VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Projektinis darbas

Realaus objektų sukamųjų judesių perdavimas į virtualią aplinką

Atliko: 4 kurso 1 grupės studentas
Rytis Karpuška .....
(parašas)

Darbo vadovas:
R. Krasauskas, doc.....
(parašas)

Vilnius
2014

Įvadas

Sukamųjų judesių atvaizdavimas kompiuterio trimatėje erdvėje pasinaudojant populiariais pelės ir klaviatūros įeities įrenginiais turi atvaizduoti trimačius judesius per dvimatę ar net vienamtę (klaviatūros atveju) aplinką. Atkreipiant dėmesį į šią problemą, buvo bandoma sukurti įeities įrenginį, kuris neturi šio apribojimo, ir trimačius judesius atvaizduoja trimatėje erdvėje, panaudojant laisvai prieinamą techninę įrangą. Kaip įeities įranginys buvo pasirinktas išmanusis telefonas turintis inercinius giroskopo, bei akselerometro sensorius ir magnetometro sensorių.

1. Kompiliacija bei paleidimas

1.1. Išeities kodas ir jo struktūra

Visas išeities kodas yra prieinamas github sistemoje adresu: https://github.com/jauler/VU_MIF_tasks/tree/master/KompiuterineGrafika/projektas. Aiškumo dėlei kodas yra suskaidytas į tris dalis esančias trijuose kataloguose:

- *Andro_part* Kodas skirtas išmaniajam telefonui su android operacine sistema. Šis kodas yra atsakingas už matavimų atlikimą bei duomenų perdavimą į kompiuterį.
- *PC_part* Kodas skirtas kompiuteriui su „Linux“ operacine sistema. Šis kodas sukuria trimatį objektą openGL aplinkoje, surenka ir apdoroja duomenis gautus iš android išmaniojo telefono, bei atvaizduoja trimačius judesius.
- *paper* Latex kodas šiam dokumentui.

1.2. Išmaniajam telefonui skirto kodo kompiliacija, bei instaliacija

Išmanusiojo telefono kodas buvo rašomas su „android-studio“ integruotaja kūrimo aplinka (Versija Beta 0.8.6), todėl čia aprašomas būdas, kaip sukompiliuoti kodą naudojantis šia aplinka. Tai tikrai nėra vienintelis ir nėra geriausias būdas tai atlikti.

1. Žingsnis. Parsisiunčiame projektas direktoriją iš github sistemos

2. Žingsnis. Atsidarome android-studio aplinką. Importuojame projektą pasirinkdami meniu „File“, toliau „Import Project“, parenkame Andro_part katalogą.

3. Žingsnis. Kompiliuojame. Parenkame meniu „Build“, toliau „Generate signed APK“. Suvedame reikiamus duomenis rakto sugeneravimui. Palaukiame kol kompiliacija įvyks. Instaliuojamas APK failas yra sugeneruotas direktorijoje Andro_part/app.

1.3. Išmaniajam telefonui skirtos programėlės instaliacija be kompiliacijos

Patogumo dėlei github sistemoje yra įkeltas sukompiliuotas APK failas adresu https://github.com/jauler/VU_MIF_tasks/blob/master/KompiuterineGrafika/projektas/Andro_part/app/app-release.apk. Šį failą atsidarius telefone, jis bus parsiunčiamas ir instaliuojamas automatiškai.

1.4. Kompiuteriui skirto kodo kompiliacija

Prieš pradedant šios programos kompiliavimo darbus reikia pasirūpinti, kad būtų patenkintos priklausomybės. Programa skirta kompiuteriui turi šias priklausomybes:

- „POSIX threads“ biblioteka pthread
- „openGL priklausomybės
  - „GL“ biblioteka
  - „GLU“ biblioteka
  - „glut“ biblioteka
- „m“ matematikos biblioteka
- „libc“ ir kitos GCC kompiliatoriaus automatiškai įtraukiamos bibliotekos
- Kompiuteryje turi būti instaliuota programa „make“
- Kompiuteryje turi būti instaliuotas bei korektiškai sukonfigūruotas gcc kompiliatorius

Visos šios priklausomybės turi būti prieinamos GCC kompiliatoriui.

Patenkinus šias priklausomybe kompiliacija tampa labai paprasta. Kataloge PC_part reikia įvykdyti komandą:

```
make
```

1.5. Šio dokumento kodo kompiliacija

Šio dokumento kompiliavimas nedaro įtakos programos veikimui, todėl jo kompilacija aptarsime sutrumpintai.

1. Žingsnis. Pasirūpiname visomis „latex“ priklausomybėmis reikalaujamomis šio dokumento.

2. Žingsnis. Kompiliacija. įvykdome komandą:

make

2. Programos naudojimas

Programos dalys skirtos „Android“ ir „Linux“ operacinėms sistemoms turi būti paleidžiamos tam tikra tvarka.

2.1. „Linux“ operacinei sistemai skirto kodo paleidimas

„Linux“ programa yra duomenų priėmimo serveris, todėl ji turi būti paleista pirmiausiai.

Įvykdome failą „demo“ PC_part kataloge. Pagal nutylėjimą programa užkrauna kūbo modelį, bet github sistemoje jų įkelta yra daugiau:

- *cube.obj* - Kūbo modelis, užkrauanams pagal nutylėjimą
- *monkey.obj* - Beždžionės galvos modelis
- *human.obj* - Žmogaus modelis
- *sphere.obj* - Sferos modelis

Programai galima pateikti modelio failą pateikiant kelią iki jo kaip pirmąjį argumentą, pvz.:

```
./demo human.obj
```

Jeigu programa sėkmingai įsijungė, turime matyti langą su modeliu, bei standartinėje išeityje (eng.: „standard output“ arba „stdout“) yra išspausdinama eilutė:

```
Listening on 0.0.0.0 at 10001
```

Bei šiek tiek apibendrintos informacijos apie užkrautą modelį.

2.2. „Android“ operacinei sistemai skirto kodo paleidimas

Sėkmingam abiejų įrenginių komunikavimui būtina, kad jie būtų tame pačiame potinklyje, todėl prieš paleidžiant android programą, reikia įsitikinti, kad ši sąlyga yra patenkinama. Paprašiausiais būdas tai padaryti - abu įrenginius prijungti prie to pačio Wi-Fi tinklo arba sukurti Wi-Fi prieigos tašką mobiliajame telefone ir prijungti kompiuterį prie šio tinklo. Taip pat reikalingas geras tinklo atsakas (eng. „latency“ arba „ping“), todėl antrasis sujungimo būdas yra rekomenduojamas.

Atsidariusioje aplikacijoje rodomi du įvesties laukai, viename įvedame serverio (kompiuterio) IP adresą, kitame TCP porto numerį: „10001“.

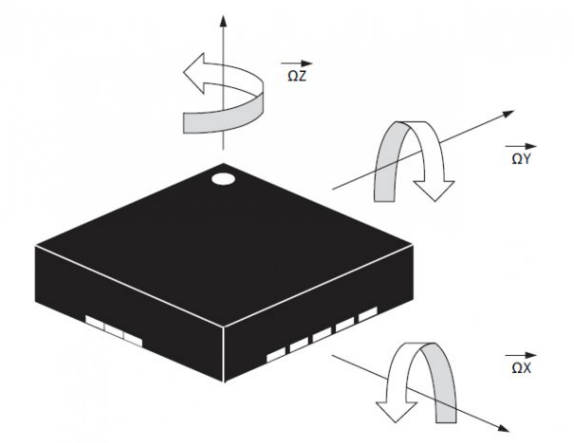
Pastebėjimas: pradinė telefono būsena yra naudojama kaip atskaitos taškas, tad intuityvumo dėlei patartina sulygiuoti telefoną su modeliu rodomu ekrane.

Toliau spaudžiame „Connect“, ir viską teisingai atlikus matysime, kad modelio kampinė pozicija atitinka išmaniojo telefono kampinę poziciją realiame pasaulyje.

3. Veikimo principas

3.1. Sensoriai

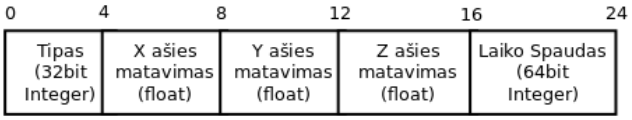
Šioje programoje naudojami MEMS giroskopo duomenys. Beveik kiekvienas išmanusis telefonas turi tokį giroskopą. Įprastai trimatis MEMS giroskopas susideda iš trijų vienmačių MEMS giroskopų erdviškai orientuotų taip, kad tarp visų jų susidarytų 90 laipsnių kampas. Iš to seka, kad trimačio MEMS giroskopo išėjime yra trys reikšmės sudarančios sukimo vektorių (eng.: „rotation vector“), kur vektoriaus kryptis nurodo ašį aplink kurią sukama, o jo ilgis - kokiu greičiu yra sukama.



1 pav. MEMS giroskopas, lenktos rodyklės atvaizduoja giroskopo išeities duomenis.

3.2. Duomenų perdavimas

Duomenų perdavimui naudojami berkeley TCP sockets tipo jungtis. Startuodama „Linux“ operacinės sistemos programa, atidaro serverio tipo TCP socket jungtį su porto numeriu 10001, ir laukia, kol bus atsiūstas prisijungimo prašymas. Leidžiamas nedaugiau kaip vienas klientas vienu metu. Prisijungęs prie serverio „Android“ išmanusis telefonas pradeda siūsti matavimų duomenis. Matavimo duomenų paketas siunčiamas iškarto po kiekvieno atlikto matavimo. Paketo formatas pavaizduotas pav. 2



2 pav. Matavimo duomenų paketo formatas. Skaičiai parodo atstumą baitais nuo paketo pradžios.

Laukas „tipas“ gali būti dviejų reiškmių:

- *0* - Akselerometro duomenys (šiuo metu jie nėra naudojami „Linux“ programoje)
- *1* - Giroskopo duomenys

Laukai X, Y, Z ašims yra Akselerometro arba giroskopo matavimai. Kiekvienas skirtingas lauko „tipas“ variantas išreiškia skirtingus matavimo vienetus:

- Akselerometro - akseleracija visomis trimis ašimis išreikšta žemės gravitacijos lauko vienetais (g, $1g \approx 9{,}8m/s^2$)
- Giroskopo - kampinis greitis visomis trimis ašimis išreikštas radianais per sekundę (rad/s)

Laiko spaudas naudojamas matavimų integracijai laiko atžvilgiu. Integruojant reikia tiksliai žinoti kiek lako praėjo nuo vieno matavimo iki kito. Verta paminėti, kad negalima pasitikėti duomenių gavimo momento laiku, nes interneto protokolai įveda didelius iškraipymus.

Laiko spaudas parodo kada buvo padarytas matavimas nanosekundėmis ($1ns = 10^{-9}s$) tam tikro (nežinomo, bet fiksuoto) taško laike atžvilgiu. Šiuo atveju žinoti kieno atžvilgiu yra daromas laiko spaudas nėra reikalo, nes programos algoritmas suformuotas taip, kad dirbama tik su laiko skirtumais tarp matavimų. Verta pastebėti, kad šioje vietoje negalima imti įprasto laiko (angliškai vadinamo „wall clock“), nes jis yra koreguojamas ir įvykus koregavimui, laiko skirtumas tarp matavimų nebeatspindės realaus prabėgusio laiko. Įprastam mobiliajame telefone šie koregavimai įvyksta palyginus dažnai, todėl būtinai reikia operacinės sistemos užklausti nekoreguojamo laiko.

3.3. Duomenų apdorojimas

Šio darbo tikslas yra atvaizduoti realaus pasaulio trimačius sukamuosius judesius virtulioje aplinkoje. Mūsų įeities duomenys yra sukamasis vektorius lokalioje (išmaniojo telefono) koordinačių sistemoje, tad tai reikia atvaizduoti į openGL modelio aplinką. OpenGL aplinka pateikia funkcija glRotate, kuri leidžia pasukti objektą. Jeigu į objekto matricą nėra užkraunama identity matrica, tuomet ši funkcija atvaizduoja pasukimą pagal krypties vektorių ir kampo didumą lokalioje to objekto koordinačių sistemoje. Šiame algoritme, gavus naują duomenų paketą, gautas sukimo vektorius yra padauginamas iš konstantos normalizuojančios sukimo greičius ir išsaugomas į kampinio greičio registrą. Kiekvieno kadro paišymo metu, objektas yra pasukamas tuo sukimo vektoriumi (žinoma konvertuojant į glRotate funkcijos reikalaujamą sukimo vektoriaus su atskiru kampu formatą). Tokie veiksmai atitinka sukimo vektoriaus integraciją laiko atžvilgiu, o tai pagal fiziką išduoda kampinę poziciją.

4. Esamos problemos ir galimi patobulinimai

4.1. Dreifavimas

Dabartinėje programos versijoje naudojami tik giroskopo duomenys, integruojami laiko atžvilgiu. Tai sukelia vadinamą dreifavimo (eng.: „Drift“) problemą. Dėl šios problemos, matavimo paklaidos laikui bėgant sumuojasi, ir nukrypsta nuo realios padėties.

Galimas sprendimas būtų panaudoti sensorių duomenų sintezę (eng.: „sensor fusion“) ir įtraukti magnetometro, bei akselerometro matavimus į kampinės padėties matavimus. Duomenų apjungimui galėtų būti panaudotas kalmano filtras, kuris leistų absoliučiai išvengti dreifavimo problemos.

4.2. Žemas matavimų dažnis

Mobilusis telefonas matavimus atlieka palyginus žemu dažniu, todėl greitai pakračius telefoną dreifavimo problema įveda katastrofiškas paklaidas. Iš dalies šią problemą galėtų spręsti kalmano filtras ir „sensor fusion“ algoritmas apibūdintas prieštai buvusioje posekcijoje, bet speciali techninė įranga su matavimo dažniu siekančiu 500-1000Hz leistų atlikti itin tikslius kampinės pozicijos matavimus.

5. Išvados

Šiame darbe yra pasiūlomas ir išbandomas integracinis metodas laisvai prienamų, išmaniuosiuose telefonuose montuojamų, MEMS giroskopų panaudojimui trimačių objektų manipuliacijai. Dėja rezultatai rodo, kad vien integruojami giroskopo duomenys suteikia stiprų dreifavimą, ir patogiai trimačių objektų manipuliacijai būtina kompensuoti dreifavimą.

[AAJ+01] - *Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magnetic Sensors*, http://franciscoraulortega.com/pubs/Algo3DFusionsMems.pdf

[SSF+11] - *A sensor fusion algorithm for an integrated angular position estimation with inertial measurement units*, http://www.date-conference.com/proceedings/PAPERS/2011/DATE11/PDFFILES/IP1_06.PDF

[MS11] - *Modeling, Design and Experimental Study for a Quadcopter System Construction*, http://brage.bibsys.no/xmlui/bitstream/id/86811/uiareport.pdf