11/11/2013

FACULTATEA DE AUTOMATICA SI CALCULATOARE

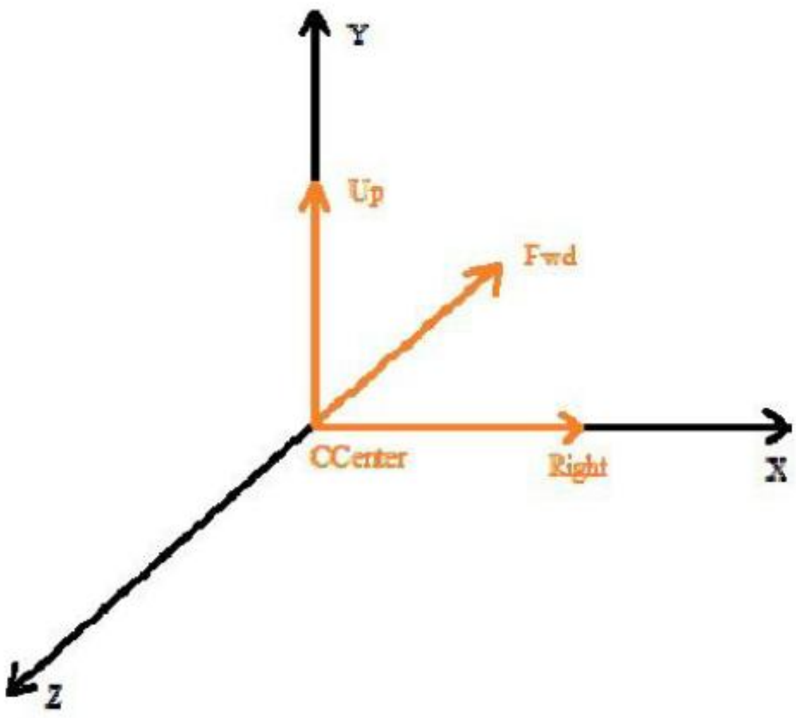
ELEMENTE DE GRAFICA PE CALCULATOR

Laborator 6

CAMERA

Sisteme de coordonate

O camera reprezinta o metoda prin care se poate specifica matricea de vizualizare a scenei intr-un mod ce pastreaza continuitate (nu ca o serie de vederi clar distincte). O camera este intotdeauna definita de 3 vectori: fata(directia de privire), dreapta si sus. Este necesar ca vectorii sa fie perpendiculari intre ei, pentru a pastra un spatiu euclidian. Acesti vectori, numiti axe, sunt usor de observat in figura de mai jos.



Astfel cand se apeleaza functia **glm::LookAt()** primele trei coordonate vor fi pozitia curenta a observatorului (CCenter), urmatorul set de coordonate vor reprezenta un punct de pe semidreapta formata de CCenter si Forward (de exemplu Ccenter + Fwd), iar ultimii trei parametri ai functiei **glm::LookAt()** sunt chiar componentele vectorului Up. De exemplu daca vrem sa ne situam pe axa Z si sa ne uitam pe –Z, putem face urmatorul apel de functii: **glm::LookAt**(0,0,3, 0,0,0, 0,1,0), adica, din pozitia (0,0,3) ma uit catre (0,0,0), si vectorul sus coincide cu axa Oy. Acum ca stim cum sa ne definim o camera static se pune problema mutarii camerei. Ce este vital de mentionat este ca **<u>toate operatiile sunt vectoriale si toate operatiile sunt relative la cadrul curent.</u>**

Pentru a implementa o camera trebuie sa putem reprezinta doua concepte:

- Pozitie
- Orientare

Amandoua pot fi reprezentate printr-o varietate de metode. Pozitia poate fi reprezentata in coordonate lume sau in coordonate relatative la a alt sistem de coordonate. In acelasi timp poate fi reprezentata in coordonate euleriene, polare, cilindrice, etc.

Orientarile la randul lor au un numar mare de implementari: unghiuri Euler (ordinea conteaza!), matrici de rotatie, quaternioni.

Camera First Person

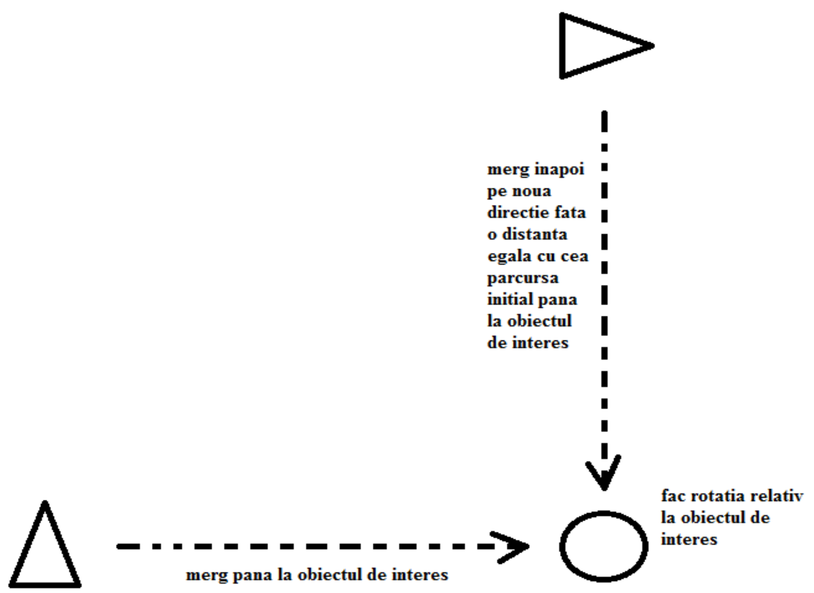
Problema principala in aceasta discutie o reprezinta tipul miscarii camerei. Intr-o camera de tip First Person rotatiile se fac pastrand observatorul pe loc si modificand pozitia in care se priveste. Astfel, daca dorim sa mutam observatorul pe oricare din axe va trebui sa calculam pozitia sa viitoare relativ la pozitia curenta si la orientarea curenta.
De exemplu translatia se poate rezolva in modul urmator:

Translatie in fata: **New_Observator = Observator + normalizare(Forward)*distanta**
Translatie la dreapta: **New_Observator = Observator + normalizare(Right)*distanta**
Translatie la sus: **New_Observator = Observator + normalizare(Up)*distanta**
Normalizarea este necesare pentru a obtine directia axei. In mod normal Forward, Right si Up sunt mentinuti normalizati.

Daca dorim sa facem o rotatie a camerei la dreapta(deci fata de axa OY):
New_Forward = Forward *cos(factor) + Right*sin(factor)
New_Right = Forward x Up
Adica calculam noua directia fata si apoi pastram corectitudinea sistemului de axe prin ajustarea vectorului dreapta pe o directie perpendiculara pe directia noii directii fata. Vectorul Up nu se schimba pentru ca e invariant la rotatie pe Oy relativ la cadrul curent.

Camera Third Person

In camera de tip Third Person observatorul se muta in jurul unui obiect de interes, ce reprezinta intotdeauna centrul atentiei. Deci rotatiile se fac intr-un mod diferit: Daca dorim sa facem o **rotatie** a camerei la dreapta relativ la obiectul de interes(deci fata de axa OY) vom proceda astfel: vom muta observatorul in locul obiectului de interes, vom face o rotatie simpla in **noul** spatiu iar apoi vom muta camera inapoi, pe directia **noului** vector de privire. Un exemplu este dat in imaginea de mai jos.



Putem descrie aceasta noua orientarea si pozitie prin ecuatiile:

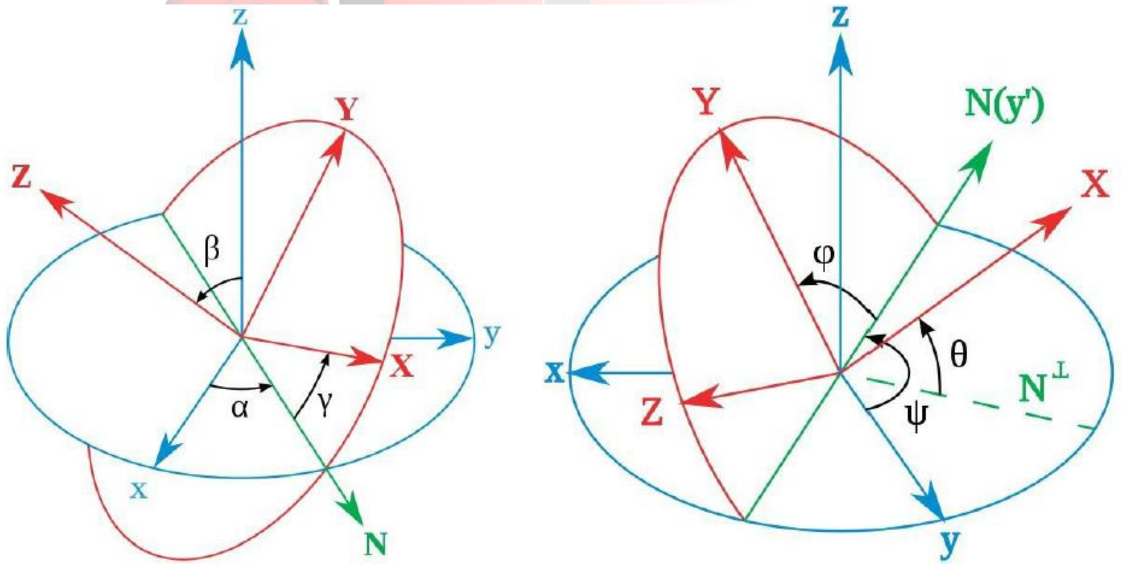
Distanta = distanta(Observator, Punct_de_Interes)
New_Forward = rotate(Forward, unghi)
New_Right = Forward x Up
New_Observator = Observator + Distanta*Forward – Distanta*New_Forward

Rotatiile relativ la celelalte axe sunt similare.

<u>Potentiale Probleme</u>

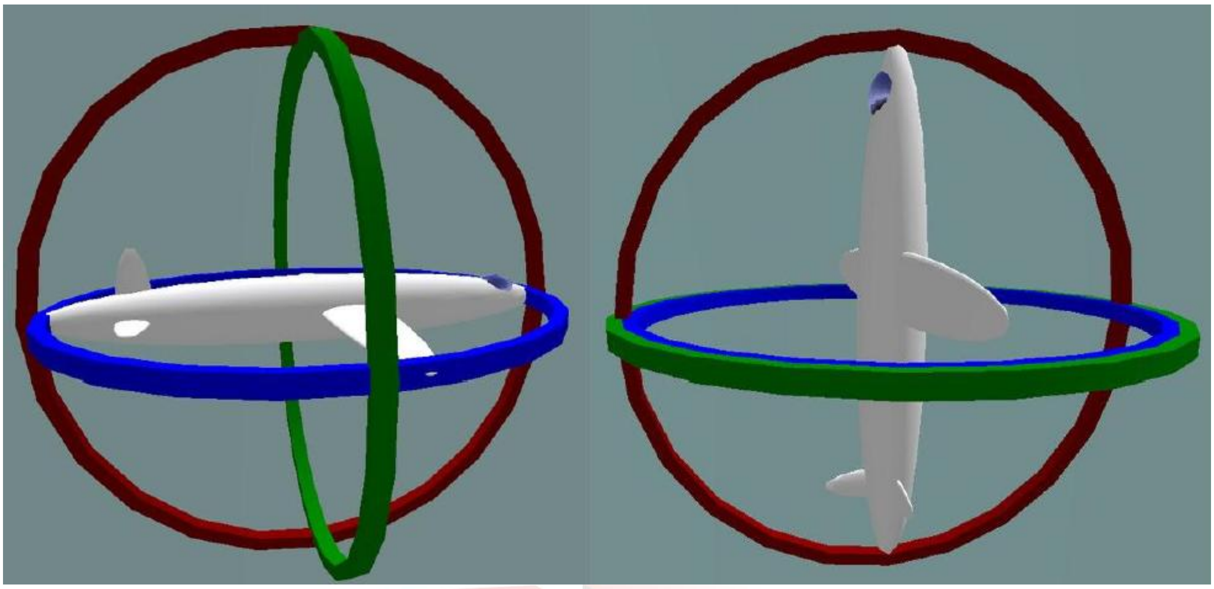
Fie ca folosim vectori sau ca folosim doar unghiuri de tip Euler este vital de precizat ca poate aparea o problema numita Gimbal Lock. Aceasta problema apare in momentul in care doua axe ale sistemului de axe se suprapun, privand sistemul de axe de un grad de libertate. Dar cum se poate intampla asa ceva cand sistemul este ortonormat? Raspunsul este unul simplu, **daca folosim o metoda de reprezentare a orientarii relativ la un singur <u>cadru global</u> intotdeauna vom ajunge la cazuri marginale. <u>INDIFERENT DACA FOLOSIM MATRICI SAU QUATERNIONI !</u> Operatiile sunt complet relative la cadrul curent (adica modificam si axele o data cu orientarea) ori cazurile particulare (singularitatile) trebuie tratate!**

Mai mult Intr-un spatiu eulerian conteaza ordinea operatiilor. Exista mai multe ordini de legare a rotatiilor fata de cele 3 axe, in imaginile urmatoare vor fi prezentate doar doua, xyz (intai rotatia fata de x, apoi fata de y si apoi fata de z) si zyx.



Dupa cum se poate observa in imagine, ordinea operatiile poate duce la rezultate complet diferite.

Un exemplu de gimbal lock:



Sa consideram ca toate rotatiile le exprimam ca unghiuri Euler relative la un sistem de axe global (care nu este definit relative la orientarea avionului).

Daca rotim avionul fata de axa albastra cu 90 se poate observa in figura de mai sus cum **efectele de rotatie (nu axele!)** date de axa verde si axa albastra ajung sa se suprapuna si vor ramane suprapuse. Astfel se pierde un grad de libertate. In functie de ordinea operatiilor se poate ajunge la situatia de Gimbal lock in diferite cazuri, pentru ca practic privim cele 3 cercuri ca pe o structura arborescenta de ordine, iar in functie de ordinea din structura ajungem la singularitati in situatii diferite.

Trivia: https://en.wikipedia.org/wiki/Gimbal_lock#Gimbal_lock_on_Apollo_11.