Mobiele applicaties en aanraakbewegingen (1)

Marcus Perryman

Mobiele applicaties zijn hot. Diverse mobiele platformen als Nokia, Apple, Google (Android) en sinds kort ook RIM Blackberry bieden middels zogenaamde app stores de eindgebruiker toegang tot een grote hoeveelheid mobiele applicaties die deels gratis en deels tegen betaling te downloaden zijn. Als laatste in de rij is Windows Marketplace for Mobile, de applicatiewinkel voor Windows telefoons aangekondigd, wat dit najaar beschikbaar gaat komen op de nieuwe toestellen met Windows Mobile 6.5 en ook zelfs voor Windows Mobile 6.0 en 6.1. Met name die nieuwe versie Windows Mobile 6.5 biedt kansen voor applicatie ontwikkelaars. Blogger Marcus Perryman schreef een artikel over het ontwikkelen van applicaties voor Windows Mobile 6.5 en de ondersteuning van Touch Gestures.

Windows Mobile (en CE) biedt al sinds haar introductie ondersteuning voor interfaces met een aanraakscherm, maar met de release van Windows Mobile 6.5 krijgt het platform een nieuwe functie: ondersteuning voor gebaren (touch gestures). Gebaren zijn bedoeld voor een natuurlijkere interactie met het apparaat via het aanraakscherm, waardoor een emotionele band wordt gevormd tussen de gebruiker en de toepassingen onder hun vinger of stylus. Technisch gezien zijn gebaren een verzameling invoerpunten die worden gegenereerd door het scherm aan te raken in bepaalde patronen die het systeem herkent. De aanraakoplossing bestaat echter uit veel meer dan alleen de gebaren. Het draait ook om de animatie en interactie die plaatsvinden als gevolg van de gebareninvoer, zoals vloeiend door een lijst met items scrollen of ervoor zorgen dat een item in de gebruikersinterface blijft vastzitten aan de 'vinger' van de gebruiker om de illusie te wekken dat de gebruiker de scherminhoud rechtstreeks manipuleert.

Het verschil tussen gebaren en gewone muisinvoer

In eerste instantie lijkt het erop dat muisopdrachten en gebarenopdrachten veel gemeen hebben, zoals selecteren = klikken met de muis, pannen = muis bewegen en dubbel selecteren = dubbelklikken. De code voor gebarenherkenning is echter ontworpen voor een reeks invoerbeperkingen die aanzienlijk afwijken van muisinvoer. De muisinvoer voor mobiele apparaten wordt voornamelijk uitgevoerd met een aanwijsapparaat, zoals een stylus of een fysieke muis. Gebarenopdrachten kunnen echter op zeer verschillende manieren worden uitgevoerd, zoals met een vinger of duim of door het apparaat te schudden. Er zijn zelfs bredere invoermethoden mogelijk, zoals glimlachherkenning op een camera. De meeste gebruikers zullen in eerste instantie in contact komen met gebaren via invoer met het aanraakscherm en voor 6.5 hebben we daarom veel tijd besteed aan het perfectioneren van vingerinvoer en de bijbehorende reacties.

Het gebruik van een stylus of muis leidt tot verrassend nauwkeurige aanraakgegevens, waardoor kleine schermcontrols kunnen worden gerealiseerd. In deze situatie kunnen de toleranties voor klikken, dubbelklikken en tikken en vasthouden zeer klein zijn. Wanneer een vinger in plaats van een stylus wordt gebruikt, moeten er echter een aantal dingen veranderen. De toleranties voor klikken, dubbelklikken en tikken en vasthouden moeten aanzienlijk worden vergroot om te anticiperen op de grote verscheidenheid aan vormen en formaten van menselijke vingers. Wanneer u uw vinger over het scherm beweegt, verandert bovendien de vorm die tegen het scherm wordt gedrukt, op basis van de hoek waaronder u uw vinger houdt. Dit leidt vaak tot onverwachte invoerpunten aan het einde van een paninvoer waardoor de beweging verkeerd kan worden geïnterpreteerd.

Een opmerking over schermen

Windows Mobile-apparaten met aanraakmogelijkheden zijn gewoonlijk voorzien van een stylus met een plastic punt en een aanraakscherm dat is gebaseerd op resistieve technologie.

Kortweg is de resistieve schermtechnologie gebaseerd op twee lagen transparant, geleidend materiaal (Indium Tin Oxide of ITO) gescheiden door een luchtlaag die in stand wordt gehouden dankzij minuscule, isolerende plastic bolletjes. Als er op het scherm wordt gedrukt, worden de twee lagen vervormd en wordt contact tussen de twee gemaakt. Door de wijziging in resistentie kan de schermfirmware bepalen waar de stylus is geplaatst. Deze technologie heeft veel verschillende varianten.

Resistieve schermen hebben een aantal geweldige properties. Ze

zijn goedkoop, zeer nauwkeurig voor een stylus en ze blijven werken in behoorlijk onvriendelijke omgevingen, zoals vuile schermen.
Er zijn echter ook negatieve punten. Ze vereisen een bepaalde hoeveelheid druk voordat het scherm wordt vervormd en contact maakt tussen de geleidende lagen. Door de meerdere plastic lagen op het scherm en de luchtlaag neemt de helderheid van het scherm af. Goedkope en algemeen verkrijgbare traditionele resistieve schermen bieden slechts ondersteuning voor één aanraakpunt. Er zijn echter geavanceerdere digitale resistieve sensoren gedemonstreerd die meerdere aanraakpunten ondersteunen, maar dit is een toekomstige ontwikkeling. Daarnaast is het behoorlijk moeilijk om meer informatie te verkrijgen dan alleen de locatie van het punt (dat wil zeggen: de grootte van het aanraakgebied). Ook kan duurzaamheid een probleem zijn wegens het gebruik van bewegende onderdelen (dat wil zeggen: vervorming van het scherm).
Een andere aanraaktechnologie die in korte tijd zeer populair is geworden, is capacitief (zoals in de iPhone en Android G1). Bij deze technologie wordt de capacitieve property van verschillende gedeelten van het scherm doorlopend gemeten. Wanneer geleidend materiaal, zoals een vinger, op het scherm wordt geplaatst, veranderen de bijbehorende capacitieve properties en kan het schermstuurprogramma op basis van de veranderingen bepalen waar de vinger zich bevindt.
Capacitieve technologie heeft verschillende voordelen. Er is geen druk vereist om iets in te voeren, omdat er geen onderdelen zijn die moeten worden vervormd. Dit heeft een veel natuurlijker gebruik van de interface tot gevolg. Hoewel er extra materiaal op het scherm is geplaatst, is er geen luchtlaag, dus de optische helderheid is aanzienlijk verbeterd waardoor er minder achtergrondverlichting nodig is en er minder stroom wordt verbruikt. Er kunnen meerdere invoerpunten worden ondersteund. Ook kan informatie, zoals de grootte van het aanraakoppervlak en de druk, worden geëxtrapoleerd uit de capacitieve gegevens.
Er zijn echter ook negatieve punten. Doorgaans zijn de kosten hoger dan voor vergelijkbare resistieve schermen. Het is moeilijk om ondersteuning voor een stylus te bieden, omdat deze moet zijn gemaakt van geleidend materiaal en voldoende contact moet maken om de capacitieve property van het scherm te veranderen. Op bepaalde gedeelten van het scherm is de nauwkeurigheid minder dan bij resistieve schermen, bijvoorbeeld aan de randen van het scherm. Gecombineerd met het gebrek aan een stylus en lagere bemonsteringsfrequenties worden dingen als handgeschreven invoer daarom zeer moeilijk.
Er worden voortdurend andere invoertechnologieën ontwikkeld, maar momenteel vertegenwoordigen deze twee bijna de gehele markt voor mobiele apparaten.
Windows Mobile 6.5 is voornamelijk ontwikkeld voor resistieve schermen, omdat sommige invoergedeelten nog steeds vertrouwen op kleine controls en een hoge invoernauwkeurigheid nodig hebben die niet gemakkelijk met een vinger kan worden verkregen en daarom een stylus vereist. Sommige apparaatfabrikanten overwegen echter opties om capacitieve schermen te leveren.
Met het oog op de toekomst denkt het mobiele team na over het aanpakken van deze problemen en het ondersteunen van veel meer schermtypen, waaronder capacitieve schermen.

Welke gebaren worden ondersteund?

In Windows Mobile 6.5 hebben we vijf primaire gebaren geïmplementeerd:

Selecteren	De gebruiker tikt korter dan een specifieke tijdsperiode op het scherm, waarbij de beweging minder is dan een ingestelde drempelwaarde.
Dubbel selecteren	Een tweede selectieopdracht wordt gedetecteerd binnen de time-outperiode van de eerste selectieopdracht.
Vasthouden	De gebruiker tikt langer dan een specifieke tijdsperiode op het scherm.
Pannen	Zodra de bewogen afstand een drempelwaarde overschrijdt, wordt de aanraakbeweging weergegeven als pangebaar.
Scrollen	Aan het einde van een aanraaksessie, indien de voorafgaande punten ongeveer lineair zijn en een minimumsnelheid overschrijden.

Gebaren worden afgeleverd via een nieuwe opdracht WM_GESTURE die wordt begeleid door de gebaar-id en een handle waarmee de rest van de gebaargegevens, zoals de hoek en snelheid van een scrolopdracht of de locatie van een pangebaar, kan worden verkregen via de API GetGestureInfo (). Windows 7 voor desktopcomputers gebruikt dezelfde opdracht en biedt momenteel een enigszins afwijkende reeks gebaren dan beschikbaar is op mobiele apparaten. Let dus goed op wanneer u in MSDN-documenten zoekt naar de juiste gebaren (momenteel zijn de MSDN-documenten voor mobiele apparaten nog niet gepubliceerd).

Hoe werken gebaren?

U moet een aantal dingen weten wanneer u rechtstreeks met gebaren werkt:

- Gebaren en muisopdrachten zijn niet bedoeld om onderling uitwisselbaar te zijn. Hoewel u in WM 6.5 waarschijnlijk kunt wegkomen met muisopdrachten in plaats van gebaren voor selecteren of dubbel selecteren, verandert dit zeer waarschijnlijk in de toekomst, omdat er nieuwe hardware wordt ontwikkeld die optimaal kan profiteren van de aanraakinfrastructuur. De aanraaktechnologie is ontwikkeld om verschillende gedeelten voor aanraak- en muisinvoer mogelijk te maken, dus u kunt hierbij denken aan een apparaat met een muispad en een aanraakscherm, waarbij het aanraakscherm alleen kan worden gebruikt voor gebaren.
  In het ideale geval moet u uw code zodanig schrijven dat deze met muisopdrachten of met gebaren werkt, maar niet met beide tegelijk.
- Gebaren worden altijd afgeleverd aan het venster onder de aanvankelijke invoer (dat wil zeggen: de aanraaklocatie). U hebt hier waarschijnlijk nooit rekening mee gehouden voor muisopdrachten, maar het is logisch dat alle muisopdrachten worden afgeleverd aan het venster direct onder de muis op het moment dat een muisbewerking wordt uitgevoerd (tenzij de opdracht geforceerd wordt afgeleverd aan een specifiek venster met SetCapture()).
  Voor gebaren is dit enigszins anders. Als de gebruiker een scrolgebaar wil verzenden naar een specifiek gedeelte van het scherm, kan de aanraakinvoer beginnen in het 'doelvenster'. Omdat het beschrijven van een scrolgebaar echter tijd en afstand vereist, kan het einde van het gebaar in een ander venster elders op het scherm eindigen. De enginecode voor gebaren onthoudt dus waar het startpunt was en zorgt ervoor dat de scrolopdracht ook daar wordt afgeleverd. Hetzelfde geldt voor een vasthoudopdracht. De invoer kan 'rondzwerven' onder de vinger van de gebruiker, maar de vasthoudopdracht wordt verzonden naar het venster onder het aanvankelijke invoerpunt. Als wegens enige reden het venster onder de aanvankelijke in-

voer verloren gaat, gaat die gehele 'invoersessie' verloren. De sessie begint pas weer nadat de gebruiker zijn vinger heeft opgetild en weer op het scherm heeft geplaatst.

- We hebben ook speciale routing voor de opdracht WM_GESTURE toegevoegd om de grootte van het aanraakgebied te maximaliseren. Als we een opdracht WM_GESTURE in DefWindowProc() ontvangen, betekent dit dat het doelvenster de opdracht niet heeft verwerkt, omdat aanraakbewerkingen helemaal niet worden ondersteund of omdat het specifieke gebaar geen betekenis heeft voor de control. DefWindowProc() verzendt de opdracht WM_GESTURE naar het bovenliggende venster voor het geval er een grotere control is die het gebaar ondersteunt. Denk aan het voorbeeld van een formulier met labels: een pangebaar betekent niets voor de afzonderlijke control. Het formulier zelf kan echter redelijkerwijs reageren op de panbeweging en als gevolg daarvan het gehele formulier ronddraaien.
  Let dus op het volgende: stuur geen gebarenopdrachten van het bovenliggende naar het onderliggende venster. We hebben een lusbeveiliging toegevoegd om een stackoverflow te voorkomen, maar deze beveiliging werkt nog niet echt efficiënt en ik weet zeker dat u de beveiliging kunt omzeilen als u dat probeert!

Kan ik de lijst met gebaren uitbreiden?

Nee, momenteel kan dit niet, maar mogelijk zullen we dit in de toekomst overwegen.

Het gebruik van gebaren

De Windows Mobile 6.5 Developer Tool Kit bevat een aantal voorbeelden waarin u kunt zien hoe u de opdracht WM_GESTURE kunt gebruiken. De basisprincipes vindt u hieronder:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
        case WM_GESTURE:
        {
        GESTUREINFO gi = {sizeof(gi)};

// Go get the gesture - will return FALSE if the gesture engine is
not present in the system.
        if (TKGetGestureInfo(reinterpret_cast(lParam), &gi))
        {
            switch (wParam)
            {
                case GID_PAN:
                {
                    ...
                    fHandled = TRUE;
                }
                break;

                case GID_SCROLL:
                {
                    ...
                    fHandled = TRUE;
                }
                break;

            }
        }
        if (!fHandled)
        {
            return DefWindowProc(hWnd, message, wParam, lParam);
        }
        break;
    }
}
```

Belangrijk is dat het apparaat consistente reacties geeft zodat de gebruiker het vol vertrouwen kan bedienen

Wat heeft dit te maken met fysica?

Tot nu toe heb ik slechts de helft van het verhaal verteld... of misschien zelfs minder, omdat de gebruiker alleen de reactie op gebaren kan waarnemen. Zonder de juiste reacties kan de gebruiker geen echte band met het apparaat krijgen.

De belangrijkste is dat het apparaat consistente reacties geeft in alle toepassingen zodat de gebruiker vol vertrouwen zijn interacties met het apparaat uitvoert. De verwachte reacties zijn de volgende:

Selecteren en dubbel selecteren	Gedetailleerdere weergave of actie voor het geselecteerde item.
Vasthouden	Een snelmenu weergeven.
Pannen	Inhoud onder de vinger van de gebruiker beweegt in directe verhouding tot de beweging van de vinger (dat wil zeggen: directe manipulatie).
Scrollen	Inhoud onder de vinger van de gebruiker blijft bewegen in de richting van de laatste panbeweging en met dezelfde snelheid, en komt langzaam tot stilstand.

Op basis hiervan zien we dat er in feite slechts één gebaar is waarvoor een reactie op basis van fysica is vereist, en dat is het scrolgebaar. Wat we nodig hebben, is een manier om een consistente beweging te implementeren als reactie op het gebaar. Om dit mogelijk te maken, hebben we een aantal routines in de physics engine geïmplementeerd waarmee de aanroepende instantie de vorm van het gegevensgebied en het clientgebied kan identificeren. Vervolgens kunnen de snelheid en de hoek van het scrolgebaar worden ingevoerd (beide beschikbaar via GetGestureInfo()) en kan de locatie van het clientgebied doorlopend worden opgevraagd tot het gebaar tot stilstand komt.
Er zijn een aantal 'animatiemodi' beschikbaar voor de physics engine naast alleen vertraging en deze worden intern gecombineerd om de locatie van inhoud naar de grenzen van het gegevensgebied te bewegen en vervolgens van de ene modus over te schakelen naar een andere (dat wil zeggen: overschakelen van vertragen naar rubberbanding), zodat het eindpunt van de animatie altijd geldige gegevens weergeeft.
Door dit gedrag te implementeren in een centrale physics enginemodule kan elke aanraakbare component van de gebruikersinterface consistente en natuurlijke feedback geven aan de gebruiker. Dit is van essentieel belang bij het vergroten van het vertrouwen dat de gebruiker in het apparaat heeft en bij het tot stand brengen van emotionele betrokkenheid bij de ervaring.
Bekijk het voorbeeld van de physics engine in de Windows Mobile 6.5 Developer Tool Kit voor meer informatie.

Links

http://developer.windowsmobile.com

Marcus Perryman, blogt op http://blog.msdn.com/marcpe.