10/08/2012

FACULTATEA DE AUTOMATICA SI CALCULATOARE

ELEMENTE DE GRAFICA PE CALCULATOR

Laborator 1

1. C/C++

In cadrul laboratoarelor de EGC va fi folosit C/C++ cu STL(doar liste/vectori). Nu se vor folosi mecanisme de tipul mostenire, polimorfism, metaprogramare cu template-uri, etc. La acest link gasiti un tutorial de C/C++ ce acopera tot ce veti avea nevoie in cadrul laboratoarelor de EGC: http://www.cplusplus.com/doc/tutorial/

2. IDE

Microsoft Visual Studio este IDE-ul(Integrated Development Enivornment) in care vom lucra la laborator. Aceasta nu inseamna ca sunteti fortati sa il folositi pe statiile voastre. Codul laboratoarelor poate fi usor mutat in alte IDE-uri si/sau in alte sisteme de operare.

Pentru cei ce nu sunt familiari cu vreun IDE este bine de punctat ca exista 2 mari notiuni: solutiile si proiectele. La nivelul cel mai simplu de exprimare proiectul este „programul vostru" cu toate setarile de rigoare(dependinte, link, etc), iar solutiile sunt „manageri" de proiecte(proiecte dependinta, batch builds, etc).

Cunostintele minime de utilizare de IDE-uri de care veti avea nevoie la laborator sunt:

- Debug/run -> incepe executia programului
- Compile -> compileaza programul si creeaza executabilul

Cunostinte extra ce va vor fi utile:

- Toggle breakpoint -> in timpul executiei programul se va opri si va astepta, oferind oportunitatea de a observa variabile/clase/etc in starea lor curenta
- Step Into/Step Over -> executie instructiune cu instructiune

3. Primii pasi in grafica

Primul lucru pe care il puteti intreba la o ora de grafica este „cum apar obiectele pe ecran?". Explicatia intuitiva ar fi urmatoarea:

- se pleaca de la niste date reprezentate prin puncte sau alte metode (ex: o carte cu coperti verzi si un logo pe prima pagina).
- Aceste date sunt intai procesate de programul pe care il scriem a.i. sa aiba o semnificatie(ex: cartea este deschisa, cu coperta in sus, luminata de sus cu lumina rosie si de jos cu o lumina albastra)
- Apoi datele sunt trimise la placa grafica ce printr-o serie de transformari (pe care le vom explora incepand din acest laborator) ajung intr-un format usor de procesat pentru programele de pe placa video.
- Dupa procesare, driverul video afiseaza rezultatul pe ecran.

Grafica se preda cel mai usor plecand de la conceptele simple, clare din punct de vedere matematic si apoi avansand la cele mai avansate(atat ingineresti cat si algoritmice si matematice). De aceea in cadrul laboratorului vom folosi intai un framework simplu pentru primele concepte si apoi o librarie pentru a facilita interactiunea dintre cod si driverul video.

4. Obiectele

Primul pas pentru a desena ceva este acela de a defini obiectele pe care ne dorim sa le desenam. Un obiect este reprezentat de o serie de puncte in spatiu (2D pentru acest laborator) care sunt legate intre ele formand triunghiuri. Legatura dintre ele se numeste topologie.

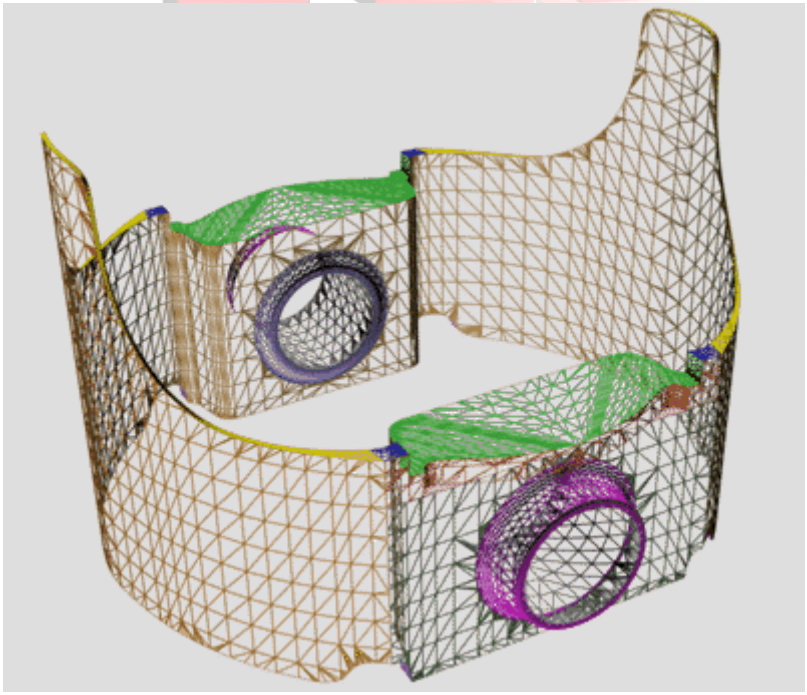
De exemplu un triunghi este format din 3 puncte:

A (0,0) B (0,10) C(10,10);

Si dintr-o lista de indecsi in lista de puncte ce, luati 3 cate 3, definesc topologia:

0 1 2

Folosind acest proces simplu se pot defini obiecte de forma oricat de complexa, independent de tipul spatiului (2d,3d):



5. Sisteme de coordonate

Un obiect nu are sens decat daca este reprezentat fata de ceva. Un centimetru pe ecran poate reprezenta un centimetru sau un kilometru in spatiul reprezentat. Este doar o problema de referinta. De aceea este vital ca tot timpul sa pornim de la un sistem de coordonate parinte ce ne defineste spatiul.

Toate celelalte obiecte vor avea acum sens atat timp cat se pastreaza conventia aleasa.

6. Transformarile

Pentru a modela si reprezenta un obiect trebuie sa putem sa il transformam din starea sa initiala in starea dorita. Pentru aceasta, intr-un spatiu 2D exista mai multe transformari:

Translatie:
x' = x +tx;
y' = y +ty;

Rotatie:
x' = x*cos(u) – y*sin(u)
y' = x*sin(u) + y*cos(u)
Rotatia relativa la un punct oarecare se rezolva in cel mai simplu mod prin:

- translatarea atat a punctului asupra carui se aplica rotatia cat si a punctului in jurul caruia se face rotatia a.i. cel din urma sa fie originea sistemului de coordonate.
- rotatia normala,
- translatarea rezultatului a.i. punctul in jurul caruia s-a facut rotatia sa ajunga in pozitia sa initiala

Scalare:
x' = x* factorx;
y' = y* factory;
Daca factorx=factory atunci avem scalare uniforma, altfel avem scalare neuniforma. Scalarea relativa la un punct oarecare se rezolva similar cu rotatia relativa la un punct oarecare.

Dupa cum puteti sa observati termenul „relativ" este foarte des intalnit in grafica :)

7. Framework-ul.

Pentru a putea face primii pasi in grafica fara a ne lovi de cum accesam driverul video sau de cum sincronizam firul de executie sau alte probleme vom folosi un mic framework. Singurele fisiere ca va sunt necesare atat pentru laborator cat si pentru viitoarea tema sunt main.cpp si Transformation2d.cpp. In rest nu este necesar sa lucrati cu sau sa modificati codul suport, insa sunteti incurajati sa explorati!

- Frameworkul este initializat prin constructor astfel:

```
WorldDrawer2d
wd2d(argc,argv,window_width,window_height,window_width_initial_positi
on,window_height_initial_position,nume_afisat_ca_titlu_pe_fereastra);
```

E usor de intuit ce face fiecare argument.

- O functie de initializare care e apelata o singura data, inainte de a se incepe procesul de desenare. In aceasta functie va este indicat sa initializati toate obiectele cu care lucrati.

```
wd2d.init();
```

- O functie ce se executa o data per cadru. Programul apeleaza aceasta functie de fiecare data inainte de a afisa tot ce este atasat ori sistemului de coordonate original, denumit cs_basis, ori listei de sisteme de coordonate pe care le adaugati voi. Programul deseneaza in continuu: cadru 1, cadru 2, cadru 3,..... Framework-ul nu afiseaza decat obiectele atasate unui sistem de coordonate atasat WorldDrawer2d.

- O functie ce se executa de fiecare data cand este apasata o tasta.

In Transformation2d.cpp se gaseste codul pentru toate transformarile efectuate de program, in functii ca: translate(x,y,z), rotate(angle), scale(factor), etc.

Mai mult obiectele cat si sistemele de coordonate se pot translata roti etc deci stiind ca obiectele sunt atasabile si detasabile la sistemele de coordonate care la randul lor sunt atasabile si detasabile lumii 2D se pot efectua operatii relativ la mai multe spatii.

De exemplu:
- pot atasa lumii un sistem de coordonate pe care apoi sa il translatez si sa il rotesc.
- apoi pot atasa acestui sistem un obiect care sa se comporte relativ la sistem.

Atentie: datorita celor specificate anterior trebuie sa fiti atenti ce functii folositi, Obiect->translatie(5,5) poate avea alt rezultat decat SistemDeCoordonate->translatie(obiect,5,5)

Succes!

Responsabil:
Lucian Petrescu