Osnove virtualnih okruženja

Laboratorijske vježbe

Vježba 2
Graf scene: VRML

FER – ZTE – Igor S. Pandžić
Suradnja na pripremi vježbe:
Tomislav Košutić
Mišel Mamula

1. Uvod

Graf scene je struktura u koju se sprema virtualna scena na organiziran i strukturiran način. Detaljnije o grafu scene može se pročitati u skripti s predavanja (*Graf scene i geom. transformacije*). VRML (*Virtual Reality Modeling Language*) je jezik za tekstualni opis 3D scene a zasniva se na grafu scene. VRML se može za početak shvatiti kao HTML za 3D (iako je sintaksa bitno različita). Osim kompletne 3D scene, u VRML-u je moguće definirati i izvore svjetla, položaj kamere, teksture, animaciju, interakciju s korisnikom itd. Sve što je potrebno je editor teksta (npr. Notepad).

Cilj vježbe je upoznavanje sa sintaksom VRML jezika kroz ove upute i službenu dokumentaciju, a rješavanjem konkretnih zadataka dobit će se bolji uvid u graf scene, geometrijske transformacije i osnove animacije.

2. Alati potrebni za izvođenje vježbe

Za izvođenje vježbe potreban je Cortona VRML plugin za web preglednik (Internet explorer ili neki drugi), editor teksta (Notepad itd.) i te za npredni zadatak alat za pisanje Java programa (npr. Borland Jbuilder, Eclipse…).

2.1 Izvođenje vježbe na vlastitom računalu

Cortona plugin je dostupan na web stranicama predmeta; prilikom instalacije plugin će se integrirati u Internet explorer, te će se pokretati automatski kada naiđe na VRML sadržaj. Plugin je intuitivan za korištenje i ima jednostavne upute (potrebno je kliknuti desnom tipkom miša na scenu i odabrati Help -> User's guide).

3. Teorijska podloga

3.1 Osnove VRML-a

Sastavni dijelovi VRML-a su **čvorovi (node)**. VRML datoteka (.wrl) je niz čvorova koji opisuju 3D scenu (objekte, njihov smještaj, animaciju...). Svaki čvor posjeduje **polja (field)** koja ga definiraju.

Osnovni građevni čvor je Shape, pomoću kojeg se definiraju svi 3D predmeti, od najjednostavnijih do najsloženijih:

PRIMJER: čvor Shape

```
#VRML V2.0 utf8
# A Cylinder
Shape {
    appearance Appearance {
        material Material { }
    }
    geometry Cylinder {
        height 2.0
        radius 1.5
    }
}
```

Ovaj primjer je ujedno i potpuno funkcionalan VRML dokument. S web stranica predmeta možete skinuti arhivu VRMLprimjeri.zip i pogledati primjer u pretraživaču (datoteka *cilindar.wrl* u arhivi). Linije koje počinju s # su komentari, svaka VRML datoteka počinje s linijom #VRML V2.0 utf8.

Shape, Appearance, Material i Cylinder su čvorovi (počinju velikim slovom) a appearance, material... su polja (počinju malim slovom). Za čvor Shape osnovna polja su:

- appearance (definira izgled predmeta preko čvora Material)
- geometry - definira oblik predmeta, koriste se čvorovi osnovnih oblika (Box, Sphere, Cylinder, Cone) ili za složenije oblike IndexedFaceSet

Specifikacija svakog čvora sastoji se od popisa njegovih polja - field.

Svako polje ima:

- tip (npr. *SFFloat*) - sastoji se od prefiksa (SF – single value, MF – multi value) i tipa podatka (*Float*, *Bool*, *String*, *Color*, *Rotation*...)
- naziv (*height*, *bottomRadius* ...)
- početnu (default) vrijednost

Svaki čvor je potpuno definiran u samoj specifikaciji VRML-a, dakle sva polja su postavljena na neke početne vrijednosti. Specifikacija za svaki čvor može se naći u VRML97 dokumentaciji (nalazi se na web stranici predmeta). Ono što mi radimo kad pišemo u VRML-u je mijenjanje tih početnih vrijednosti na nama potrebne vrijednosti:

PRIMJER: čvor Cone (stožac)

VRML specifikacija za Cone izgleda ovako:

```
Cone {
  field     SFFloat   bottomRadius 1
  field     SFFloat   height       2
  field     SFBool    side         TRUE
  field     SFBool    bottom       TRUE
}
```

Ako se u VRML datoteci napiše instanca čvora Cone ovako:

```
Cone{
}
```

iscrtat će se stožac s radijusom baze 1, visinom 2, i vidljivom čitavom površinom.

Ako se napiše:

```
Cone{
  height 5
  bottom FALSE
}
```

iscrtava se stožac s visinom 5, bez dna (prozirno dno). Sve ostale vrijednosti ostaju nepromjenjene (default).

3.2 Čvorovi za grupiranje i transformacije,

Jedan od najbitnijih čvorova u izgradnji scene je grupni čvor Transform, pa će ovdje biti detaljnije objašnjen. Kompletna specifikacija za ostale čvorove nalazi se u VRML dokumentaciji.

Osnovni oblik:

```
Transform {
    translation X Y Z
    rotation X Y Z angle
    scale X Y Z
    children  [ . . . ]
}
```

children [ ] je osnovno polje svih grupnih čvorova, unutar polja nalaze se svi čvorovi koje grupni čvor sadrži. To može biti bilo koji čvor (npr drugi Transform čvorovi). U ovom primjeru čvor Transform je parent (roditelj) svim čvorovima u polju children (djeca).

- *translation:* transformacija translacije, unosi se *float* vrijednost (jedinica je metar)
  Npr. `translation 1.0 0.0 0.0` pomiče sustav za jedan metar po X osi

- *rotation:* transformacija rotacije, bira se po kojoj osi se rotira
  angle (kut) su unosi u **radijanima** rad = (deg/180)*π
  Npr. `rotate 1.0 0.0 0.0 0.52` rotira za 30° oko X-osi:

- *scale:* mijenja proporcije predmeta
  Npr. `scale 0.5 0.5 0.5` daje upola manji predmet

Bitno: sve transformacije obavljaju se relativne na čvor roditelj (ovo je bilo objašnjeno na predavanjima). Npr. ako čvor radi translaciju za 1m u smjeru X-osi, i sadrži drugi Transform čvor (u children[ ]) koji radi translaciju za 2m u smjeru X-osi, djeca tog drugog čvora pomaknuta su za 3m u odnosu na globalni koordinatni sustav.

3.3 Uvod u animaciju

Pomoću VRML-a moguće je jednostavno animirati predmete tako da se pomiču, okreću, mijenjaju veličinu ili slično.

Općenito, animacija se odvija u nekom vremenskom intervalu. Napredovanje animacije u vremenskom intervalu u VRML-u opisuje se modelom protoka podataka (*data flow model*). Model opisuje komunikaciju između čvorova u VRML-u. Čvorovi izmjenjuju podatke putem ruta (ili grana). Čvor koji šalje podatke generira događaj slanja podatka. Podatak putuje rutom i dolazi do slijedećeg čvora koji generira događaj primanja podatka. Čvor može podatak obraditi i/ili ga prosljediti slijedećem čvoru. Čvorovi imaju svojstvo primanja podataka s više strana i slanja podataka na više odredišta. Podatak može biti vektor (npr koordinate translacije) ili vrijeme (kada je podatak poslan).

Npr. da bi smo zaokrenuli predmet spojimo čvor koji šalje koordinate za rotaciju s *rotation* poljem u *Transform* čvoru.

Za opis animacije potrebne su nam slijedeće komponente:

- čvor koji šalje podatke tj. generira događaje (mora biti imenovan s DEF);
- čvor koji prima podatke (također mora biti imenovan s DEF);
- ruta koja povezuje ova dva čvora.

Svaki čvor sastoji se od polja koja mogu biti četiri različita tipa:

- *field* – može se postaviti samo kod inicijalizacije;
- *eventIn* – postavlja tok podatka;
- *eventOut* – ne može se postaviti, moguće je samo čitati podatak ovog tipa;
- *exposedField* – tip koji je ujedno *field*, *eventIn* i *eventOut*.
  (npr. ako čvor ima exposedField s nazivom *startTime*, to znači da ima field *startTime*, eventIn *set_ startTime* i eventOut *startTime_changed*)

Npr.: *Transform* čvor ima ova *eventIn* polja: *set_translation*, *set_rotation* i *set_scale*; *PositionInterpolator* čvor ima *eventOut* polje: *value_changed* (šalje koordinate translacije).

EventIn i *eventOut* polja za ostale čvorove možete naći u VRML dokumentaciji. Na svim poljima se primjenjuje konvencija: sva *eventIn* polja su oblika *set_xxx*, *eventOut* polja *xxx_changed*.

Ruta između dva čvora definira se naredbom:

ROUTE MySender.*rotation_changed* **TO** MyReceiver.*set_rotation*

Ključne riječi ROUTE i TO moraju biti napisane velikim slovima.

Animiranje transformacija

Za opis animacije u VRML-u uz model toka podataka potrebna nam je i kontrola vremena. Kada animacija započinje, kada završava i koliko brzo treba ići. Čvor ***TimeSensor*** kontrolira početak i kraj animacije, te generira vremenske događaje (sličan je običnoj štoperici). Može generirati događaj u zadano vrijeme (a*bsolute time event*) ili u određenim vremenskim intervalima (*fractional time event*). Događaji u vremenskim intervalima generiraju brojeve od 0.0 (na početku animacije) do 1.0 (na kraju animacije). Vrijeme (u sekundama) koje prođe između 0.0 i 1.0 kontrlira se sa intervalom ciklusa (*cycle interval*).

PRIMJER:

```
TimeSensor {
    cycleInterval 1.0     #vrijeme ciklusa 1 sekunda
    loop FALSE
    startTime 0.0
    stopTime 0.0
}
```

Načini izvođenja ciklusa:

- *loop* TRUE i *stopTime* > *startTime* - izvršavanje ciklusa do *stopTime*;

- *loop* FALSE i *stopTime* <= *startTime* – izvršavanje samo jednog ciklusa;
- *loop* TRUE i *stopTime* <= *startTime* – beskonačno ponavljanje.

eventIn polja:

- *set_startTime* – postavlja početak rada timera;
- *set_stopTime* – zaustavlja rad timera.

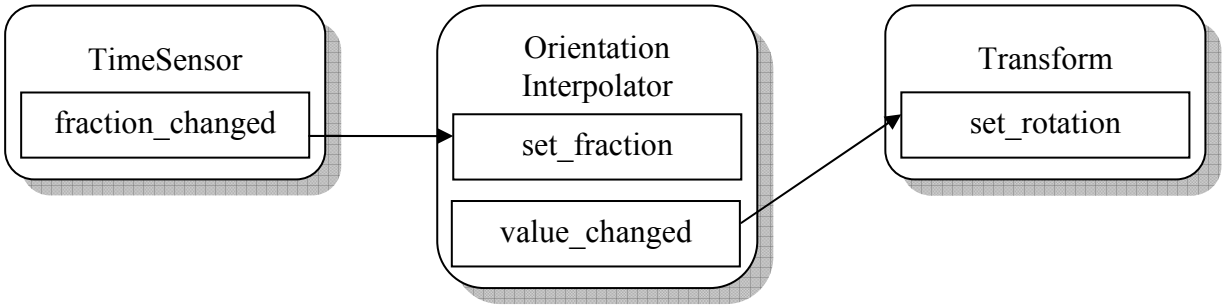
eventOut polja:

- *isActive* – vraća TRUE kada timer počne s radom, FALSE kada se timer zaustavi;
- *time* – vraća trenutno (apsolutno) vrijeme;
- *fraction_changed* – vraća vrijednosti između 0.0 i 1.0 za vrijeme ciklusa timera.

PRIMJER: Animacija rotacije kocke (datoteka *animiranaKocka.wrl* u arhivi VRMLprimjeri.zip)

```
DEF TRANS Transform {
     children [
          Shape {
               appearance Appearance {
                    material Material {
                         diffuseColor 0.84 0.86 0.042
                    }
               }
               geometry Box {
                    size 2.0 2.0 2.0
               }
          }
     ]
}
DEF TIMER TimeSensor {
     loop TRUE
     cycleInterval 2.0
}
DEF ROTATOR OrientationInterpolator {
     key [ 0, 0.5, 1 ]
     keyValue [ 0 1 0 0, 0 1 0 3.141, 0 1 0 6.282 ]
}
ROUTE TIMER.fraction_changed TO ROTATOR.set_fraction
ROUTE ROTATOR.value_changed TO TRANS.set_rotation
```

Graf modela toka podataka za čvorove iz navedenog primjera:



Čvor ***OrientationInterlopator*** pretvara ulazno vrijeme (*key*) u izlazne parametre rotacije (*keyValue*). Polje *key* predstavlja niz ulaznih vrijednosti, u našem primjeru tri vremenska intervala. Polje *keyValue* predstavlja niz odgovarajućih izlaznih vrijednosti, tj za svaki od tri vremenska intervala po jedan parametar rotacije (oko y-osi za 360º). U slučaju da je ulazno vrijeme između dva vremenska intervala, interpolator izračunava srednju vrijednost (interpolira) parametra rotacije.

eventIn polje: *set_fraction* – postavlja trenutni vremenski interval

eventOut polje: *value_changed* – vraća izlaznu vrijednost svaki put kada se postavi novi vremenski interval

U našem primjeru izlazne vrijednosti interlopatora su parametri rotacije, a općenito mogu biti bilo što (koordinate, boja). Za opis ostalih interpolatora (*PositionInterpolator*, C*olorInterpolator*, *ScalarInterpolator*) proučiti te čvorove u VRML dokumentaciji.

3.4 EAI (External Authoring Interface)

VRML scenu moguće je kontrolirati iz vanjske aplikacije (npr. Java appleta). Za sada su podržani jezici JavaScript i Java. Podrška za EAI ugrađena je u sam VRML standard, a podržavaju ga svi bitniji VRML pluginovi (i Cortona s kojom se radi na ovim vježbama).

Za kontrolu scene pomoću Jave uz Cortonu dolaze već definirane klase. Korištenjem ovih klasa u aplikaciji ili appletu, možemo direktno utjecati na scenu na bilo koji način definiran u VRML-u (dodavati i brisati predmete, mjenjati im izgled, dodati im animaciju...)

Za EAI potrebne su tri stvari (uz svaku je dan kratki primjer):

1. **VRML scena** (.wrl) s najmanje jednim definiranim i imenovanim (DEF) čvorom:

```
#VRML V2.0 utf8
DEF Root Transform {}
```

   U sceni je samo jedan prazan čvor, s nazivom Root.

2. **Java applet:**

```
import VRML.external.*
import VRML.external.field.*
import VRML.external.exception.*
import java.awt.*;
import java.applet.*;

public class SuncevSustavEAI extends Applet {
```

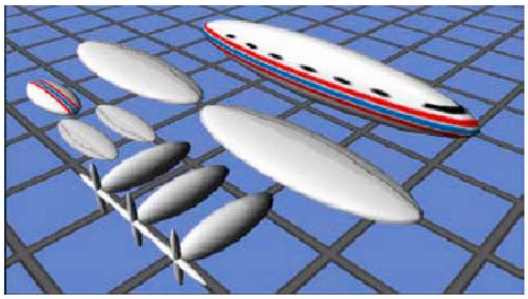
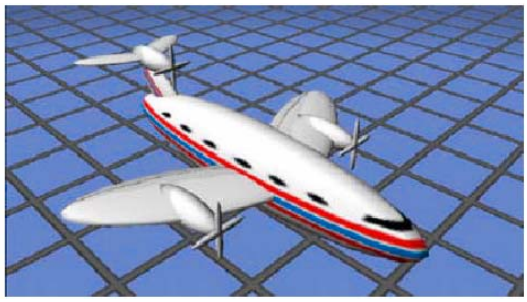
```
  Browser browser = null;
  Node node = null;

  Public void start() {
    browser = Browser.getBrowser(this);
    node = browser.getNode("Root");
    }
}
```

Metoda getBrowser() vraća vezu na VRML preglednik, a getNode() čita čvor s nazivom root, nakon čega su nam na raspolaganju metode za manipulaciju čvorovima iz klase Node i Field. Arhiva sa detaljnijim opisom klasa nalazi se na stranicama (Classes.zip). Dodatni primjer nalazi se u VRMLprimjeri.zip, u EAI_primjer.txt, gdje je dan source scene i jednostavnog appleta koji omogućava promjenu boje objekta pritiskom na button.

3. **HTML dokument**:

```
<html>
    <embed src="SuncevSustavEAI.wrl" height=500 width=500>
    <applet code="SuncevSustavEAI.class" height=300 width=300>
    </applet>
</html>
```

4. Opis zadatka

1. ZADATAK:

Od kugle kao početnog oblika potrebno je geometrijskim transformacijama napraviti osnovne dijelove aviona: trup, krilo, rep, motor s propelerom. Zatim korištenjem transformacija od postojećih osnovnih dijelova treba sastaviti cijeli avion.

Na jedno od krila treba dodati vlastite inicijale.

2. ZADATAK:

Potrebno je napraviti simulaciju (animaciju) sunčevog sustava koristeći animiranje transformacija u VRML-u. Simulirajte gibanje Zemlje oko Sunca i gibanje Mjeseca oko Zemlje.

3. ZADATAK:

U 2. zadatku zanemarene su rotacije zemlje i mjeseca oko vlastite osi – ove rotacije se ionako ne bi vidjele jer su tijela simulirana jednobojnim kuglama. Dodati teksture mjeseca i zemlje, te ispravne rotacije oko vlastitih osi.

5. Upute za rad

1. ZADATAK:

- Pogledati VRML datoteku *avion.wrl* (u arhivi VRMLprimjeri.zip) u kojoj je kao primjer već napravljen trup i rep;
- Modelirati ostale dijelove aviona u istoj datoteci – krilo i motor s propelerom;
- Transformacijama (čvor *Transform*) od već definiranih dijelova treba dovršiti avion (koristiti naredbe DEF i USE**);**
- Dimenzije aviona su proizvoljne, bitno je da proporcije otprilike odgovaraju onima na slici;
- Dodati teksturu (sliku) na trup aviona (koristiti sliku *trup.jpg*);
- Korištenjem bilo kojeg VRML čvora dodati vlastite inicijale na jedno od krila aviona.

2. ZADATAK:

- Sunce, Zemlju i Mjesec možemo aproksimirati kuglama (čvor Sphere) različitih boja: Sunce žutom (0.84 0.86 0.042), Zemlju plavom (0.0 0.5 0.75) i Mjesec sivom (0.75 0.75 0.75) bojom. Polumjere planeta i putanja možemo također pojednostavniti (u stvarnosti je Sunce oko 100 puta veće od Zemlje):

  → polumjer Sunca: 6.96
  → polumjer Zemlje: 0.63
  → polumjer Mjeseca: 0.17
  → polumjer putanje Zemlje oko Sunca: 14.96
  → polumjer putanje Mjeseca oko Zemlje: 1.0

- Vrijeme kruženja Zemlje oko Sunca postaviti na 60 sekundi, dok je vrijeme kruženje Mjeseca oko Zemlje 13 puta manje (4.61 sekundu).
- Koristiti *OrientationInterpolator* kao u primjeru animiranaKocka.wrl.
- Prilikom izrade simulacije dobro je koristiti čvor *Viewpoint* (pogledati u VRML dokumentaciji kako se koristi) koji pozicionira pogled korisnika u virtualnu scenu. Njime se smanjuje mogućnost da se planeti "zagube" u sceni.

3. ZADATAK:

- Dodati teksture na planete kako bi se vidjele njihove vlastite rotacije. Teksture Zemlje, Mjeseca i Sunca potrebno je pronaci na Internetu.
- Trajanje rotacije Zemlje oko vlastite osi postaviti na 0.164, a trajanje rotacije Sunca oko vlastite osi postaviti na 4.1 (Suncu treba 25 dana za 1 krug oko vlastite osi, a buduci da 1 dan u nasem modeliranom sustavu traje 0.164 s, slijedi da je 25 x 0.164 = 4.1).

- Olakotna okolnost je sto u stvarnosti Mjesec nema rotaciju oko vlastite osi (cijelo vrijeme vidimo istu stranu Mjeseca) pa je niti u zadatku nije potrebno dodati.

6. Pomoćni materijali za izradu vježbe

Primjeri navedeni u ovim uputama i svi dodatni materijali potrebni za izradu vježbe nalaze se u arhivi VRMLprimjeri.zip koja se nalazi na web stranicama predmeta.

Detaljna specifikacija svakog čvora nalazi se u VRML 97 dokumentaciji dostupnoj na web stranicama predmeta. Dodatno se može proučiti i VRML tutorial (na stranicama predmeta) koji sadrži još detaljnija objašnjenja i primjere.

Opis Java klasa za EAI (potrebnih za 1. napredni zadatak) nalazi se u Classes.zip (index.html). Same klase (.class) dolaze sa instalacijom Cortone i nalaze se (nakon instalacije) u: C:\Program Files\Common Files\ParallelGraphics\Cortona\corteai.zip

7. Predavanje rezultata vježbe

Rezultati vježbe se predaju zapakirani u arhivu ***OVO-V2- Rezultati-<ImePrezime>.zip*** koja treba sadržavati:

- Izvještaj o izvođenju vježbe s opisom rješenja svakog zadatka.
- Rješenje 1. zadatka: datoteka *avion.wrl*
- Rješenje 2. i 3. zadatka: datoteka *SuncevSustav.wrl*
- Dodatne datoteke potrebne za ispravnu funkciju svih rješenja (npr. teksture).

VRML i Java kod potrebno je komentirati (dobro objasniti što se radi u svakom dijelu dokumenta).

Navedena arhiva treba biti predana korištenjem *Web aplikacije za predaju vježbi* dostupne preko web stranica predmeta.

Napomena: Rezultati se šalju isključivo preko gore navedene aplikacije. U slučaju problema, javiti se email-om na adresu ovo@tel.fer.hr. Sačuvajte kopiju poslanih rezultata.

8. Napredni zadaci (ovaj dio nije obavezan)

1. Korištenjem EAI mehanizma potrebno je napisati jednostavni Java applet koji ima klizač (slider) kojim je moguće mjenjati brzinu simulacije (tj. rotacije Zemlje i Mjeseca, te rotacija Zemlje i sunca oko vlastite osi). Iz zadatka 3 može se iskoristiti geometrija scene, a algoritam animacije, tj. promjene kuta rotacije u vremenu, potrebno je napisati u appletu.

2. Napraviti model biljarskog stola s tri kugle (jedna bijela i dvije crvene) i simulaciju igre uz detekciju sudara i fizikalnu simulaciju odbijanja kugli.

Upute za 1. napredni zadatak:

- Za model Sunčevog sustava koristite VRML scenu iz trećeg zadatka ali bez mehanizma animacije.

- Proučiti klase u paketu vrml.external (i vrml.external.exception, vrml.external.field) i EAI_primjer.txt; uočiti kako se iz appleta mijenjaju vrijednosti unutar čvora.

- Applet SuncevSustavEAI.java vec sadrži kod za dohvat VRML scene, potrebno napisati jednostavni algoritam animacije, tj. promjene kuteva rotacije Zemlje i Mjeseca u odnosu na vrijeme i zadanu brzinu simulacije. Animaciju implementirajte u posebnoj niti (Thread) appleta.

- EAI mehanizmom se kutevi rotacije preslikavaju na polje rotacije u odgovarajućim Transform cvorovima u sceni (koristiti funkcije/klase getNode, getEventIn, EventInSFRotation, setValue).

- U appletu implementirati klizač za promjenu brzine simulacije.

- Kreirati HTML dokument koji povezuje scenu i applet.

- Ako koristite JBuilder, na žalost applet nije moguce testirati iz Jbuilder razvojne okoline zbog VRML scene u html dokumentu. Kompilirajte applet i testirajte ga putem HTML dokumenta u web pregledniku.

- *Napomena:* moguće je da će applet raditi samo u Internet Exploreru s JVM-om.

- Dobro komentirajte bitne dijelove koda, te pošaljite rješenje (datoteke *SuncevSustavEAI{.html,.java,.class}*)!