– Lab3 –
Grafisk klient – Händelsestyrd programmering

1 Syfte

Målet med denna laboration är att ni skall lära er att konstruera program med hjälp av händelsestyrd programmering. Dessutom ingår ett par andra aspekter, såsom programmering mot ett givet gränssnitt (serverns gränssnitt). Labben ger också en insikt om fönsterhantering i Windows.

2 Uppgift

Labben går ut på att skapa ett grafiskt interface till den klient som ni implementerade i laboration 2. Användare skall alltså kunna mata in planeter och skicka dessa till servern. I denna laboration sker detta via ett grafiskt fönster. Utöver funktionaliteten i klienten ni gjorde i laboration 2 kommer denna klient bland annat kunna hantera solsystem, d.v.s. grupper med planeter, och skicka dessa till servern. Interfacet och koden för servern skall vara densamma som i lab 2.

3 Krav på klienten

Funktionella krav:

1. Användaren skall på ett enkelt och intuitivt sätt kunna mata in nya planeter.
2. Stöd skall finnas för att spara och läsa in planetsystem från fil.
3. Planeter som är inmatade (eller inlästa från fil) men ännu ej skickade till servern (så kallade *lokala planeter*) skall presenteras, i en lista eller dylikt, för användaren. Denna presentation skall minst innehålla planetens namn.
4. Användaren skall godtyckligt kunna välja en mängd av de lokala planeterna och kunna antingen skicka dessa till servern, eller spara dessa i en fil på hårddisken.
5. Planeter som är skickade till serven, och som fortfarande lever, skall presenteras, i en lista eller dylikt, för användaren.
6. Meddelanden från servern skall tas emot och presenteras. Det skall vara möjligt att se historik i dessa meddelanden. Det är alltså inte okej att använda popupfönster för denna funktionalitet.
7. Då meddelande från servern om att en planet har dött, eller försvunnit utanför området mottagits, skall planeten tas bort från listan över skickade planeter.
8. Antalet existerande lokala planeter skall presenteras för användaren.

Implementationstekniska krav:

1. Programmet skall skrivas i språket C.
2. Fönsterhanteringen och interaktionen med operativsystemet skall ske m.h.a. win32.
3. För att underlätta designen av fönstren får resurseditorn i Visual c++ användas, dock får inga andra hjälpmedel, typ Borland Builder, användas.
   Tips: Använd fönstertypen *Dialog*.
4. *Alla* fönster i klienten skall vara av så kallad icke modal typ.
5. Ingen automatgenererad kod, förutom den i filerna `.rc` och `resource.h` om resurseditorn används, får finnas i projektet.
6. Programmet måste innehålla *minst två* fönster, varav båda skall innehålla objekt (knappar, boxar o dyl) som användaren direkt kan använda (klicka på, mata in information i osv.)
7. Det skall vara möjligt att ”tabba” sig mellan objekt i minst två fönster samtidigt. Med

”tabba” menas att med hjälp av tabulatortangenten byta fokus på objekt i ett fönster.

8. Ett av fönstren i klienten skall vara huvudfönstret. När användaren väljer att stänga huvudfönstret skall alla övriga fönster stängas, samt applikationen terminera.
9. Klienten skall vara utformad så att vid minst en händelse skall planetinformation skickas mellan två fönster.
10. Formatet på planetfiler skall vara enligt följande. I filen skall *endast* en mängd av structen `planet_type` (finns specificerad i `wrapper.h`) finnas, lagrade i binär form. Programmet skall kunna läsa och förstå filen `solsystem.dat` som finns på hemsidan.

Dessutom måste en design av den grafiska klienten godkännas av laborationsassistent *före* implementationsarbetet påbörjas.

4 Filer

Ni utgår från era egna filer som ni gjorde i lab 2. Notera att servern inte ändras alls. Dessutom finns det en fil, `empty.c`, att ladda ner från hemsidan, denna fil skall läggas in i ett *tomt* win32 projekt.

5 Krav för godkänt

För att få godkänt på labben skall:

1. En programdesign visas upp och godkännas av laborationsassistent.
2. En fullt fungerande klient, enligt designen och kravspecifikationen, implemenenteras.
3. En implementation redovisas för samt godkännas av laborationsassistent.
4. En laborationsrapport vara godkänd av laborationsassistent. **Viktigt!** Denna rapport skall innehålla ett testprotokoll som visar vilka tester ni utfört på systemet och naturligtvis utfallet för varje test. Det är med testprotokollet du visar för användaren (labbassen) att din lösning fungerar som det är tänkt. Hur testar ni tex att servern skickar tillbaka information till rätt klient?

6 Hur kommer ni igång?

Vi rekommenderar följande arbetsordning i början av projektet:

1. **LÄS, LÄS, LÄS!**
   Att få fönsterhantering i win32 att fungera är inte helt enkelt. En förståelse för hur detta hanteras fås genom att läsa i MSDN, pekare till lämpliga avsnitt finns på hemsidan. I det första skedet är det viktigaste att försöka hitta hur icke-modala fönster fungerar, och hur man knyter en händelse på ett objekt (knapp el. dyl) till någon kod som ni skrivit.
2. **Skapa ett tomt icke-modalt fönster**
   Använder ni resurseditorn i VC++ är detta enkelt, läs i appendix A om hur man gör detta.
3. **Skapa en knapp i fönstret**
   När ni skapat denna knapp så måste ni få den att köra just er förnämliga kod. Gör t ex en enkel kod som genererar en messagebox varje gång ni trycker på knappen.
4. **Skapa en editBox i fönstret**
   Kan ni få knappen att räkna upp och visa en siffra i editboxen när man trycker på den? När ni kommit så här långt har ni kommit en mycket god bit på väg i projektet. När ni känner att ni verkligen förstår hur programmet fungerar, och hur mekanismerna bakom fungerar så kan ni gå vidare.
5. **Skriv en design över hur klienten skall fungera**

I designen skall framgå vilka fönster ni skall ha, vilka objekt som skall finnas i varje fönster, vad som händer när användaren manipulerar olika objekt, samt hur mjukvaruarkitekturen på klienten skall se ut. Kort sagt skall ni med hjälp av designen själv förstå hur programmet kommer att fungera, och inte minst, lyckas övertyga laborationsassistenterna om samma sak.

6. **Visa upp din design, samt redovisa din provimplementation för en laborationsassistent.**
   Om vi inte ser att ni har förstått hur fönsterhanteringen fungerar, samt hur klienten skall fungera skall ni inte gå vidare i projektet före detta steg är klart. Ingen godkänd design och redovisning av provimplementation = Ingen hjälp av labassistent.

7 Övriga tips

Denna laboration kommer till stor del att bestå i att hitta vilka API anrop som behövs samt hur de fungerar, detta görs genom att leta och hitta i MSDN. Ett bra avsnitt är Platform SDK och speciellt User Interface Services för grafiken (det finns redan mycket användbara knappar, listrutor och textboxar).

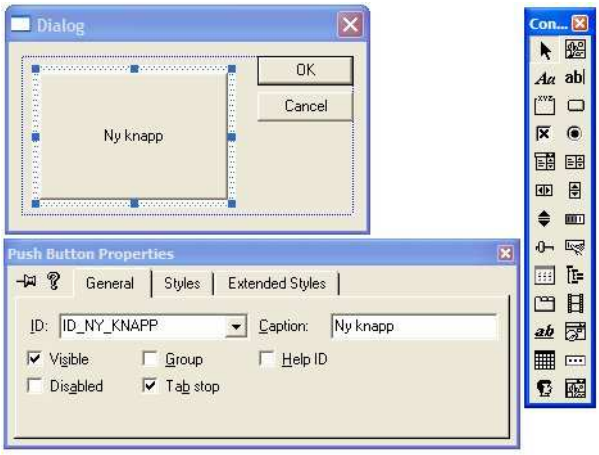
Några tips:

1. Gör en modeless Dialogbox på högsta nivå
2. En egen händelse (USER DEFINED EVENTS) görs genom WM_USER+ngt tal
3. I messageloopen kan IsWindow och IsDialogMessage vara användbara
4. OpenFileDialog finns i `wrapper.c` används för att öppna en fildialog
5. Tänk på hur du stänger ett fönster! Skillnad på modal och modeless.

Mer information om tips & tricks och vanliga frågor kommer att finnas tillgängligt på WebCT.

A Resurseditorn i Visual Studio

1. I menyn klickar du Insert och väljer sen Resource.
2. Vill du sen ha en Dialog box markerar du denna och klickar New.
3. Du får nu två defaultknappar: **OK** och **Cancel**.
4. Du infogar en ny knbap helt enkel genom ett ”drag & drop” från verktygsmenyn. I bilden så har vi lagt till en ny knapp i dialogrutan.
5. Dubbelklickar man på knappen så får man fram knappens egenskaper. I bilden nedan har vi döpt knappen till Ny knapp (**Caption**) och den har fått **ID**: ID_NY_KNAPP, det är med denna man särskiljer vilket av fönstren (knapparna i vårt fall) i dialogrutan som har emottagit en händelse.



   Sen har vi bestämt att knappen skall vara synlig och att om man ”tabbar” sig fram mellan de olika knapparna i dialogrutan så ska den stanna vid denna knapp (Tab-ordningen kan fås genom menyn Layout och Tab Order, och sedan klickar man på knapparna i den ordningen man vill ha). Sedan kan man välja olika stilar man vill ha på sin knapp, det är bara att prova sig fram. Och för andra resurser, som tex edit rutor, så finns det andra typer av stilar och val man kan göra.
6. När man sparat sin resurs så sparas den i en `.rc` fil. Om man vill gå tillbaka och lägga till eller ändra så klickar man bara på den.
7. För att kunna använda resursen i sin kod är det bara att inkludera header filen `resoruce.h`. Sedan kan man använda sina resursen, t.ex. i skapandet av en dialog (tänk på att hela dialogen också får ett eget ID) och i sina händelseloopar.