ELEC-C8203 Automaatiojärjestelmät 2

Automaation tietotekniikka labra 1

Oppimistavoitteet

- Johdanto Java kieleen, olettaen että osaat Pythonin perusteet (mikä on esitietovaatimus)
- Johdanto olio-ohjelmointiin käytännössä –asia on lyhyesti esitelty Python kurssilla, mutta emme oleta esitietovaatimuksena varsinaista olio-ohjelmointitaitoa
- Sovellusohjelmointirajapinnan (API Application Programming Interface) käyttö
- Johdanto 3D virtuaaliympäristöihin
- Yllämainittujen asioiden relevanssi automaatioinsinöörille

Tehtävänanto

Nykyään kaikki 3D Java ympäristöt ovat pelimoottoreita, koska niiden kehitys menee eteenpäin niin nopeasti, että muut hankkeet ovat luovuttaneet. Pelimoottoreissa pyritään realististen 3D virtuaaliympäristöjen luomiseksi ja niiden tapa mallintaa 3D maailmaa on sama kuin automaatioinsinööreille relevanteissa robotiikka, tehdassimulointi ja CAD (mekaniikkasuunnittelu) työkaluissa, eli opitut asiat ovat relevantteja myös näissä ympäristöissä. Tällä kurssilla käytetään jMonkeyEngine moottoria. Avaa kurssin MyCourses sivulta ’Eclipse Hello World’ dokumentti ja seuraa siinä annettuja ohjeita. Kun olet saanut Hello Worldin ajettua, niin laitetaan pystyyn jMonkeyEnginen perusprojekti, joka on sininen kuutio.

Luo paketti mygame:

- Klikkaa oikealla src/main/java
- Valitse New/package
- Anna nimeksi mygame ja klikkaa Finish

Kaikki samassa paketissa olevat lähdekooditiedostot ’näkevät’ toisensa.

Luodaan Main luokka:

- Klikkaa oikealla mygame pakettia
- Valitse New/Class
- Anna nimeksi Main ja klikkaa Finish
- Korvaa Main.java tiedostoon generoitunut koodi seuraavalla:

```
package mygame;

import com.jme3.app.SimpleApplication;
import com.jme3.material.Material;
import com.jme3.math.ColorRGBA;
import com.jme3.renderer.RenderManager;
import com.jme3.scene.Geometry;
import com.jme3.scene.shape.Box;

/**
 * This is the Main Class of your Game. You should only do initialization here.
 * Move your Logic into AppStates or Controls
 * @author normenhansen
 */
public class Main extends SimpleApplication {

    public static void main(String[] args) {
        Main app = new Main();
        app.start();
    }

    @Override
    public void simpleInitApp() {
        Box b = new Box(1, 1, 1);
        Geometry geom = new Geometry("Box", b);

        Material mat = new Material(assetManager, "Common/MatDefs/Misc/Unshaded.j3md");
        mat.setColor("Color", ColorRGBA.Blue);
        geom.setMaterial(mat);

        rootNode.attachChild(geom);
    }

    @Override
    public void simpleUpdate(float tpf) {
        //TODO: add update code
    }

    @Override
    public void simpleRender(RenderManager rm) {
        //TODO: add render code
    }
}
```

Klikkaa oikealla Main.java ja valitse Run As/Java Application. Ilmestyy jMonkey ikkuna, jossa valitaan Continue. (Fullscreen on parempi jättää ruksaamatta – toimii luotettavammin). Pitäisi tulla näkyviin sininen kuutio. Hiirellä voi vaihtaa kuvakulmaa ja näppäimilla ’ qwasdz’ voi liikkua. Kun olet kokeillut liikkumista, sulje sovellus painamalla Esc.

Hiiri voi olla yliherkkä, jos sitä käytetään etäyhteyden yli. Yllä listatusta koodista on saatavilla toinen versio, jossa hiiri on korvattu nuolinäppäimillä. Kts. MyCoursesissa Materials sivulla ’ 3D Java sininen kuutio ilman hiirtä’.

Tehtävät pystyy tekemään ilman että vaihtaa kuvakulmaa.

Kurssilla tehdään ensin 3D lego ja sitten yksinkertainen robotti joka siirtelee näitä legoja varastopaikasta kokoonpanoasemalle.

Tiedostossa Main.java on luokka 'Main' jossa on 4 metodia.

```
public class Main extends SimpleApplication
```

tarkoittaa että luokka perii jMSimpleApplication-luokan ominaisuudet, eli pelimoottorin tekemät asiat tapahtuu konepellin alla ja päästään muutamalla koodirivillä tekemään asioita.

Aloitetaan tekemällä lego. Sitä ennen vähän teoriaa:

Teoria
Geometria kuvaa jonkin kappaleen 3D muodon suurena määränä kolmioita, jotka ovat niin pieniä että ihminen ei havaitse että kysymyksessä on kolmio. Geometrian voi luoda CAD tiedostosta. jMonkey tarjoaa valmiina joitakin yksinkertaisia geometrioita (laatikkoja, palloja yms.)
Noodi on id, jolla päästään käsiksi 3D ympäristössä näkyvään geometriaan. Kaikki noodit ovat graafissa (scene graph), jonka juuri on 'rootNode'. Jotta luomasi geometria tulee näkyviin, se pitää liittää rootNodeen tai johonkin sen alla olevaan noodiin. Jos noodia liikutellaan, niin kaikki sen alla olevat noodit liikkuu.

Teoria
Sovellusohjelmointirajapinta (API Application Programming Interface): vaaditaan paljon yksityiskohtaista koodia, jotta saadaan yksinkertainen 3D muoto ruudulle niin että sitä voidaan katsella eri kulmista ja niin että valo heijastuu siitä realistisella tavalla. Nämä koodit on paketoitu luokkiin jMonkey APIssa, joten riittää että käytetään näitä luokkia ilman että tiedetään miten ne on toteutettu. Näin yllämainitut asiat saadaan aikaiseksi kirjoittamalla muutama rivi koodia.

Klikkaa oikealla 'mygame' ja valitse New/Class ja anna nimeksi 'Lego'. Deletoi olemassa oleva koodi ja kopioi siihen seuraava:

```
package mygame;

import com.jme3.asset.AssetManager;
import com.jme3.material.Material;
import com.jme3.math.ColorRGBA;
import com.jme3.scene.Geometry;
import com.jme3.scene.Node;
import com.jme3.scene.shape.Box;

public class Lego {
    Node node = new Node();
    Geometry geom;
    Box box;

    public Lego(AssetManager assetManager) {
        box = new Box(0.8f, 0.2f, 0.4f);
```

```
        geom = new Geometry("Box", box);
        node.attachChild(geom);

        Material mat = new Material(assetManager,
"Common/MatDefs/Light/Lighting.j3md");
        mat.setBoolean("UseMaterialColors",true);
        ColorRGBA c = ColorRGBA.Green;
        mat.setColor("Diffuse",c);
        geom.setMaterial(mat);
    }
}
```

Huomioita:

- Node, Geometry, Box ja Material ovat jMonkey APIn luokkia, jotka ovat valmiit käytettäviksi kun on tehty import komento, jolla otetaan kyseinen osa APIa käyttöön.
- 'f' kirjain esim 0.8f tarkoittaa että tietotyyppi on float (API vaatii tätä)
- 'Lego' metodi on konstruktori, joka luo yhden instanssin (olion) tästä luokasta. Aina kun sitä kutsuu uudestaan, niin luodaan uusi lego.
- Material luokka määrittelee pintamateriaalin ja miten se heijastaa valoa (ei mennä näihin valaistusteknisiin yksityiskohtiin tarkemmin tällä kurssilla)
- assetManager olio tarvitaan, jotta päästään käsiksi "Lighting.j3md" resurssiin, jolla luodaan materiaali

Päivitä Main.javan simpleInitApp()

```
    public void simpleInitApp() {
        flyCam.setMoveSpeed(10);
        Lego lego = new Lego(assetManager);
        rootNode.attachChild(lego.node);

        PointLight lamp_light = new PointLight();
        lamp_light.setColor(ColorRGBA.White);
        lamp_light.setRadius(400f);
        lamp_light.setPosition(new Vector3f(2f, 8.0f, 10.0f));
        rootNode.addLight(lamp_light);
    }
```

ja lisää muiden importtien joukkoon:

```
import com.jme3.light.PointLight;
```

Tulee seuraava ongelma:

Vector3f on jMonkeyn oma tyyppi ja sekin pitää importata. Klikkaa lampun kuvaketta, jonka päällä on punainen huutomerkki. Valitse listan ensimmäinen:

```
Add import for com.jme3.math.Vector3f
```

mikä lisää importin luultavasti äsken lisäämäsi PointLight importin perään. HUOM: jatkossa tulet monta kertaa törmäämään tähän ongelmaan, joka yleisesti ottaen ratkeaa samalla tavalla: valitse listan ensimmäinen 'Add import'.

Kohta päästään itse koodaamaan, mutta aluksi tutustutaan muutamiin Java ja jMonkey perusasioihin kopioimassamme koodissa. Huomioita:

- flyCam.setMoveSpeed() kertoo miten nopeasti näppäimillä 'qwasdz' liikutaan
- luodaan 'Lego' luokasta yksi instanssi (olio) 'lego'
- lisätään legon noodi rootNoden (muuten lego ei tule näkyviin)
- lisätään valonlähde kameran taakse (ilman valoa mitään ei näy) – liikuskele 'qwasdz' näppäimillä niin huomaat miten legon eri pinnat on eri värisiä riippuen siitä missä kulmassa valo osuu siihen

Java vs Python

olion attribuuttiin tai metodiin päästään käsiksi samalla tavalla kuin Pythonissa, eli

```
<olio nimi>.<attribuutin tai metodin nimi>
```

esim rootNode ja lego on olion nimiä:

```
rootNode.attachChild(lego.node);
```

Javassa sisennyksellä ei ole mitään väliä – mutta luettavuuden takia kannattaa noudattaa samanlaista sisennystä kun Pythonissä. Kaarisuluilla ilmaistaan että mitkä rivit kuuluu luokkamäärittelyyn/metodiin/silmukkaan/if lauseeseen

if syntaksi:

```
if(<ehto>) {
    <1 tai useampi rivi>
} else if (<ehto>) {
    <1 tai useampi rivi>
} else {
    <1 tai useampi rivi>
}
```

puolipistettä käytetään rivin lopussa

metodin parametrien ja muuttujien määrittelyn yhteydessä pitää kertoa mikä tyyppi on kysymyksessä (primitiivinen tyyppi kuten String, itse määritelty tyyppi kuten Lego tai APIssa määritelty tyyppi kuten AssetManager)

Tehtävä

Lisää Lego luokan konstruktoriin String tyyppinen parametri 'color', jolla voi olla joku seuraavista arvoista: "green", "red", "yellow", "pink", "blue". Päivitä konstruktorin koodi (kts if-lauseen syntaksi yllä) siten että väri on parametrin mukainen tai tummanharmaa jos parametrin arvo on joku muu.

Käytä:

```
ColorRGBA.Blue
ColorRGBA.Pink
ColorRGBA.Yellow
ColorRGBA.Green
ColorRGBA.Red
ColorRGBA.DarkGray
```

Main.javan simpleInitApp()ssa luo nyt 5 legoa näillä väreillä. Aja sovellus niin huomaat että ne meni kaikki päällekäin ja vain yksi näkyy. Legoa voi siirtää X-akselilla eteenpäin näin:

```
legoGreen.node.setLocalTranslation(2f, 0, 0);
```

`legoGreen` on `Lego` luokan instanssin eli olion nimi. Pisteellä päästään kiinni olion attribuutteihin ja metodeihin. `node` on attribuutti, eli kutsutaan sen `setLocalTranslation()` metodia. Tästä syystä metodin kutsuminen siirtää ainoastaan vihreää legoa.

Siirtele legot niin että ne on vierekkäin X-akselilla.

Legoihin tarvitaan vielä pienet sylinterin muotoiset kiinnityskohdat. Lisää lego luokan konstruktorin loppuun:

```
Cylinder cyl = new Cylinder(20, 20, 0.1f, 0.1f, true);
Geometry g = new Geometry("C", cyl);
g.rotate(FastMath.HALF_PI, 0, 0);
g.setLocalTranslation(0.6f, 0.2f, 0.2f);
g.setMaterial(mat);
node.attachChild(g);
```

Rotaatio tarvitaan, koska sylinterin luodaan siten että sileä pinta on pystysuoraan. setLocalTranslation(x,y,z) tarvitaan, koska muuten sylinteri menisi legon keskelle eikä edes näkyisi legon ulkopuolelta. y arvo 0.2f on sama kuin legon ”box” attribuutin y-arvo. x ja z koordinaatit on määritelty niin että sylinteri menee halutulle paikalle. Kun ajat ohjelman, huomaat että jokaiseen viiteen legoon tuli yksi sylinteri. Tämä johtuu siitä että uusi sylinteri luodaan aina kuin ’Lego’ luokasta tehdään uusi instanssi eli olio.

Java vs Python

for syntaksi Javassa:

```
for(int i = 0; <ehto>; i++) {
    <1 tai useampi rivi>
}
```

Tehtävä

Nyt legolla on vain 1 sylinteri-kiinnike, mutta niitä pitäisi olla 8. Siirrä sylinterin luonut koodinpätkä for silmukkaan, joka luo kaikki 8. Vihjeitä:

- i / x on jakolasku ilman jakojäännös, esim jos i on 5 niin i/2 on 2
- i % x on jakojäännös, esim jos i on 5 niin i%2 on 1

Reflektointi

Motivointi

Kuten Automaatiopyramidi-lukupaketista käy ilmi, automaatio on paljon muutakin kuin sensoreihin ja toimilaitteisiin suoraan liittyviä ohjausohjelmistoja. Tekoälyn, pilviteknologian ja esineiden internetin myötä korkeamman tason ohjelmistojen merkitys vaan kasvaa. Laiteläheinen ohjelmointi tehdään

PLC:n ja mikrokontrollerien kaltaisilla teknologioilla, mutta kaikki muu on olio-ohjelmointia. Monenlaisten APIen käyttö on välttämätön osa ammattitaitoa. Java ja Python ovat yleisimmät kielet ja jos niitä jotenkin osaa niin ne kannattaa mainita CV:ssä kun hakee kesätöitä.

Reflektoidaan dokumentin alussa mainittuihin tavoitteisiin liittyviä asioita.

Mieti seuraavia kysymyksiä ja sitten kun olet valmis kertomaan niistä niin demoa ratkaisusi opettajalle – samassa yhteydessä käydään läpi nämä kysymykset. Pisteet saa siitä, että on miettinyt asioita.

- Missä määrit olit oppinut olio-ohjelmointia aikaisemmilla kursseilla?
- Selitä omin sanoin mikä on luokan ja olion ero käyttäen esimerkkejä harjoitustyöstä.
- Ajankäytön suunnittelu: meillä on PC luokassa 2 tunnin sessio kerran viikossa 5 viikon ajan, missä tehdään ”Automaation tietotekniikka” osuutta, joka on 25% 5op kurssista. Näin ollen tämän 10h kontaktiopetuksen lisäksi on oletus, että tehdään myös itsenäistä työtä. Kurssilaisten aiemmassa tietotekniikkaosaamisessa on hyvinkin suuria eroja, joten työmäärä on yksilöllinen. Työmäärä on mitoitettu siten, että opiskelijalta ei oleteta mitään muuta esitietoja kuin OODissa mainitut esitieto kurssit, mikä tarkoittaa sitä, että opiskelijoiden oletetaan tekevän itsenäistä työtä PC luokka sessioiden ulkopuolella. Suunnittele ajankäyttöä ryhmässäsi: miten työskentelette näiden sessioiden ulkopuolella, jotta pysytte aikataulussa ja voitte hyödyntää viikoittaiset sessiot demoamiseen ja teknisen tuen pyytämiseen.