Mobiilijärjestelmien ohjelmointi

11.03.2004

Sasu Tarkoma
sasu.tarkoma@cs.helsinki.fi

Sisällys

- Johdanto
- Tarvittavat ohjelmat
- Harjoitustyön esittely
  - Vaatimukset
  - Demo
  - Arvostelu
- Esimerkkisovellus
- Visual Studio
- Debuggaus
- Laitteeseen siirto

Yleistä

- Työ tehdään Microsoft Visual Studio 6.0:lla sekä Nokia Series 60 SDK 1.2:lla
  - Muita tarvittavia ohjelmia:
    - ActivePerl 5.8
    - JRE 1.3.1 (AIFBuilder ja Sisar)
    - Asennusohjeet on verkossa
- Harjoitustyön voi tehdä myös Borland C++ Mobile Editionilla, jonka saa Nokia Forumista
  - Turvautuu vähemmän komentoriviin
- Ohjelmat on asennettu mikroluokkaan D325
  - Marjaniemi-koneet (12 konetta)
  - Ohjeet ja nämä kalvot tulevat kotisivulle
- Symbian päivystys torstaisin 10-12 A217

Huomioita

- Koneille ei saa jättää lähdekoodeja muiden luettavaksi
  - Kannattaa pitää projektihakemistoa fs:llä kotihakemistossa ja siirtää tarvittaessa koneelle
  - Visual Studio tallettaa käännetyt tiedostot eri paikkaan ja niitä ei tarvitse erikseen poistella ellette halua
  - Käyttäkää ryhmänne numeroa projektin nimessä niin ei tule konflikteja..

Dokumentaatio

- Series 60 SDK dokumentaatio
  - API-dokumentaatio
    - `\Symbian\6.1\Series60\Series60Doc`
  - Getting started
- Symbian Developer Library
- Forum Nokia

Ympäristön asennus

- Ennen Visual Studion ja Nokian SDK:n käyttöä muistakaa ajaa vcvars32.bat
- C:\Program Files\Microsoft Visual Studio\VC98\Bin\vcvars32.bat
- Tämä skripti alustaa tarvittavat ympäristömuuttujat

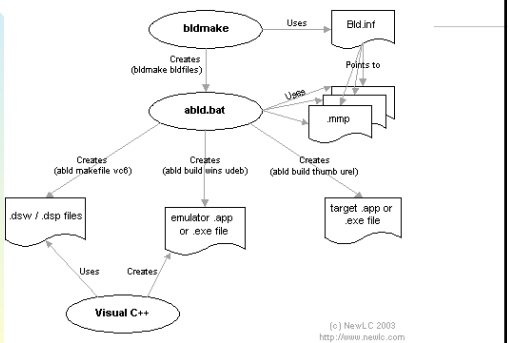
Hakemistorakenne

- \epoc32\data\ - sisältää epoc.ini ja asetuksia
- \epoc32\release\ - julkaistava materiaali
- \epoc32\release\wins\ - emulaattori
- \epoc32\release\wins\udeb – debug build emulaattorille
- \epoc32\release\wins\udeb\z\ - Symbian z: drive sisältää kaikki paitsi jaetut DLL:t, jotka ovat \system\libs
- \epoc32\wins\c\ - datatiedostot mutta ei ajettavaa koodia, emulaattorissa tallennetut tiedostot tulevat tänne
- Symbian tiedostojärjestelmä:
  - c: RAM disk, full read/write file system
    - ..\epoc32\wins\c (emuloitu c:)
  - z: ROM disk, read-only file system.
    - ..\release\wins\udeb\z (emuloitu z:)

Hakemistorakenne II

- Udeb = unicode debug hakemisto ohjelmille, urel = unicode release hakemisto ohjelmille
- Wins = Windows emulator, armi = ARM build
- \epoc32\ - sets apart all shared tools
- \epoc32\gcc\bin – sisältää GNU compilerin
- \epoc32\tools - Työkaluja
- \System\ systeemitiedostot
- Emulaattorissa näkyvät (z: drive)
  - \system\apps - applikaatiot
    - \system\apps\appname\appname.app

Prosessi



Harjoitustyö

Harjoitustyö

- Harjoitustyön aiheena on toteuttaa muistipelin logiikkakomponentti (engine) Symbian OS C++ -sovellukseen.
- Opiskelijat saavat valmiin käyttöliittymäkomponentin (.app -tiedosto, graafinen sovellus) lähdekoodeineen, johon tulee toteuttaa annetun rajapinnan mukainen logiikkakomponentti, engine (.dll -tiedosto).
- Muistipeliä pelataan yksin, tavoitteena löytää kaikki parit mahdollisimman nopeasti ja ilman virheitä.
- Nämä ohjeet löytyvät myös verkosta:
  - http://www.tol.oulu.fi/~antti/Mobo/index.html

Muistipeli

Säännöt

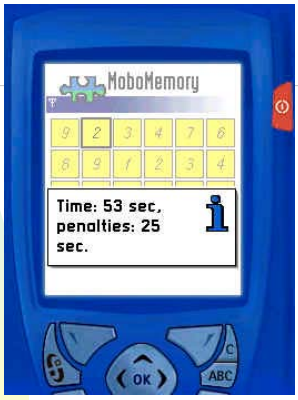
- Muistipelin säännöt ovat tutut. Pelilaudalla on parillinen määrä peitettyjä symboleita.
- Pelaaja paljastaa aina kaksi pelimerkkiä kerrallaan. Jos merkit ovat samat, ne saa jättää näkyville. Jos merkit ovat erit, molemmat merkit peitetään, ja käyttäjä jatkaa parien etsintää.
- Kun kaikki parit on löydetty, peli päättyy.

Säännöt II

- Symboleiden lukumääräksi rajataan tässä harjoitustyössä 30 kappaleeseen (15 paria), eli peliruudukon kooksi tulee 6 x 5 (valmiina annettu koodi piirtää kuusi ruutua per rivi).
- Rajaus on tehty Series 60 -puhelimen näytön koko huomioonottaen. Optionaalisena laajennuksena voi tehdä pelin jonka symbolien lukumäärä on järkevissä rajoissa (näytön koko) muuttuva.

Pelin pisteytys

- Jokainen peli pisteytetään. Mitä nopeammin peli ratkaistaan ja mitä vähemmän virheellisiä pareja pelaaja kääntää, sitä paremman pistemäärän pelaaja saa.
- Matalampi pistemäärä on siis parempi. Pistemäärän laskenta tapahtuu siten, että jokaisesta pelatusta sekunnista saa pisteen, ja jokaisesta väärin käännetystä parista saa viisi pistettä.



Pelin tallennus

- Pelisovellus tallentaa sekä pelitilanteen että kymmenen parasta pelitulosta. Jokaisesta pelistä joka mahtuu kymmenen parhaan joukkoon, tallennetaan pelaajan nimi, pelin päivämäärä sekä pistemäärä.
- Peliohjelmassa voi tarkastella kymmentä parasta pelitulosta paremmuusjärjestyksessä.
- Pelaajan nimi kysytään käyttäjältä vain jos pelaajan tulos pääsee kymmenen parhaan listalle.
- Peliä on pystyttävä jatkamaan edellisestä pelitilanteesta, vaikka ohjelma olisi lopetettu..

Ohjelman demo

Käynnistä

Engine I

- Logiikkakomponentin eli enginen vastuulla on pelisääntöjen toteutus, pelin tilan hallinta ja tietojen tallentaminen tiedostoon ja lukeminen tiedostosta
  - engine pitää yllä tilatietoa käännetyistä pareista ja vielä kääntämättömistä pareista
  - engine huolehtii siitä, että käyttäjä voi kääntää aina vain 1 tai 2 pelimerkkiä kerrallaan
  - kun 1. merkki käännetään, on käännettävä joku toinen merkki. 1. merkkiä ei saa kääntää piiloon sen esiin kääntämisen jälkeen.

Engine II

- jos yritetään kääntää jo käännettyä merkkiä, ei tehdä mitään.
- engine huolehtii siitä, että jos pelaaja kääntää samanlaiset parit, ne merkitään pysyvästi käännetyksi
- engine huolehtii siitä, että erilaisten merkkien kääntämisen jälkeen ne käännetään takaisin piiloon
- engine huolehtii siitä että jo löydettyjä pareja ei voi kääntää (takaisin piiloon)
- ilmoittaa käyttöliittymälle annetun ohjelmointirajapinnan kautta parien löytymisestä ja toisaalta siitä että pareja ei löydetty

Engine III

- huolehtia pistelaskun oikeellisuudesta.
- engine huolehtii siitä, peliä ei voi jatkaa sen päättymisen jälkeen.
- pelitilanteen tallentamisesta ja muistiin lataamisesta, käyttäen hyväksi Symbianin tiedostojärjestelmän ohjelmointirajapintoja.
- Pelitilanteesta tallennetaan nykyinen pelilaudan tilanne, nykyinen pelin pistemäärä ja kesto, sekä kymmenen parasta (tai vähemmän, jos kymmentä ei vielä ole) pelitulosta

Vaatimuksia

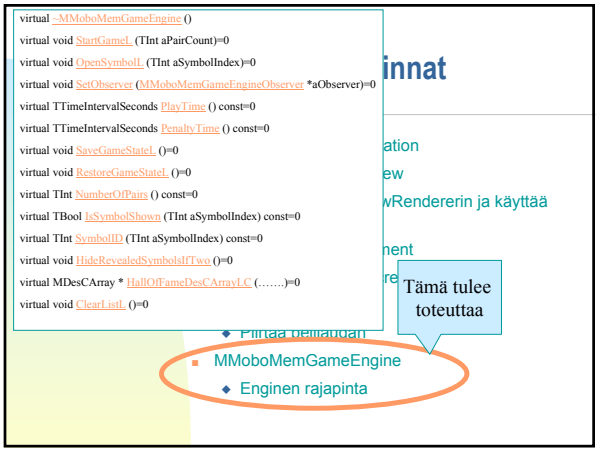
- Peliohjelma ei saa vuotaa muistia tai muita järjestelmän resursseja, ei edes virhetilanteissa (kun tapahtuu poikkeus, ns leave).
- Sovelluksen tiedosto, jonne pelitilanne tallennetaan, ei sekään saa korruptoitua jos tiedostoa kirjoitettaessa tapahtuu poikkeus.
- Tiedosto (virhetilannetta edeltävä sisältö) on voitava lukea muistiin, vaikka viimeksi tehty kirjoitus epäonnistuisi.

Optionaaliset ominaisuudet

- Pelitilanne keskeytetään kun peliohjelma lopetetaan tai peliohjelma siirtyy taustalle, ts. ei ole enää aktiivinen sovellus. Kun sovellus ei ole aktiivinen, peliaikaa ei lasketa.
- Pelilaudan koko voidaan määrätä pelikohtaisesti. Erilaiset pelilaudan koot olisi syytä ottaa huomioon myös pistelaskennassa...
- Oma käyttöliittymä tai annetun käyttöliittymän parantaminen esimerkiksi grafiikkaominaisuuksia parantamalla

Mitä annetaan

- Valmis käyttöliittymän referenssitoteutus lähdekoodina.
- Logiikkapuolen rajapintaluokka ja muita aputiedostoja.
- Harjoitustyössä tätä rajapintaa ei saa muuttaa tai laajentaa millään tavalla.
- Logiikkaosa on toteutettava omaksi DLL:ksi, johon sovellus (käyttöliittymä) linkitetään.
- Sovellusosa on kytköksissä logiikkaosaan ainoastaan annetun rajapinnan kautta.
- Toimivan ohjelman binäärit, jolloin malliratkaisun toiminnallisuuteen voi tutustua sekä emulaattorissa että oikealla Series 60 -laitteella.



Luokat ja rajapinnat

- MMoboMemGameEngineObserver
  - Observer-rajapinta jolla engine tiedottaa käyttöliittymäkomponentteja pelin tapahtumista
    - Start/end game
    - Query name
    - Invalid pair / Symbol pair was found
- CHelloWorldPLusAppUi
  - Toteuttaa Observerin ja saa tapahtumia engineltä
  - Sisältää näkymän ja käyttää engineä
- MoboMemEngineFactory

EXPORT_C MMoboMemGameEngine * MoboMemEngineFactory::CreateEngineL ( ) [static]
Tämä metodi tulee toteuttaa

Ohjelman vaiheet I

- Engine luodaan sovelluksen document -objektissa
- Sovelluksen Ui -objekti asettaa itsensä enginen tarkkailijaksi omassa konstruktiovaiheessaan
- StartGameL:ää kutsutaan Ui -objektista käsin kun käyttäjä haluaa aloittaa uuden pelin (valikosta)
  - engine kutsuu tarkkailijan GameStartedNotifyL -funktiota
- Käyttöliittymän päivityksen yhteydessä kutsutaan enginen funktioita NumberOfPairs, IsSymbolShown, SymbolID

Ohjelmat vaiheet II

- Tämän jälkeen kutsutaan toistuvasti OpenSymbolL -funktiota, kun näppäimistötapahtumia käsitellään (sovelluksen View -objekti)
  - availlen symboleita ja toivottavasti löytäen niitä etsittäviä pareja.
  - Engine kutsuu tarkkailijan function PairsFoundNotifyL, InvalidPairsOpenedNotifyL ja GameEndedNotifyL, sekä mahdollisesti function GetPlayerNameL.
  - tässä yhteydessä UI kutsuu myös enginen funktiota HideRevealedSymbolsIfTwo, jotta engine piilottaisi sellaiset juuri paljastetut kaksi symbolia, jotka eivät ole pareja

Huomioita

- Annettuun lähdekoodiin ei saa tehdä harjoitustyötä toteutettaessa muutoksia lukuunottamatta MoboMemEngineFactory -luokkaa.
  - Tähän luokkaan lisätään oman engine -objektin luominen.
  - Tätä lisäystä lukuun ottamatta, työskentelet ainoastaan engine -projektin parissa

Päivämäärät

- 1-vaiheen palautus :
  - palauta suunnitelma ja osatoteutus assistentille viikolla 14.
- 2-vaiheen palautus :
  - palauta lopullinen ohjelma assistentille viikolla 17(deadline 23.4 klo 16.00)

Vaihe 1: Suunnitelma

- Ensimmäisessä vaiheessa palautetaan suunnitelma, josta selviävät ohjelman logiikka, luokat ja käytettävät toteutusratkaisut.
- Suunnitelma on muodoltaan vapaa, mutta esimerkiksi luokkakaavion ja sekvenssikaavion käyttöä suositellaan.
- Suunnitelma esitellään harjoituksissa assistentille. Myös toteutuksen on oltava vauhdissa, ja enginen tulee toteuttaa joitakin sille asetettuja vaatimuksia, jotka esitetään assistentille.

Vaihe 2: lopullinen ohjelma

- Vaiheessa 2 palautetaan lopullinen kaikki vaatimukset täyttävä ohjelma.
- Ohjelma dokumentaatioineen toimitetaan zipattuna harjoitustyön tarkastajalle. Toimituksessa on ajettavan ohjelman lisäksi myös lähdekoodi ja mahdolliset muut kääntämisen kannalta tarpeelliset tiedostot. Lisäksi tulee toimittaa käännös, asennus ja ajo ohjeet readme.txt -tiedostossa, mukaanlukien ryhmän jäsenten henkilötiedot.
- Ohjelma on voitava kääntää komentorivikomennoilla bldmake bldfiles ja abld build wins udeb, jonka jälkeen ohjelman testaaminen emulaattoriympäristössä on oltava mahdollista.

Arvostelu

- Harjoitustyöstä ei jaeta pisteitä, vaan se arvostellaan asteikolla hyväksytty/hylätty.
- Hyväksyminen edellyttää, että kaikki vaiheet on suoritettu hyväksytysti.
- Kurssin suoritus edellyttää hyväksytysti suoritettua harjoitustyötä.

Esimerkkisovellus

Esimerkkisovellus

1. Asenna kysymys/vastaus-ohjelma hakemistoon (c:\qanda)
2. Hakemistorakenne
3. Tiedostot
   - group\qanda.mmp group\qaeng.mmp
   - group\bld.inf group\qanda.rss
   - inc\qanda.hrh inc\qanda.pan
   - src*.cpp
   - engine\qaeng.cpp
   - sis\qanda.pkg

Käännösympäristö

1. Luo abld.bat-tiedosto group-hakemistoon ajamalla siellä komento:
   - `bldmake bldfiles`
2. Luo Visual Studion projekti komennoilla:
   - `abld build wins`
   - `abld build vc6`
3. Projektille luotu hakemisto:
   - `\Symbian\6.1\Series60\Epoc32\BUILD\QANDA`

Kääntäminen Visual Studiossa

1. Avaa projekti. Projekti sijaitsee hakemistossa:
   - `\Symbian\6.1\Series60\Epoc32\BUILD\QANDA\GROUP\QANDA\`
2. Käännetty ja linkattu ohjelma asennettiin:
   - `\Symbian\6.1\Series60\Epoc32\Release\wins\udeb\z\system\apps\qanda`
3. Kopioi vielä qfile.txt ja afile.txt asennushakemistoon.

Ajaminen emulaattorissa

- Käynnistä sovellus Visual Studiosta ja anna emulaattori suoritettavaksi ohjelmaksi. Emulaattori löytyy seuraavasta hakemistosta:
  - `\Symbian\6.1\Series60\Epoc32\Release\wins\udeb\Epoc.exe`

Debuggaus

- Avainten käyttö (projekti):
  - ctrl+alt+shift+a (paljonko ohjelmasäie on varannut muistia keosta)
  - ctrl+alt+shift+b (montako tiedostokahvaa sovelluksella on auki)
  - ctrl+alt+shift+p (aseta systemaattisia tai randomaattisia muistin varauksien ja/tai tiedoston avauksien epäonnistumisia)

Debug tulostus

- **Tulostus Visual Studion Debug-konsoliin**
  - Käytetään kysymys-vastaus-ohjelmaa. Lisää seuraava rivi HandleCommandL-funktion loppuun CQAApUi-luokan toteutuksessa:
  - RDebug::Print(_L("Käyttäjä valitsi komennon %i"), aCommand);
  - Aja ohjelma debug-tilassa (F5). Käyttäjän valitsemaa komentoa vastaava numero tulostetaan Visual Studion Debug-konsoliin.

Debug breakpoint

- Breakpointin käyttö Visual Studiossa
  - Kuten normaalisti Visual Studiossa kanssa
    - Insert/Remove breakpoint
    - Go/Step into/Run to cursor…

Uuden sovelluksen luominen

- Application Wizard -työkalulla pystyy luomaan minimaalisen Symbian-sovelluksen Visual Studion kautta, jota pystyy sitten kehittämään eteenpäin.
- Application Wizardin asennus:
  - Kopioi AvkonAppWiz.awx and AvkonAppWiz.hlp tiedostot \Program Files\Microsoft Visual Studio\Common\MsDev98\Template -hakemistoon
- Tämä ei toistaiseksi toimi mikroluokassa

Uuden projektin luominen

1. Käynnistä Visual Studio
2. Valitse File->New
   - Projects: Series 60 AppWizard
   - Location: Esim. c:\mobo\
   - Project name: HelloWorld
3. Sovelluksen asetukset
   - Application title: HelloWorld
4. Tiedostot
5. Rakenna ohjelma/Aja ohjelma

Asentaminen puhelimeen

- Tämän voi tehdä esimerkiksi Bluetooth yhteyden tai infrapunan yli. Tässä käytetty Bluetooth-yhteyttä.
- Joudut todennäköisesti muokkaamaan qanda.pkg-tiedoston hakemistopolkuja ennen makesis-komennon ajamista.
- Käännä kysymys-vastaus-ohjelma laitteelle group-hakemistossa:
- `abld build armi urel`

Asentaminen II

- Näytä mihin hakemistoon käännös sijoitettiin:
- `\Symbian\6.1\Series60\Epoc32\Release\armi\urel`
- Luo sis-paketti makesis-ohjelmalla sis-hakemistossa. Joudut todennäköisesti muokkaamaan qanda.pkg-tiedoston (lähde)hakemistopolkuja ennen makesis-komennon ajamista:
- `makesis qanda.pkg`

Asentaminen III

- Luo Bluetooth-yhteys kännykkään PC:ltä
- Käynnistä "PC Suite for Nokia 1234" -ohjelma
- Asenna sovellus PC Suite -ohjelman Tools-valikosta löytyvän "Install device software"-toiminnon kautta
- Aja ohjelma puhelimessa