UNIVERZITET U TUZLI
FAKULTET ELEKTROTEHNIKE

Izvještaj o projektu

Računarska grafika

Student: Tarik Hamidović
Usmjerenje: Računarstvo i informatika
Broj indeksa: 17045

Profesor:
Dr.sc. Emir Skejić
vanr. prof.

Tuzla, juli 2020.

Sadržaj

Zadatak

Napisati OpenGL program koji, uz minimalnu upotrebu OpenGL API-ja, zadovoljava sljedeće specifikacije:

1. Kreirati 3D scenu koja se sastoji od barem 5 objekata. Ukoliko želite, možete koristiti GLU i/ili GLUT objekte. (Imajte na umu da ćete, ako se odlučite za kreiranje vlastitih objekata, morati generirati i njihove normale za osvjetljenje i svojstva materijala.)

2. Koristiti barem dvije različite non-default specifikacije koordinata kamere. (Kontrolu "situacije" možete vršiti pomoću tastature za odabir postavki ili možete kreirati animaciju koja uključuje kretanje kamere.)

3. Animirati jedan ili više objekata pomoću glRotate, glScale i glTranslate.

4. Koristiti miš za određivanje brzine i smjera animacije. Lijevo dugme miša koristiti za pokretanje animacije pomoću rotacije, srednje dugme za pokretanje animacije pomoću translacije, a desno dugme za pokretanje animacije pomoću skaliranja. (Ovo uključuje korištenje glutMouseFunc, glutMotionFunc/glutPassiveMotionFunc, glutIdleFunc.)

5. Održavati animaciju kada nijedno dugme miša nije pritisnuto, pomoću glutIdleFunc callback funkcije.

6. Na barem 2 objekta u sceni primijeniti jednostavno teksturiranje.

7. Konačno, u program dodati dva različita izvora svjetlosti i dva različita materijala.

Putem tastature omogućiti sljedeće funkcionalnosti:

- 0 – uključuje i isključuje svjetlo 0
- 1 – uključuje i isključuje svjetlo 1
- 2 – uključuje i isključuje materijal 1
- 3 – uključuje i isključuje materijal 2

Rješenja

Prva specifikacija

Kreirana je scena sa sedam 3D objekata koji simuliraju model atoma. Objekti koji su se koristili u ovoj sceni su:

- 4 sfere (kreirane pomoću funkcije gluSphere(GLUquadric* quad, GLdouble radius, GLint slices, GLint stacks)
- 3 torusa (kreirana pomoću funkcije gluSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings)

Sfera koja predstavlja jezgro se crta funkcijom *draw_core*, torus koji predstavlja orbitu se crta funkcijom *draw_orbit*, a ostale sfere koje prestavljaju elektrone se crtaju funkcijom *draw_electron* koja se poziva unutar funkcije *draw_orbit*. Kompletna scena se crta *display* funkcijom.

Druga specifikacija

Mijenjanje pozicije kamere se ostvaruje pomoću navigacijskih tipki. Globalne varijable *eye_x*, *eye_y* i *eye_z* specificiraju koordinate "oka" kamere. Funkcijom *special_keys* dobijamo callback tipke specijalnih karaktera tastature. Navigacijskim tipkama povećavamo/smanjujemo vrijednosti gore navedenih globalnih varijabli.

```
// Funkcija za callback pritiskom na specijalne karaktere tastature.
// Omogucava pomijeranje kamere pomocu navigacijskih tipki.
void special_keys(int key, int x, int y)
{
  switch (key) {
    case GLUT_KEY_UP:
      eye_y += 0.1;
      break;
    case GLUT_KEY_DOWN:
      eye_y -= 0.1;
      break;
    case GLUT_KEY_RIGHT:
      eye_x += 0.1;
      break;
    case GLUT_KEY_LEFT:
      eye_x -= 0.1;
      break;
    default:
      return;
  }
  glutPostRedisplay();
}
```

Pomijeranje kamere se vrši funkcijom *gluLookAt* koja se poziva unutar *display* funkcije.

Treća specifikacija

Rotacija i skaliranje se vrše unutar funkcije *draw_orbit* pozivom funkcija *glRotatef* i *glScalef*. Rotacijom se orbite kreću oko jezgra za zadani ugao. Skaliranjem se orbita i elektron povećavaju/smanjuju.

```
// Funkcija koja crta orbitu, kao i elektron, kojem proslijedimo radius i ugao
void draw_orbit(const float& rotation, const float& radius, const float& angle)
{
  GLfloat mat_ambient[] = {0.0, 0.0, 0.1, 1.0};
  GLfloat mat_diffuse[] = {0.0, 0.0, 0.3, 1.0};
  GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
  GLfloat mat_shininess[] = {100.0};

  glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

  glScalef(scale_factor, scale_factor, scale_factor);
  glRotatef(rotation, 1, 0, 0);
  glutSolidTorus(0.01, radius, 30, 30);
  draw_electron(angle, radius);
}
```

Translacija se vrši unutar funkcije *draw_orbit* pozivom funkcije *glTranslatef*. Ovo omogućava kretanje elektrona po orbiti.

```
// Funkcija za crtanje elektrona koja je pozvana u draw_orbit()
void draw_electron(const float& angle, const float& radius)
{
  GLfloat mat_ambient[] = {0.2, 0.2, 0.2, 1.0};
  GLfloat mat_diffuse[] = {0.6, 0.6, 0.6, 1.0};
  GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
  GLfloat mat_shininess[] = {100.0};

  if (!material_two_on) {
    mat_ambient[0] = mat_diffuse[0] = mat_specular[0] = mat_shininess[0] = 0.0;
    mat_ambient[1] = mat_diffuse[1] = mat_specular[1] = 0.0;
    mat_ambient[2] = mat_diffuse[2] = mat_specular[2] = 0.0;
  }

  glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
  glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
  glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
  glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);

  float x = cos(angle);
  float y = sin(angle);
  glTranslatef(radius*x, radius*y, 0);

  GLUquadric *quad = gluNewQuadric();
  ice_texture();
  gluQuadricTexture(quad, 1);
  gluSphere(quad, 0.08, 15, 15);
  glDisable(GL_TEXTURE_2D);
}
```

Četvrta specifikacija

Tasteri miša pokreću različite animacije. Lijevi taster pokreće animaciju rotacijom, desni pokreće animaciju skaliranjem a srednji pokreće animaciju translacijom. Sve animacije se pokreću/zaustavljaju kada je taster pritisnut (state = GLUT_DOWN).

```
// Funkcija za callback pritiskom tastera misa.
// Lijevi klik pokrece rotaciju, desni skaliranje i srednji translaciju.
void mouse(int button, int state, int x, int y)
{
  switch (button) {
    case GLUT_LEFT_BUTTON:
      if (state == GLUT_DOWN) rotate_animation = !rotate_animation;
      break;
    case GLUT_RIGHT_BUTTON:
      if (state == GLUT_DOWN) scale_animation = !scale_animation;
      break;
    case GLUT_MIDDLE_BUTTON:
      if (state == GLUT_DOWN) translate_animation = !translate_animation;
      break;
    default:
      break;
  }
}
```

Da li će jedna od ovih animacija biti pokrenuta određujuju globalne varijable *rotate_animation*, *scale_animation* i *translate_animation*.

Promjena smijera kretanja elektrona je implementirana unutar funkcije *passive_mouse_motion.* Ako kursor prelazi polovinu ekrana po x osi, elektroni mijenjaju smijer kretanja. Globalna varijabla *theta* predstavlja pomjeraj animacije transliranja.

```
// Promjena smijera kretanja elektrona kada kursor predje polovinu
// ekrana po x osi
void passive_mouse_motion(int x, int y)
{
  if (x < 500 && theta > 0 || x >= 500 && theta < 0) theta *= -1;
}
```

Peta specifikacija

Funkcijom *idle* održava se animacija kada niti jedna tipka nije pritisnuta. Ponovo, da li će anmiacija biti pokrenuta, određeno je globalnim varijablama *rotate_animation*, *scale_animation* i *translate_animation*.

```
// Omoguci odrzavanje animacija kada niti jedna tipka nije pritisnuta
void idle()
{
  if (rotate_animation) rotation += 0.5;
  if (translate_animation) angle += theta;
  if (scale_animation) {
    if (scale_factor >= 1.5 || scale_factor <= 1) delta *= -1;
    scale_factor += delta;
  }
  glutPostRedisplay();
}
```

Globalna varijabla *scale_factor* predstavlja koeficjent skaliranja, te su ovoj funkciji postavljenje određene granice izvan kojih ne smije izlaziti. Globalna varijabla *delta* predstavlja pomjeraj za skaliranje.

Šesta specifikacija

Unutar datoteke *textures* nalaze C kod fajlovi *hex.c* i *ice.c* za teksturiranje oblika. Učitavanje ovih tekstura vrši se u sljedeće dvije funkcije

```
// Ucitava teksture heksagona
void hex_texture()
{
  static int texture = 0;
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, texture);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, gimp_image_hex.width, gimp_image_hex.height,
               0, GL_RGBA, GL_UNSIGNED_BYTE, gimp_image_hex.pixel_data);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
}

// Ucitava teksture leda
void ice_texture()
{
  static int texture = 0;
  glEnable(GL_TEXTURE_2D);
  glBindTexture(GL_TEXTURE_2D, texture);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, gimp_image_ice.width, gimp_image_ice.height,
               0, GL_RGBA, GL_UNSIGNED_BYTE, gimp_image_ice.pixel_data);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
}
```

Teksturiranje jezgra se vriši u sljedećem dijelu funkcije *draw_core*

```
GLUquadric *quad = gluNewQuadric();
hex_texture();
gluQuadricTexture(quad, 1);
gluSphere(quad, 0.23, 50, 50);
glDisable(GL_TEXTURE_2D);
```

a teksturiranje elektrona u funkciji *draw_electron*

```
GLUquadric *quad = gluNewQuadric();
ice_texture();
gluQuadricTexture(quad, 1);
gluSphere(quad, 0.08, 15, 15);
glDisable(GL_TEXTURE_2D);
```

Sedma specifikacija

Imamo dva izvora svjetlosti, jedno je van modela atoma a drugo unutar jezgra. Svjetlo s vana ima ambijentalnu, difuznu i spekularnu komponentu svjetla dok svjetlo jezgra ima samo ambijentalnu i difuznu komponentu.

```
// Svjetlo kamere
void camera_light()
{
  GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
  GLfloat white_light[] = {1.0, 1.0, 1.0, 1.0};
  GLfloat no_light[] = {0, 0, 0, 0};

  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glLightfv(GL_LIGHT0, GL_AMBIENT, white_light);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);
  glLightfv(GL_LIGHT0, GL_SPECULAR, white_light);
}

// Svjetlo jezgra
void core_light()
{
  GLfloat light_position[] = {0.0, 0.0, 0.0, 0.1};
  GLfloat white_light[] = {1.0, 1.0, 0.6, 1.0};
  GLfloat no_light[] = {0, 0, 0, 0};

  glLightfv(GL_LIGHT1, GL_POSITION, light_position);
  glLightfv(GL_LIGHT1, GL_AMBIENT, white_light);
  glLightfv(GL_LIGHT1, GL_DIFFUSE, white_light);
  glLightfv(GL_LIGHT1, GL_SPECULAR, no_light);
}
```

Jezgro se sastoji od jednog materijala a elektroni od drugog. Oba materijala reaguju na sve tri komponente svjetla. Implementacija materijala jezgra je data sljedećim kodom

```
GLfloat mat_ambient[] = {0.3, 0.3, 0.3, 1.0};
GLfloat mat_diffuse[] = {0.8, 0.8, 0.8, 1.0};
GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat mat_shininess[] = {100.0};

if (!material_one_on) {
    mat_ambient[0] = mat_diffuse[0] = mat_specular[0] = mat_shininess[0] = 0.0;
    mat_ambient[1] = mat_diffuse[1] = mat_specular[1] = 0.0;
    mat_ambient[2] = mat_diffuse[2] = mat_specular[2] = 0.0;
}
```

Implementacija materijala elektrona je data sa

```
GLfloat mat_ambient[] = {0.2, 0.2, 0.2, 1.0};
GLfloat mat_diffuse[] = {0.6, 0.6, 0.6, 1.0};
GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
GLfloat mat_shininess[] = {100.0};

if (!material_two_on) {
    mat_ambient[0] = mat_diffuse[0] = mat_specular[0] = mat_shininess[0] = 0.0;
    mat_ambient[1] = mat_diffuse[1] = mat_specular[1] = 0.0;
    mat_ambient[2] = mat_diffuse[2] = mat_specular[2] = 0.0;
}
```

Svjetlo s vana palimo/gasimo tipkom tastature “0” a svjetlo jezgra tipkom “1”. Materijal jezgra palimo/gasimio tipkom “2” a materijal elektrona tipkom “3”. Da li će svjetla i materijali biti upaljeni ili ugašeni, određuje se globalnim varijablama *light_one_on, light_two_on, material_one_on, material_two_on*. Implementacija je data u sljedećoj funkciji.

```
// Omogucava paljenje/gasenje svjetala i materijala
void keyboard(unsigned char key, int x, int y)
{
  switch (key) {
    case 27:
      exit(0);
      break;
    case '0':
      light_one_on ? glDisable(GL_LIGHT0) : glEnable(GL_LIGHT0);
      light_one_on = !light_one_on;
      break;
    case '1':
      light_two_on ? glDisable(GL_LIGHT1) : glEnable(GL_LIGHT1);
      light_two_on = !light_two_on;
      break;
    case '2':
      material_one_on = !material_one_on;
      break;
    case '3':
      material_two_on = !material_two_on;
      break;
    default:
      break;
  }
  glutPostRedisplay();
}
```