om de BIOS-instellingen aan te passen. Mocht het onverhoopt toch niet lukken, dan verwijzen we je naar het volgende artikel van Microsoft voor extra tips:

http://msdn.microsoft.com/library/windows/apps/jj863509(v=vs.105).aspx

De informatie achter deze link gaat over de Windows Phone 8-emulator, maar is nog steeds relevant voor Windows 10-emulators, omdat die ook op het Hyper-V-principe gebaseerd zijn.