OpenGL勉強会
第三回

慶應義塾大学大学院
システムデザイン・マネジメント研究科
立山　義祐

今回学ぶOpenGLテクニック

- ライト
  - ライトの設定
  - 法線ベクトル（normal vectors）
- テクスチャ

今回使うサンプルプログラム

- gltest11.c ライトの位置を変更する
- gltest12.c ライトの色、複数のライト
- gltest13.c 材質の設定
- gltest14.c, gltest15.c 法線ベクトルの設定
- gltest16.c テクスチャを貼る前
- gltest17.c テクスチャを貼った後
- gltest18.c テクスチャの繰り返し
- gltest19.c MIPMAP

gltest11.c ライトの位置を変更

```
float g_r = 0.0f;
float g_r2 = 0.0f;

GLfloat light0pos[] = {0.0f, 0.0f, 0.0f, 1.0f};

void
display(void)
{
  int i;

  glClear(GL_COLOR_BUFFER_BIT |
          GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
  glEnable(GL_DEPTH_TEST);
  gluLookAt(0.0, 0.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  light0pos[0] = (float)(30.0 * cos(g_r2 * M_PI / 180.0));
  light0pos[1] = (float)(30.0 * sin(g_r2 * M_PI / 180.0));
  light0pos[2] = 10.0f;
  glLightfv(GL_LIGHT0, GL_POSITION, light0pos);
 //glRotatef(g_r, 0.0f, 1.0f, 0.0f);
  glutSolidTeapot(5.0f);

  glutSwapBuffers();
}
```

```
void
idle(void)
{
  g_r = g_r + 10.0f;
  g_r2 = g_r2 + 15.0f;

  if (g_r >= 360.0f) {
    g_r = g_r - 360.0f;
  }

  if (g_r2 >= 360.0f) {
    g_r2 = g_r2 - 360.0f;
  }

  glutPostRedisplay();
  usleep(50000);
}
```

補足： C言語のマクロ（macro）

- コンパイラがコンパイルする前にソースコードを処理（変形）する
- #include <stdio.h>
- #define M_PI 3.14159265358979323846
- #if ～ #else～#endif

```
#define VERBOSE

#ifdef VERBOSE
      printf(“debug: value = %d¥n”, value);
#endif
```

```
#if 0
      process_a();
#endif
```


gltest12.c ライトの色、複数のライト

```
float g_r = 0.0f;
float g_r2 = 0.0f;

GLfloat light0pos[] = {0.0f, 0.0f, 0.0f, 1.0f};
GLfloat light1pos[] = {0.0f, 0.0f, 0.0f, 1.0f};
GLfloat red[] = {0.8f, 0.0f, 0.0f, 1.0f};
GLfloat green[] = {0.0f, 0.8f, 0.0f, 1.0f};
GLfloat blue[] = {0.0f, 0.0f, 0.8f, 1.0f};

void
display(void)
{
  int i;

  glClear(GL_COLOR_BUFFER_BIT |
        GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();
  glEnable(GL_DEPTH_TEST);
  gluLookAt(0.0, 0.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

  light0pos[0] = (float)(30.0 * cos(g_r2 * M_PI / 180.0));
  light0pos[1] = (float)(30.0 * sin(g_r2 * M_PI / 180.0));
  light0pos[2] = 10.0f;
  glLightfv(GL_LIGHT0, GL_POSITION, light0pos);
  glLightfv(GL_LIGHT0, GL_DIFFUSE, red);

  light1pos[0] = (float)(30.0 * cos(g_r * M_PI / 180.0));
  light1pos[1] = (float)(30.0 * sin(g_r * M_PI / 180.0));
  light1pos[2] = 10.0f;
  glLightfv(GL_LIGHT1, GL_POSITION, light1pos);
  glLightfv(GL_LIGHT1, GL_DIFFUSE, blue);

  //glTranslatef(0.0, 0.0, -10.0f);
  //glRotatef(g_r, 0.0f, 1.0f, 0.0f);

  glutSolidTeapot(5.0f);

  glutSwapBuffers();
}

void
init(void)
{
  //glClearColor(1.0, 1.0, 1.0, 0.0);
  glClearColor(0.0, 0.0, 0.0, 0.0);

  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
  glEnable(GL_LIGHT1);
}
```

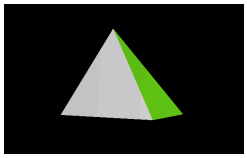
gltest13.c 材質の設定

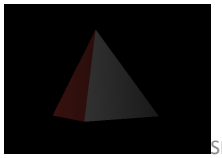
```
GLfloat light0pos[] = {0.0f, 0.0f, 10.0f, 1.0f};

GLfloat red[] = {0.8f, 0.0f, 0.0f, 1.0f};
GLfloat green[] = {0.0f, 0.8f, 0.0f, 1.0f};
GLfloat blue[] = {0.0f, 0.0f, 0.8f, 1.0f};
GLfloat yellow[] = {0.8f, 0.8f, 0.0f, 1.0f};
GLfloat yellow2[] = {0.1f, 0.1f, 0.0f, 1.0f};

void
display(void)
{
 int i;

 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glLoadIdentity();
 glEnable(GL_DEPTH_TEST);
 gluLookAt(0.0, 0.0, 50.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

 glLightfv(GL_LIGHT0, GL_POSITION, light0pos);

 glRotatef(g_r, 0.0f, 1.0f, 0.0f);
```

```
 glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, yellow);
 glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 30.0f);
 glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, yellow);

 glutSolidTeapot(5.0f);

 glutSwapBuffers();
}
```

gltest14.c ライティング : 法線ベクトルの設定前 (difference from gltest4.c)

```
float p1[] = {1.0f, -.5f, 1.0f},
  p2[] = {0.0f, 1.0f, 0.0f},
  p3[] = {-1.0f, -.5f, 0.0f},
  p4[] = {1.0f, -.5f, -1.0f};

void
display(void)
{
 int i;
 float norm[3], v1[3], v2[3];

 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glLoadIdentity();
 glEnable(GL_DEPTH_TEST);
 gluLookAt(0.0, 0.0, 10.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

 glLightfv(GL_LIGHT0, GL_POSITION, light0pos);

 //glTranslatef(0.0, 0.0, -10.0f);
 glRotatef(g_r, 0.0f, 1.0f, 0.0f);
```





```
 glBegin(GL_TRIANGLES);

 // face 1
 //glColor3f(1.0f, 1.0f, 1.0f); // White
 glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, white);
 glVertex3fv(p1);
 glVertex3fv(p2);
 glVertex3fv(p3);

 // face 2
 //glColor3f(1.0f, 0.0f, 0.0f); // Red
 glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, red);
 glVertex3fv(p3); // -1.0f, -.5f, 0.0f);
 glVertex3fv(p2); // 0.0f, 1.0f, 0.0f);
 glVertex3fv(p4); // 1.0f, -.5f, -1.0f);

 // face 3
 //glColor3f(0.0f, 1.0f, 0.0f); // Green
 glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, green);
 glVertex3fv(p4); // 1.0f, -.5f, -1.0f);
 glVertex3fv(p2); // 0.0f, 1.0f, 0.0f);
 glVertex3fv(p1); // 1.0f, -.5f, 1.0f);

 glEnd();

 glutSwapBuffers();
```

gltest15.c 法線ベクトルの設定後

```
glBegin(GL_TRIANGLES);

// face 1
//glColor3f(1.0f, 1.0f, 1.0f); // White
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, white);
vec_subtract(v1, p2, p1);
vec_subtract(v2, p3, p1);
vec_cross_product(norm, v1, v2);
glNormal3fv(norm);
glVertex3fv(p1);
glNormal3fv(norm);
glVertex3fv(p2);
glNormal3fv(norm);
glVertex3fv(p3);

// face 2
//glColor3f(1.0f, 0.0f, 0.0f); // Red
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, red);
vec_subtract(v1, p2, p3);
vec_subtract(v2, p4, p3);
vec_cross_product(norm, v1, v2);
glNormal3fv(norm);
glVertex3fv(p3); // -1.0f, -.5f, 0.0f);
glNormal3fv(norm);
glVertex3fv(p2); // 0.0f, 1.0f, 0.0f);
glNormal3fv(norm);
glVertex3fv(p4); // 1.0f, -.5f, -1.0f);

// face 3
//glColor3f(0.0f, 1.0f, 0.0f); // Green
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, green);
vec_subtract(v1, p2, p4);
vec_subtract(v2, p1, p4);
vec_cross_product(norm, v1, v2);
glNormal3fv(norm);
glVertex3fv(p4); // 1.0f, -.5f, -1.0f);
glNormal3fv(norm);
glVertex3fv(p2); // 0.0f, 1.0f, 0.0f);
glNormal3fv(norm);
glVertex3fv(p1); // 1.0f, -.5f, 1.0f);

glEnd();
```


ベクトル計算ルーチン：外積と引き算

```
void
vec_cross_product(float ret[3], float a[3], float b[3])
{
  float x, y, z, len;

  x = a[1]*b[2] - a[2]*b[1];
  y = a[2]*b[0] - a[0]*b[2];
  z = a[0]*b[1] - a[1]*b[0];

  len = (float)sqrt(x * x + y * y + z * z);

  ret[0] = x / len;
  ret[1] = y / len;
  ret[2] = z / len;
}
```

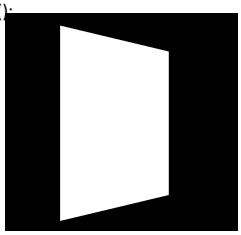
```
void
vec_subtract(float ret[3], float a[3], float b[3])
{
  ret[0] = a[0] - b[0];
  ret[1] = a[1] - b[1];
  ret[2] = a[2] - b[2];
}
```

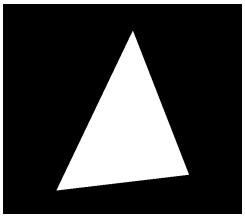
gltest16.c テクスチャを貼る前

gltest16.c ← gltest3.c

```
void
display(void)
{
  int i;

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0f, 1.0f, 1.0f);
  glLoadIdentity();
  glTranslatef(0.0, 0.0, -5.0f);
  glRotatef(g_r, 0.0f, 1.0f, 0.0f);

  glBegin(GL_POLYGON);
  glVertex3f(-1.0f, -1.0f, 0.0f);
  glVertex3f( 1.0f, -1.0f, 0.0f);
  glVertex3f( 1.0f,  1.0f, 0.0f);
  glVertex3f(-1.0f,  1.0f, 0.0f);
  glEnd();

  glutSwapBuffers();
}
```

```
void
display(void)
{
  int i;

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0f, 1.0f, 1.0f);
  glLoadIdentity();
  glTranslatef(0.0, 0.0, -5.0f);
  glRotatef(g_r, 0.0f, 1.0f, 0.0f);

  glBegin(GL_TRIANGLES);
  glVertex3f(1.0f, -.5f, 0.0f);
  glVertex3f(0.0f, 1.0f, 0.0f);
  glVertex3f(-1.0f, -.5f, 0.0f);
  glEnd();

  glutSwapBuffers();
}
```



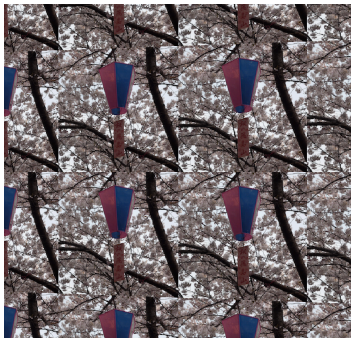


gltest17.c テクスチャを貼る

```
void
init(void)
{
  char *texture;
  int width, height;

  glClearColor(0.0, 0.0, 0.0, 0.0);

  glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

  glTexParameteri(GL_TEXTURE_2D,
        GL_TEXTURE_MAG_FILTER, GL_NEAREST);
  glTexParameteri(GL_TEXTURE_2D,
        GL_TEXTURE_MIN_FILTER, GL_NEAREST);

  texture = pbm_load("asukayama256.pbm", &width,
        &height);

  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
            GL_RGB, GL_UNSIGNED_BYTE, texture);

  pbm_free(texture);
}
```

```
void
display(void)
{
  int i;

  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0f, 1.0f, 1.0f);
  glLoadIdentity();
  glTranslatef(0.0, 0.0, -5.0f);
  glRotatef(g_r, 0.0f, 1.0f, 0.0f);

  glEnable(GL_TEXTURE_2D);

  glBegin(GL_POLYGON);
  glTexCoord2d(0.0, 1.0);
  glVertex3f(-1.0f, -1.0f, 0.0f);
  glTexCoord2d(1.0, 1.0);
  glVertex3f( 1.0f, -1.0f, 0.0f);
  glTexCoord2d(1.0, 0.0);
  glVertex3f( 1.0f,  1.0f, 0.0f);
  glTexCoord2d(0.0, 0.0);
  glVertex3f(-1.0f,  1.0f, 0.0f);
  glEnd();

  glutSwapBuffers();
}
```

gltest18.c テクスチャの繰り返し

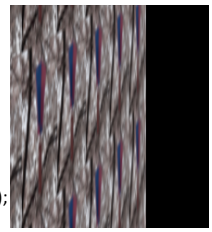


```
void
display(void)
{
 int i;

 glClear(GL_COLOR_BUFFER_BIT);
 glColor3f(1.0f, 1.0f, 1.0f);
 glLoadIdentity();
 glTranslatef(0.0, 0.0, -5.0f);
 glRotatef(g_r, 0.0f, 1.0f, 0.0f);

 glEnable(GL_TEXTURE_2D);

 glBegin(GL_POLYGON);
 glTexCoord2d(0.0, 16.0);
 glVertex3f(-8.0f, -8.0f, 0.0f);
 glTexCoord2d(16.0, 16.0);
 glVertex3f( 8.0f, -8.0f, 0.0f);
 glTexCoord2d(16.0, 0.0);
 glVertex3f( 8.0f,  8.0f, 0.0f);
 glTexCoord2d(0.0, 0.0);
 glVertex3f(-8.0f,  8.0f, 0.0f);
 glEnd();

 glutSwapBuffers();
}
```

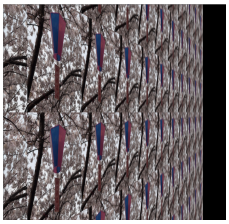

gltest19.c MIPMAP

gltest19.c ← gltest18.c

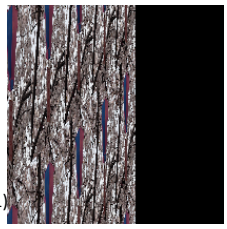
```
void
init(void)
{
 char *texture;
 int width, height;

 glClearColor(0.0, 0.0, 0.0, 0.0);

 glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
```



```
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
       GL_LINEAR);
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
     GL_LINEAR_MIPMAP_LINEAR);

 texture = pbm_load("asukayama256.pbm", &width, &height);

 gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, width, height,
   GL_RGB, GL_UNSIGNED_BYTE, texture);

 pbm_free(texture);

}
```



```
void
init(void)
{
 char *texture;
 int width, height;

 glClearColor(0.0, 0.0, 0.0, 0.0);

 glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
```



```
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
       GL_NEAREST);
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
     GL_NEAREST);

 texture = pbm_load("asukayama256.pbm", &width, &height);

 glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0,
         GL_RGB, GL_UNSIGNED_BYTE, texture);

 pbm_free(texture);
}
```