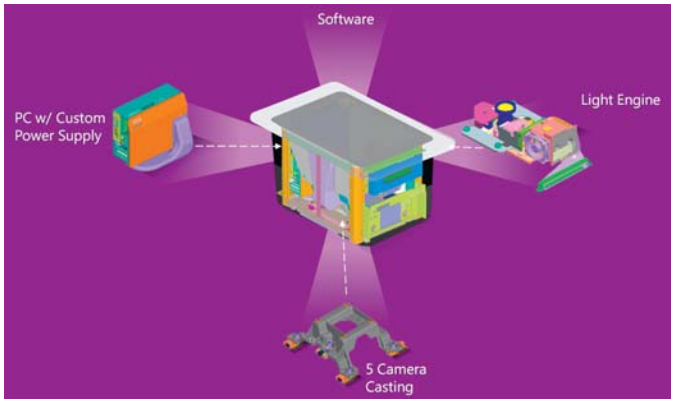
Microsoft Surface

Een nieuwe ervaring in User eXperience!

Bas Zweeris, Joost Verhoog en Marijn Moerman.

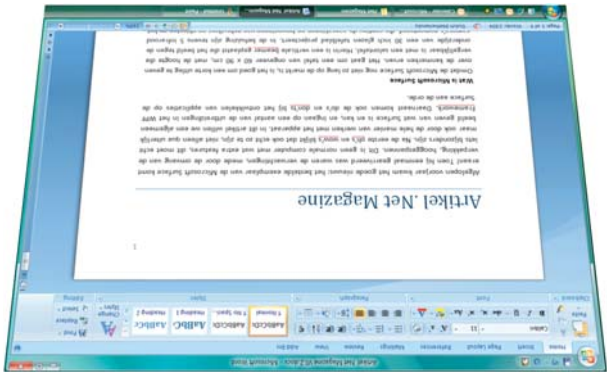
Dit voorjaar kwam het goede nieuws: het bestelde exemplaar van de Microsoft Surface kwam eraan! Toen hij eenmaal gearriveerd was, waren de verwachtingen hooggespannen. Dit is geen normale computer met wat extra features, dit moet echt iets bijzonders zijn. Na de eerste 'oh's' en 'wow's' bleek dat ook echt zo te zijn, niet alleen qua uiterlijk maar ook door de hele manier van werken met het apparaat.

Omdat de Microsoft Surface nog niet zo lang op de markt is, is het goed om een korte uitleg te geven over de kenmerken ervan. Het gaat om een tafel van ongeveer 60 x 90 cm, met de hoogte die vergelijkbaar is met een salontafel. Hierin is een verticale beamer geplaatst die het beeld tegen de onderzijde van een 30 inch glazen tafelblad projecteert. In de behuizing zijn tevens vijf infrarood camera's gemonteerd, die continu de aanrakingen en bewegingen van gebruikers en objecten op het tafelblad registreren. Surface bevat een standaard desktop PC met Windows Vista SP1. Bovenop Vista draait de Surface shell die alle operating system UI wegfiltert. Onderdeel van de Surface Shell is een aparte launcher, waarop de Surface-applicaties draaien.



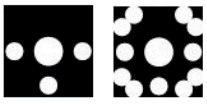
FIGUUR 1: DE MICROSOFT SURFACE UNIT.

Een tafel kenmerkt zich door de mogelijkheid om er met meerdere mensen aan te zitten aan verschillende kanten. Dit heeft dan ook grote invloed op de manier waarop het Interaction Design voor Surface-applicaties moet worden vormgegeven. Stel je voor dat je Paint, of elke andere willekeurige desktop applicatie, zou laden op de Surface, dan zou het programma voor de gebruiker aan de andere zijde niet te bedienen zijn. Zo oogt het bijvoorbeeld ook heel raar als de startknop ineens op de plaats van de sluitknop zit, en vice versa. Het één op één overzetten van standaard Windows- of webapplicaties is wel mogelijk maar wordt daarom sterk afgeraden. Het kan wel, maar dat zou hetzelfde zijn als een tv kopen om alleen maar muziek mee te luisteren of je kamer te verlichten.



FIGUUR 2: STANDAARDAPPLICATIE NIET GESCHIKT VOOR SURFACE.

Zoals eerder opgemerkt kan Surface door het gebruik van infraroodcamera's naast je vingers ook tal van andere objecten detecteren, zoals je arm, een velletje papier, geld of een beker. Er is echter ook een manier om een object te 'herkennen'. Dit gebeurt door middel van Tags. Tags zijn een soort barcodes, die op basis van hun patroon geïdentificeerd kunnen worden door de Surface. Een applicatie kan vervolgens zo worden geprogrammeerd dat er een bepaalde actie wordt uitgevoerd als er een object met een bepaalde Tag wordt geplaatst. Plak bijvoorbeeld een Tag op een mobiele telefoon en laat de Surface bij het herkennen van deze tag de foto's van de betreffende telefoon downloaden via Bluetooth.



FIGUUR 3: VOORBEELDEN VAN TAGS.

De Surface wordt geleverd met verschillende demonstratieprogramma's die elk één of meer van de unieke kenmerken van de Surface benutten, maar deze nog niet optimaal combineren. Het is dus noodzakelijk om zelf applicaties te bouwen voor de toepas-

sing en ruimte waarmee de features van de Surface het beste tot hun recht komen.

Het doel

Zoals eerder beschreven trok de komst van het apparaat nogal wat aandacht en waren ook de verwachtingen rondom Surface hooggespannen. Zo kwamen er tal van collega's vertellen dat ze nog wel een bestaande desktopapplicatie hadden die ze graag wilden laten overzetten naar de Surface Unit. Ook vonden de meesten dat het apparaat direct in de lobby moest komen te staan, zodat iedereen een eerste glimp van Surface op kon doen. Hieruit bleek dat de verwachtingen wellicht iets te hoog waren. Omdat er voor Surface op het moment van ontvangen ook nog maar weinig applicaties beschikbaar waren, leek het ons een goed idee eerst een demo te ontwikkelen die alle features van het apparaat zou belichten. Het Surface-team besloot daartoe om de business case uit een eerder gemaakte Silverlightdemo van een financiële planner te verwerken in een Surface-applicatie. Deze demoapplicatie geeft een indicatie over de realisatiemogelijkheden van wensen die je hebt. Denk bijvoorbeeld aan een nieuw huis, auto, een huwelijk, etcetera.

In de volgende paragrafen gaan we in op een aantal functionele en technische uitdagingen waar we tegenaan liepen bij het ontwikkelen van deze demo, startend met een functionele uitleg van de Silverlight demo.

De user interface moet natuurlijk wel erg intuïtief zijn en de manier van gebruiken moet direct duidelijk zijn.

Functionaliteit demo

In de Silverlightdemo dient de gebruiker een wizard te doorlopen om tot een haalbaarheidsindicatie te komen. Zo voert de gebruiker zijn gegevens in en die van een eventuele partner en kinderen, het profiel. Daarna sleept hij de wensen van de gezinsleden op een tijdlijn. Daarbij dienen per wens ook aanvullende gegevens te worden ingevoerd, zoals de kosten van de wens. Daaropvolgend worden de wensen geprioriteerd en moeten diverse gegevens worden ingevoerd betreffende de inkomsten, uitgaven, bezittingen en schulden. Op basis van de bestedingsruimte die hier uit voortvloeit wordt tot slot de haalbaarheid van de wensen bepaald en getoond.

User interface

In de Silverlightdemo worden meerdere schermen en stappen doorlopen om tot een advies te komen. In een multi-user applicatie, waar ook nog eens fysieke objecten op het scherm kunnen staan, kan het echter vreemd overkomen als er van scherm wordt veranderd. Een eerste uitgangspunt was daarom dat de Surfaceapplicatie uit één hoofdscherm zou bestaan waarin alle activiteiten kunnen worden uitgevoerd. Er is ook veel gebrainstormd over de manier waarop de applicatie vanuit 360 graden te bekijken en te bedienen zou kunnen zijn. Niets leek daar beter geschikt voor dan een ronde user interface met zoveel mogelijk ronde objecten. Maar niet alles leent zich voor een ronde user interface, zoals tekst, een listbox of plaatjes. Een goede user interface voor Surface kenmerkt zich daarom door het gebruik van ronde objecten die bovendien draaibaar en verplaatsbaar zijn.

Objectherkenning

Een ander uitgangspunt voor de demo was dat we zo goed mogelijk gebruik zouden maken van objectherkenning. Daartoe zijn met behulp van Lego®-stenen een aantal wensobjecten gebouwd in de vorm van onder andere een auto, een huisje en een kinderbedje, met ieder een eigen Tag. Bij het plaatsen van een wensobject verschijnt vervolgens een bijbehorende visualisatie en eventueel een nieuw stukje UI. Een risico hierbij is dat het snel een rommelige boel wordt op de tafel. Het aantal objecten is daarom geminimaliseerd door een wensobject symbool te laten staan voor een bepaalde categorie aan wensen. Zo wordt de gebruiker bij het plaatsen van het huis bijvoorbeeld gevraagd of het gaat om een nieuw huis, een tweede huis of een verbouwing.

FIGUUR 4: DE SURFACE-TAFEL VAN MICROSOFT.

Invoer van data

Een van de dingen waar we tegenaan liepen was het invoeren van data. Dit gaat niet meer door middel van een toetsenbord of muis en moest dus op een andere manier worden uitgevoerd. Het plaatsen van een object is al invoer op zich en ook de plek van het object op de tafel kan bijvoorbeeld een bepaalde waarde voorstellen. Dit is vervolgens uitgebreid door bij het plaatsen van een object naar de bijbehorende gegevens te vragen zoals de maandelijkse inkomsten en uitgaven of de kosten van een wens. Invoer geschiedt hierbij door middel van ListBoxen, Sliders, ComboBoxen. Maar de standaard rechte vorm van de slider bleek maar van beperkt nut. In ons geval willen we bijvoorbeeld de prijs aangeven van de auto die op tafel staat. Omdat een slider niet zijn eigen waarde weergeeft, plaatsten we er een label met deze waarde onder. Maar toen bleek dat je dit label soms met je hand bedekt als je de slider bedient, met een muis had je dit probleem niet. De oplossing die we hiervoor bedachten was een ronde slider, zoals de volumeknop op een radio. Deze zit niet standaard in het SDK, maar is relatief eenvoudig zelf gemaakt.

Intuïtief

Een belangrijk aspect waar een Surfaceapplicatie aan zou moeten voldoen is dat de user interface erg intuïtief moet zijn. Met andere woorden, de gebruiker zou zonder uitleg bij het eerste gebruik al moeten snappen wat de applicatie ongeveer doet en hoe deze te bedienen is. Dat lijkt simpel en voor de hand liggend, maar dat is het absoluut niet. Men is namelijk nog helemaal niet gewend om met een Multi-touch apparaat te werken. Sterker nog, vanaf kleins af aan hebben we eigenlijk allemaal geleerd ergens met onze vin-

gers vanaf te moeten blijven. Bedenk dus maar eens iets waardoor een gebruiker bijvoorbeeld snapt dat hij een fysiek object op de tafel kan zetten en dat de cirkel die daarbij verschijnt eigenlijk een ronde slider is. Het is dan ook niet gelukt de eerste demo volledig intuïtief te maken, want er bleef enige uitleg nodig.

Toegevoegde waarde

Ondanks de benodigde uitleg heeft de Surfacedemo zeker een toegevoegde waarde ten opzichte van de Silverlightvariant. Dit zit met name in het feit dat alle objecten die op de tafel liggen elkaar beïnvloeden. Zo wordt bij het plaatsen van bijvoorbeeld een autowens door middel van een groene of rode kleur direct weergeven of de wens haalbaar is of niet. Dit wordt afgeleid uit de opgegeven inkomsten en uitgaven. Als er vervolgens een ander wensobject geplaatst wordt, bijvoorbeeld een kind, blijkt de auto niet meer haalbaar en wordt deze automatisch rood. Dit is functionaliteit die niet in de Silverlightdemo werd ondersteund, omdat hierbij een verdeling was over verschillende schermen. Daarnaast werkt het gebruik van fysieke objecten zeer drempelverlagend en wordt een financieel plansysteem ineens erg leuk om mee te werken.

Technische uitdagingen

Surfaceapplicaties worden doorgaans geschreven in WPF. Als je de Surface SDK hebt geïnstalleerd, krijg je een aantal Surface project templates in Visual Studio. Voor het testen van je applicatie is er bovendien een Simulator. Dit is een virtuele Surface unit op je desktop waarmee je één of meerdere muizen kunt gebruiken om je multi-touch applicaties te testen. De bediening is echter niet intuïtief en staat ver van de werkelijkheid. Voor de ontwikkeling van Surfaceapplicaties is dus zeker ook een Surface unit nodig.
De controls die bij de SDK worden meegeleverd zijn voor een groot deel vergelijkbaar met bekende controls zoals labels, buttons en sliders. Deze controls zijn echter aangepast, zodat zij kunnen omgaan met de events die op een Surface voorkomen, zoals ContactDown voor aanraken en ContactUp voor loslaten.
Zoals eerder opgemerkt hebben we zelf een ronde slider gemaakt om op te kunnen geven hoeveel een wens kost. Deze slider moet natuurlijk pas verschijnen onder een wens als de betreffende wens op tafel wordt gezet. Daarnaast moet deze bij het wensobject in de buurt blijven als deze wordt verplaatst. In de Surface SDK zit een control die dit mogelijk maakt: de TagVisualizer.

TagVisualizer

Iedere TagVisualizer heeft een lege collectie van TagDefinitions. Door een definitie aan de TagVisualizer toe te voegen maak je een koppeling tussen een bepaalde Tag en een TagVisualization. Een custom TagVisualization is een standaard UserControl die erft van de base class TagVisualization. De TagVisualizer zorgt er vervolgens voor dat de TagVisualization getoond wordt en meebeweegt met de geplaatste Tag.
In Figuur 5 maken we een SurfaceWindow met een TagVisualizer die slechts één definitie bevat.

```
<s:SurfaceWindow x:Class="Surfinance.SurfaceWindow1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:s="http://schemas.microsoft.com/surface/2008">
  <Grid >
    <s:TagVisualizer x:Name="wensTagVisualizer">
       <s:TagVisualizer.Definitions>
          <s:ByteTagVisualizationDefinition
              Value="1"
              Source="ByteTagVisualization.xaml"/>
       </s:TagVisualizer.Definitions>
    </s:TagVisualizer>
  </Grid>
</s:SurfaceWindow>
```

FIGUUR 5: SURFACEWINDOW MET TAGVISUALIZER.

Bij het plaatsen van een Tag met waarde 1 wordt gezocht naar een definitie bij deze Tag. Wanneer deze definitie is gevonden wordt gezocht naar een XAML-file met de bestandsnaam zoals opgegeven in de Source property.

Deze file bevat de TagVisualization met daarin een ronde slider en een TextBlock die de waarde van de slider aangeeft. Deze waarde wordt met databinding gezet.

```
<s:TagVisualization x:Class="Surfinance.Visualizations.CarVisualization"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:s="http://schemas.microsoft.com/surface/2008"
    xmlns:local="clr-namespace:Surfinance">
  <Grid>
    <local:RoundSlider x:Name="priceSlider" Height="125"
Width="125">

<TextBlock Text="{Binding ElementName=priceSlider, Path=Value}" />
    </local:RoundSlider>
  </Grid>
</s:TagVisualization>
```

FIGUUR 6: TAGVISUALIZATION.

En hiermee zijn de twee stappen doorlopen die nodig zijn om Tags te kunnen gebruiken in een applicatie. Zoals gewoonlijk kunt je dit alles ook in de code doen, mocht je bijvoorbeeld dynamisch definities aan je TagVisualizer willen toevoegen.

Conclusie

Al met al zijn met deze eerste demo de gestelde doelen behaald. In de demo zitten alle features van de Surface unit verwerkt wat goed de toevoegde waarde van het apparaat laat zien. Daarnaast zijn we tegen verschillende technische en functionele aspecten opgelopen waar we vooraf nooit aan gedacht konden hebben. Dit kwam vooral, omdat je zo gewend bent aan de wereld van desktop en webapplicaties, die zich kenmerken door de zogenaamde WIMP's: Windows, Icons, Menu's en Pointers. Dit gaat niet op voor Surfaceapplicaties. Van de zes weken die we nodig hadden om de demo te ontwikkelen werd minimaal de helft beslagen door het bedenken van het user interactie ontwerp. Er ligt dan ook een grote uitdaging bij de User eXperience mensen en zij zijn in grote mate bepalend voor het al dan niet ontwikkelen van een succesvolle Surface applicatie. Of zoals Dr. Neil Roodyn van het Surface Team zei: "Technology is the easy part".

Bas Zweeris, is Sofware Engineer bij Capgemini. Hij is te bereiken via email bas.zweeris@capgemini.com.

Joost Verhoog Msc, is Software Engineer bij Capgemini. Hij is te bereiken via email joost.verhoog@capgemini.com.

Marijn Moerman, is Projectleider bij Capgemini. Hij is te bereiken via email marijn.moerman@capgemini.com.