UNIVERSITAS INDONESIA

PROYEK AKHIR GRAFIKA KOMPUTER

MGS. M. RIZQI FADHLURRAHMAN

1306464543

MUHAMMAD LUTHFI

1306386826

FAKULTAS ILMU KOMPUTER

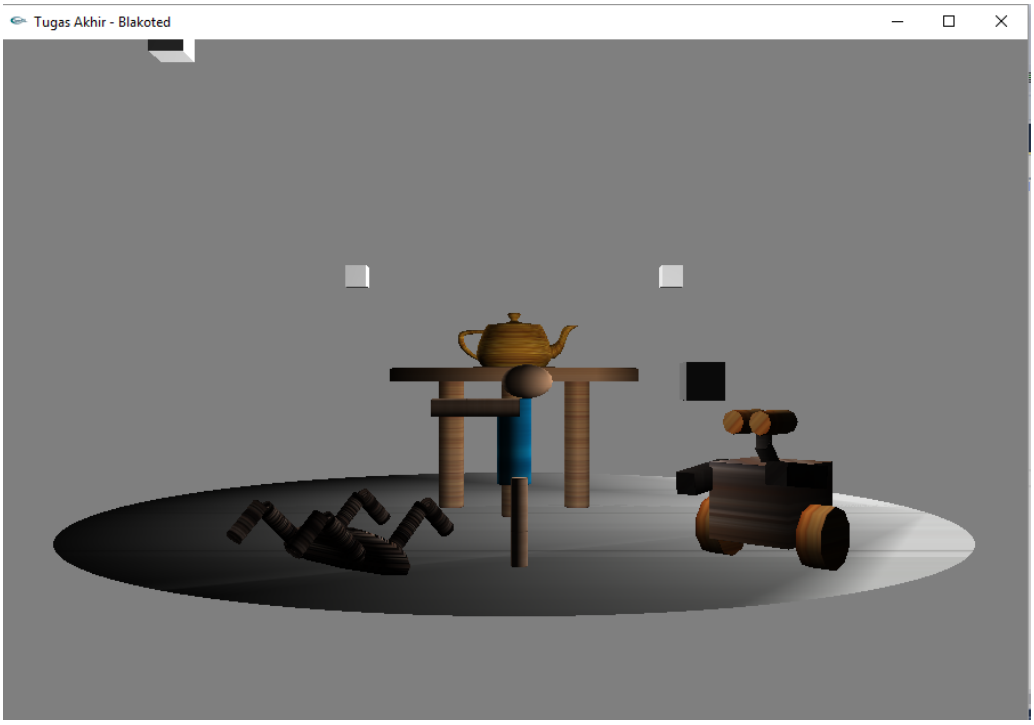
PROGRAM STUDI ILMU KOMPUTER

DEPOK

JUNI 2016

1. MANUAL KOMPILASI

Program ini akan menampilkan satu *scene* dimana terdapat 1 objek laba-laba, 1 objek *wall-e*, 1 objek manusia, 1 objek meja, dan 1 objek *teapot* yang terletak di atas meja. Masing-masing objek terletak pada sebuah bidang berbentuk bulat. Selain itu juga terdapat 4 buah lampu yang berada di sekeliling *scene*. Untuk menjalankan program ini, cukup membuka *source code* melalui *Visual Studio*, letakkan file dari *texture.zip* ke dalam folder *project* anda, dan dengan menekan tombol f5 maka program sudah dapat digunakan



Gambar 1 Tampilan Dasar Program

2. MANUAL PENGGUNA

Seperti yang telah disebutkan pada bagian 1, pada program ini terdapat 5 buah objek dan 4 buah lampu. Beberapa objek dan lampu dapat dikendalikan pergerakannya. Terdapat 1 lampu yang berperan sebagai lampu sorot. Pengguna dapat memilih menu dengan menekan klik kanan. Terdapat beberapa menu sebagai berikut :

a. *View*

Menu ini digunakan untuk mengatur posisi kamera, terdapat beberapa sub-menu yakni *camera, spider, human*, dan *wall-e*. Sub-menu *camera* digunakan apabila pengguna ingin melihat kondisi keseluruhan *scene*. Sedangkan sub-menu *spider*, *human*, dan *wall-e* digunakan apabila pengguna ingin melihat *scene* dari sudut pandang orang ketiga dari objek yang bersangkutan.

b. *Control*

Menu ini digunakan untuk mengatur pergerakan objek, terdapat beberapa sub-menu yakni *camera, spider, human, wall-e*, lamp3, dan lamp4. Semua objek dalam sub-menu ini dapat digerakkan dengan menakan tombol arah untuk bergerak kiri, depan, kanan, belakang. Khusus untuk objek *camera*, lamp3, dan lamp4 tombol arah berfungsi untuk bergerak kiri, atas, kanan, bawah dan pergerakan maju mundur dikendalikan melalui tombol 'w' dan 's'. Objek lamp3 dan lamp4 adalah 2 objek lampu yang terletak paling dekat dengan kamera. Lamp4 adalah satu-satunya objek lampu yang berperan sebagai lampu sorot dengan arah sorot kebawah.

c. *Lamp*

Menu ini digunakan untuk menyalakan dan mematikan semua lampu yang ada pada *scene*. Sub-menu lamp1 untuk mengendalikan lampu di belakang-kiri. Sub-menu lamp2 untuk mengendalikan lampu di belakang-kanan. Sub-menu lamp3 untuk mengendalikan lampu di depan-kanan. Sub-menu lamp4 untuk mengendalikan lampu di depan-kiri (lampu sorot). Keterangan: kata 'depan' mengacu pada posisi terdekat dari kamera.

d. *Object Mode*

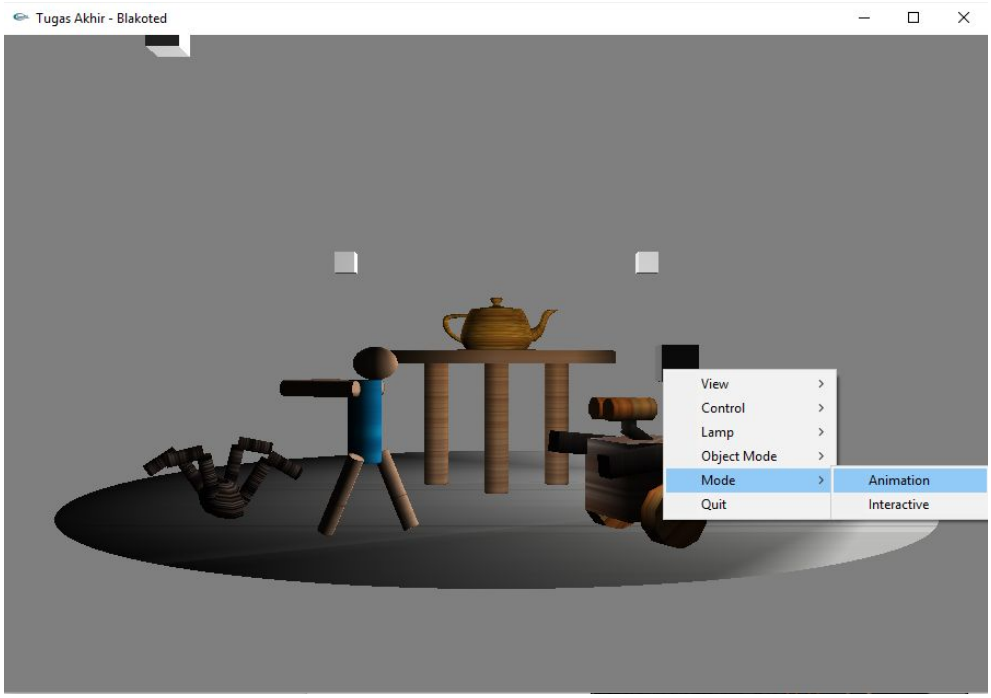
Objek mode digunakan untuk menentukan mode objek pada *method* glPolygonMode. Sub-menu *Shading* berfungsi untuk menentukan mode *Shading* pada objek, atau mode GL_FILL pada *method* glPolygonMode. Sub-menu *Wireframe* berfungsi untuk menentukan mode *Wireframe* pada objek, atau mode GL_LINE pada *method* glPolygonMode.

e. Mode

Mode digunakan untuk menentukan *state* pada *scene*. Sub-menu animasi digunakan untuk menjalankan mode animasi, dimana ketiga objek berhirarki akan bergerak mengitari titik pusat. Sedangkan sub-menu interaktif digunakan apabila pengguna ingin menggerakan ketiga objek berhirarki secara manual. Pengaturan untuk pergerakan objek dapat dilihat pada menu *Control* (b).

f. Quit

Menu ini digunakan apabila pengguna ingin keluar dari program.



Gambar 2 Penggunaan Menu Dengan Klik Kanan

3. PEMBENTUKAN OBJEK

a. Spider

Objek *spider* merupakan sebuah *tree* yang memiliki 9 *node*. Berikut representasi *tree*-nya

```
->torso
    ->upper_arm_1
        ->lower_arm_1
    ->upper_arm_2
        ->lower_arm_2
    ->upper_arm_3
        ->lower_arm_3
    ->upper_arm_4
        ->lower_arm_4
```

Gambar 3.1 Hirarki Objek *Spider*

b. Wall-E

Objek Wall-E merupakan sebuah tree yang memiliki 11 node. Berikut ini representasi tree-nya.

```
->body
    ->lowerNeck
        ->upperNeck
        ->rightEye
        ->leftEye
    ->rightUpperArm
        ->rightLowerArm
    ->leftUpperArm
```

```
        ->leftLowerArm
    ->rightWheel
    ->leftWheel
```

Gambar 3.2 Hirarki Objek *wall-e*

c. Human

Objek *human* merupakan sebuah *tree* yang memiliki 10 *node*. Berikut representasi *tree*-nya.

```
->torso
    ->head
    ->left_upper_arm
        ->left_lower_arm
    ->right_upper_arm
        ->right_lower_arm
    ->left_upper_leg
        ->left_lower_leg
    ->right_upper_leg
        ->right_lower_leg
```

Gambar 3.3 Hirarki Objek *Human*

Pembuatan masing-masing objek berhirarki menggunakan algoritma *treenode* yang kami dapatkan dari sebuah contoh yang ada pada *course* Grafkom.

d. *Table* and *Teapot*

Objek *Table* dibuat menggunakan gluCylinder sedangkan objek *Teapot* dibuat menggunakan gluTeapot. Kedua objek non-hirarki ini tidak dapat digerakkan.

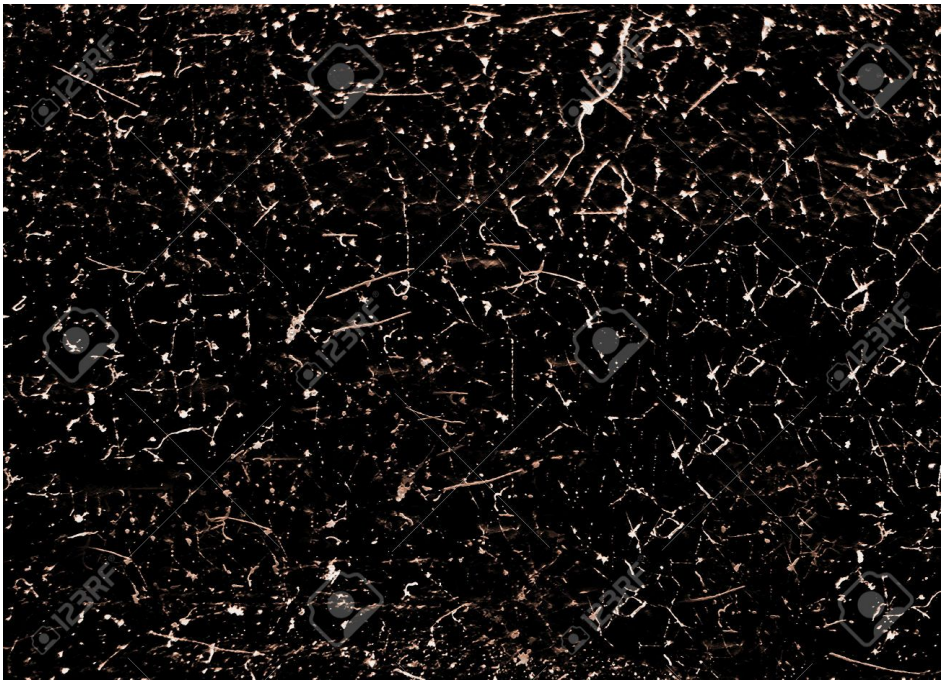
4. RENDERING OBJEK & SCENE

a. Lighting

Di dalam *scene* ini, terdapat 4 lampu. Lampu 1 & 2 tidak dapat digerakkan, berada di belakang *scene*. Lampu 3 & 4 dapat digerakkan, berada di depan *scene* (paling dekat dengan kamera). Lampu 4 adalah lampu sorot. Lampu ini mengeluarkan cahaya yang mengarah kebawah.

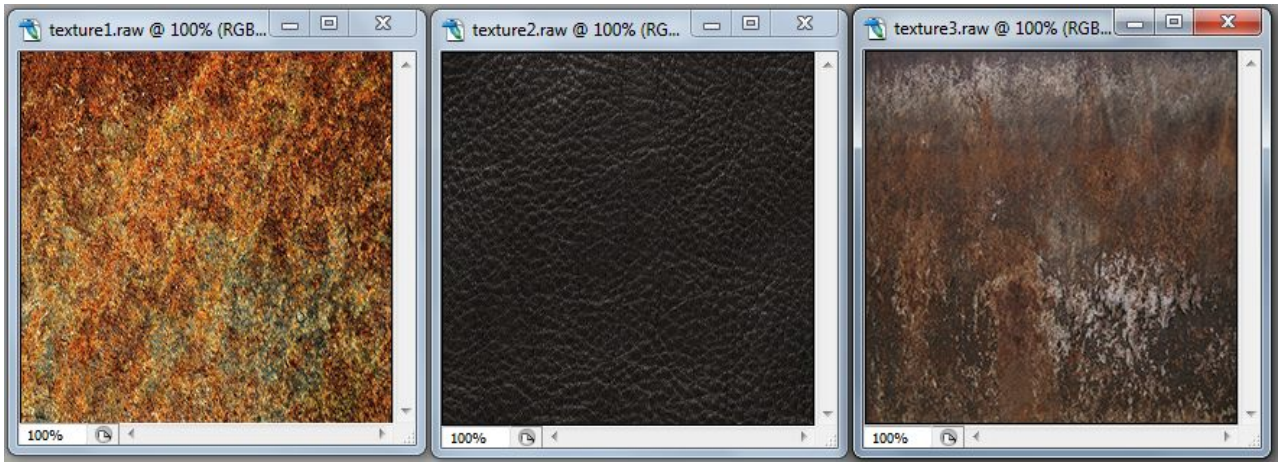
b. Texture

Untuk objek *spider* kami menggunakan satu macam *texture* yaitu spider.raw yang merupakan tekstur kulit laba-laba



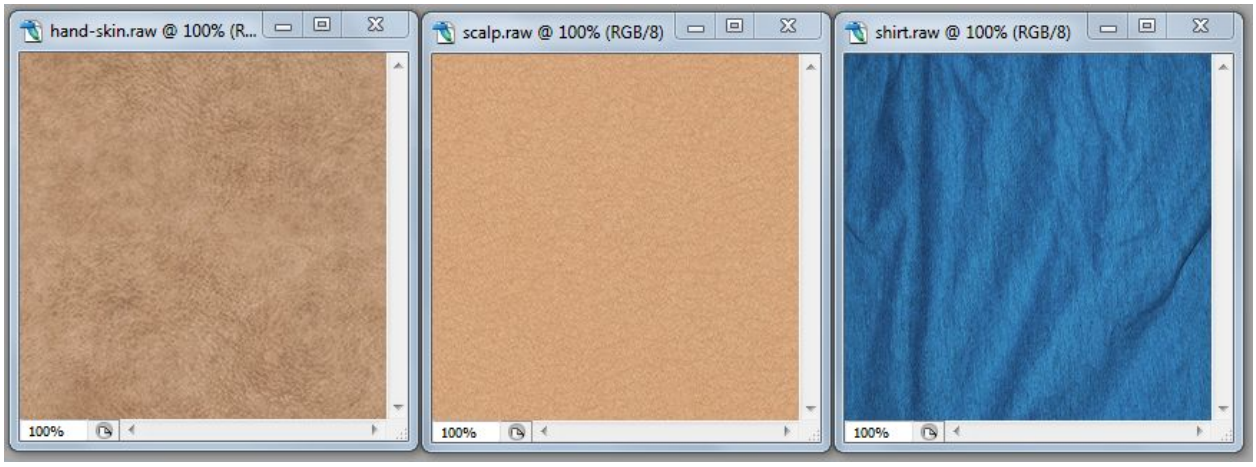
Gambar 4.1 Tekstur dari objek *spider*

Pada objek *wall-e*, diterapkan tiga *texture* berbeda, *texture* yang digunakan memiliki tampilan sebagai berikut.



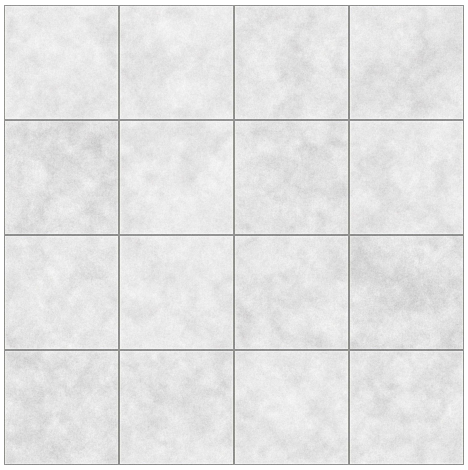
Gambar 4.2 Tekstur dari objek *wall-e*

Pada objek *human*, tekstur digunakan untuk bagian kepala, tangan dan kaki (tekstur kulit), dan tekstur baju dari *human*.



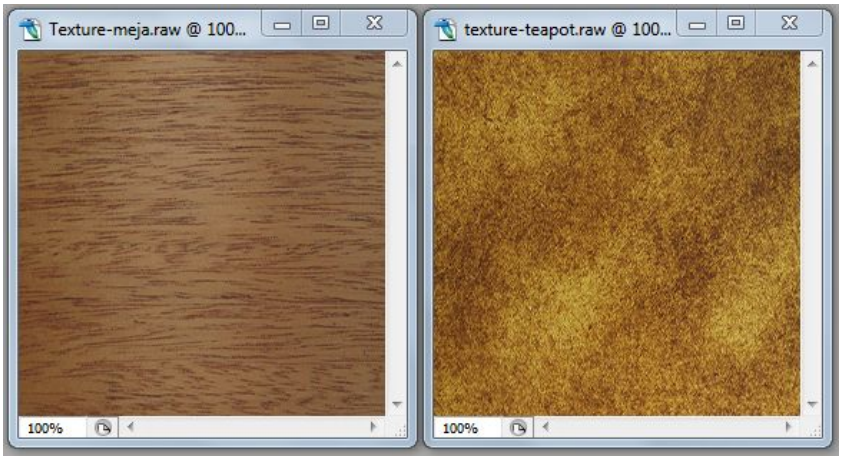
Gambar 4.3 Tekstur dari objek *human*

Selain itu, kami juga mengaplikasikan tekstur pada lantai berupa ubin. Berikut tekstur ubin yang kami gunakan.



Gambar 4.4 Tekstur lantai

Terakhir, kami juga menggunakan tekstur pada objek *table* dan *teapot* di tengah ruangan. Objek *table* menggunakan tekstur kayu, dan objek *teapot* menggunakan tekstur kuningan yang sudah karatan.



Gambar 4.5 *Texture table* dan *teapot*

5. FUNGSI-FUNGSI OPENGL

- glPushMatrix() & glPopMatrix() : Untuk menyimpan *state modelview matrix*.
- glLoadIdentitiy(): Untuk mendapatkan matriks identitas agar matriks proyeksi suatu objek tidak terganggu.
- gluSolidCube(), gluSphere, gluCylinder(), glutSolidcone(), glutSolidTeapot() : Digunakan untuk membuat objek-objek.
- gluNewQuadric(), gluQuadricDrawStyle(), gluQuadricTexture() : menginisalisasi objek berhirarki.
- glPolygonMode(), glShadeModel() : menentukan mode penggambaran objek (*shading* atau *wireframe*).
- glClear() : mengosongkan *buffer*, untuk program ini hanya *color* dan *depth buffer* yang dikosongkan.
- glGetFloatv() : untuk memposisikan matriks proyeksi suatu objek.
- glTranslate*(), glRotate*(), glScale*() : Untuk menranlasi, merotasi, dan memberi skala pada objek.
- glEnable(GL_TEXTURE_2D) : Untuk mengaktifkan tekstur.
- glBindTexture() : Untuk memilih tekstur.
- glGenTextures(), glTexEnvf(), glTexParameterf(), gluBuild2DMipmaps() : pengaturan tekstur.
- glLightFv() : pengaturan sumber cahaya.
- gluPerspective(), glViewport() : Digunakan sebagai *projection matrix*.
- glClearColor() : Menentukan warna dasar dari *scene*.
- glBegin(GL_POLYGON), glTexCoord2f, GlVertex3fv, GlEnd() : Untuk pembuatan dan penerapan tekstur pada lantai.
- glMultMatrixf() : Untuk mendapatkan proyeksi *shadow matrix* relatif terhadap *modelview matrix*. Selain itu juga digunakan untuk mendapatkan posisi *child* relatif terdapat *parent* dalam objek ber hirarki.

- glMatrixMode() : Untuk pergantian mode antara *projection matrix* dan *modelview matrix*.
- glutSwapBuffer(), glutPostRedisplay(), glutInit(), glutInitDisplayMode(), glutInitWindowSize(), glutCreateWindow(), glutReshapeFunc(), glutDisplayFunc(), glutCreateMenu(), glutAddSubMenu(), glutAddMenuEntry(), glutAttachMenu(), glutSpecialFunc(), glutKeyboardFunc(), glutIdleFunc() : keperluan *display* untuk merender *scene*.

6. ALGORITMA KHUSUS

a. Pembentukan lantai dan animasi

Untuk membentuk lantai kami menggunakan algoritma pembentukan lingkaran yang disambungkan dengan *method* GL_POLYGON. Dimana akan dihitung setiap *sinus* dan *cosinus* dari setiap sudut yang dikalikan dengan jari-jari. (DEG2RAD = pi/180)

```
glBegin(GL_POLYGON);
   for (int i=0; i <= 360; i++)
   {
      float degInRad = i*DEG2RAD;
      float xcos = cos(degInRad);
      float ysin = sin(degInRad);
      float tx = xcos * 0.5 + 0.5;
         float ty = ysin * 0.5 + 0.5;

      glTexCoord2f(tx, ty);
      glVertex3f(xcos*33, -10.5, ysin*33);
   }
glEnd();
```

Gambar 6.1 Algoritma lingkaran

Untuk animasi pergerakan objek juga mengikuti konsep yang sama dengan algoritma diatas hanya saja dengan jari-jari yang lebih kecil. Rotasi dari setiap objek juga disesuaikan dengan pergerakan lingkaran. Berikut contoh animasi pergerakan untuk objek *spider*

```
degInRad = i_global*DEG2RAD;
xcos = cos(degInRad);
ysin = sin(degInRad);
spider.translation[0] = xcos*22;
spider.translation[2] = ysin*22;
spider.rotation = -i_global + 90.0;
```

Gambar 6.2 Animasi pergerakan objek *spider*

- Animasi individu *spider*

Pada saat mode animasi, spider akan menggerakan kaki-kakinya. Kaki kiri-depan dan kanan-belakang memiliki pergerakan yang sama, begitu juga dengan kiri-belakang dan kanan-depan yang memiliki pergerakan berlawanan dari pasangan pertama. Batas pergerakan adalah 90 derajat. Berikut adalah algoritmanya

```
if(forwardSpider)
{
        thetaSpider[1] += 2.0;
        thetaSpider[7] -= 2.0;

        if(thetaSpider[5] >= 0.0)
        {
                thetaSpider[3] += 2.0;
                thetaSpider[5] -= 2.0;
        }

        if(thetaSpider[1] >= 45.0)
        {
                forwardSpider = false;
        }
}
else
{
        thetaSpider[1] -= 2.0;
        thetaSpider[7] += 2.0;
        thetaSpider[3] -= 2.0;
        thetaSpider[5] += 2.0;

        if(thetaSpider[1] <= 0.0)
        {
                forwardSpider = true;
        }
}
```

Gambar 6.3 Animasi individu objek *spider*

- Animasi individu *Wall-E*

Animasi Wall-E berisi pergerakan seluruh bagian tubuh Wall-E (mata, leher, badan, tangan, roda) ketika sedang berjalan. Batas pergerakan animasi berdasarkan animasi mata, yaitu sebesar 25 derajat. Sedangkan roda akan selalu berputar dengan pertambahan 0.45 derajat.. Berikut adalah algortimanya

```
if (stateWallE == 0) {
    if (thetaWallE[1] < 25.0) {
        thetaWallE[1] += 0.3;
        thetaWallE[2] -= 0.2;
        thetaWallE[3] += 0.2;
        thetaWallE[5] -= 0.15;
        thetaWallE[6] += 0.15;
        walleBodyTranslate += 0.001;
    }
    else {
        stateWallE = 1;
    }
}
else if (stateWallE == 1) {
    if (thetaWallE[1] > 0.0) {
        thetaWallE[1] -= 0.3;
        thetaWallE[2] += 0.2;
        thetaWallE[3] -= 0.2;
        thetaWallE[5] += 0.15;
        thetaWallE[6] -= 0.15;
        walleBodyTranslate -= 0.001;
```

```
    }
    else {
        stateWallE = 0;
    }
}
thetaWallE[9] += 0.45;
thetaWallE[10] += 0.45;
```

Gambar 6.4 Animasi individu objek Wall-E

- Animasi individu *human*

Animasi *human* berisi pergerakan kaki dengan batas pergerakan sebesar 135 derajat. Berikut adalah algortimanya

```
for (int i = 0; i < 2; i++) {
       if (thetaOfficer[2 * i + 3] > 135) forwardOfficer[i] = false;
       if (thetaOfficer[2 * i + 3] < 45) forwardOfficer[i] = true;
       if (forwardOfficer[i]) thetaOfficer[2 * i + 3] += 2.5;
       else thetaOfficer[2 * i + 3] -= 2.5;
}
```

Gambar 6.5 Animasi individu objek *Human*

b. Pergantian View Kamera

Untuk pergantian posisi kamera kami mengadopsi algoritma dari tugas akhir grafkom 2015 sebagaimana yang disebutkan pada bagian Referensi nomor (2). Untuk program kami, pergantian kamera hanya bisa dilakukan pada saat mode interaktif. Untuk mode animasi tidak bisa karena terkendala penyesuaian arah objek.

```
if (view.compare("camera") == 0)
       {
              glTranslated(camera.translation[0], camera.translation[1],
camera.translation[2]);
       }
       else if (view.compare("dog") == 0)
       {
              if (dog.direction.compare("left") == 0)
              {
                     glTranslated(dog.translation[2], dog.translation[1] + 11,
-dog.translation[0] - 15);
                     glRotated(-90, 0, 1, 0);
              }
              else if (dog.direction.compare("right") == 0)
              {
                     glTranslated(-dog.translation[2], dog.translation[1] + 11,
dog.translation[0] - 15);
                     glRotated(90, 0, 1, 0);
              }
              else if (dog.direction.compare("up") == 0)
              {
                     glTranslated(-dog.translation[0], dog.translation[1] + 11,
-dog.translation[2] - 15);
                     glRotated(0, 0, 1, 0);
```

```
               }
               else
               {
                      glTranslated(dog.translation[0], dog.translation[1] + 11,
dog.translation[2] - 15);
                      glRotated(180, 0, 1, 0);
               }

       }
       else if (view.compare("wallE") == 0)
       {
                if (wallE.direction.compare("left") == 0)
                {
                       glTranslated(wallE.translation[2], wallE.translation[1] + 12,
-wallE.translation[0] - 22.5);
                       glRotated(-90, 0, 1, 0);
                }
                else if (wallE.direction.compare("right") == 0)
                {
                       glTranslated(-wallE.translation[2], wallE.translation[1] + 12,
wallE.translation[0] - 22.5);
                       glRotated(90, 0, 1, 0);
                }
                else if (wallE.direction.compare("up") == 0)
                {
                       glTranslated(-wallE.translation[0], wallE.translation[1] + 12,
-wallE.translation[2] - 22.5);
                       glRotated(0, 0, 1, 0);
                }
                else
                {
                       glTranslated(wallE.translation[0], wallE.translation[1] + 12,
wallE.translation[2] - 22.5);
                       glRotated(180, 0, 1, 0);
                }
         }
       else
       {
              if (officer.direction.compare("left") == 0)
              {
                     glTranslated(officer.translation[2], officer.translation[1] +
5, -officer.translation[0] - 22.5);
                     glRotated(-90, 0, 1, 0);
              }
              else if (officer.direction.compare("right") == 0)
              {
                     glTranslated(-officer.translation[2], officer.translation[1] +
5, officer.translation[0] - 22.5);
                     glRotated(90, 0, 1, 0);
              }
              else if (officer.direction.compare("up") == 0)
              {
                     glTranslated(-officer.translation[0], officer.translation[1] +
5, -officer.translation[2] - 22.5);
                     glRotated(0, 0, 1, 0);
              }
```

```
            else
            {
                  glTranslated(officer.translation[0], officer.translation[1] +
5, officer.translation[2] - 22.5);
                  glRotated(180, 0, 1, 0);
            }
      }
```

Gambar 6.6 Animasi pergantian *view* kamera

7. PEMBAGIAN TUGAS DAN LOG PEKERJAAN

a. Muhammad Luthfi :
   - Membuat objek berhirarki *spider*.
   - Pengaturan lantai, lighting, animasi objek-objek.
   - Dokumentasi.

b. Mgs. M. Rizqi Fadhlurrahman:
   - Membuat objek berhirarki Wall-E
   - Membuat animasi individu Wall-E
   - Membuat objek meja
   - Dokumentasi

Log Pekerjaan:

Jumat, 3 Juni 2016
- Penyusunan ide dan pembagian kerja.

Selasa, 7 Juni 2016
- Proses implementasi
- Pembangunan objek-objek berhirarki
- Mengatur *lighting*
- Membuat animasi

Rabu, 8 Juni 2016
- Penyelesaian Implementasi
- Pengaturan kamera
- Membuat menu
- Penyelesaian setiap objek, *lighting*, dan animasi.

Kamis, 9 Juni 2016
- Menyusun laporan.

8. REFERENSI

- Algoritma lingkaran : http://stackoverflow.com/questions/8762826/texture-mapping-a-circle-made-using-gl-polygon?answertab=active#tab-top
- Tugas Akhir Grafkom 2015 atas nama Aditya Wicaksono Putro (2011), Emmanuela Hartono (2011), dan Bimo Prasetyo Sigit (2011).