リスト 6.1 「slBumpMap1.cpp」の一部

```
void makeNormalMap(Bitmap *bmp)
{//法線マップ用配列作成
  int i, j, ip, jp;
  float r, g;

  if(bmp->bi.Width != TEX_WIDTH || bmp->bi.Height != TEX_HEIGHT)
  {
    printf("TEX_WIDTH,TEX_HEIGHTとファイルサイズを一致させてください¥n");
    return;
  }
  //BMPファイルから高さマップをロード
  for(j = 0; j < TEX_HEIGHT; j++)
  {
    for(i = 0; i < TEX_WIDTH; i++)
    {
      //高度マップを取得
      if(bmp->bi.BitCount <= 24)
      normalMap[j][i][2] = (bmp->pixel[i+j * TEX_WIDTH]).r;
      else//32bit color
      normalMap[j][i][2] = (bmp->pixelA[i+j * TEX_WIDTH]).r;
    }
  }

  //高さマップから法線マップを作成
  for(j = 0; j < TEX_HEIGHT; j++)
  {
    jp = j+1;
    if(jp >= TEX_HEIGHT) jp = 0;
    for(i = 0; i < TEX_WIDTH; i++)
    {
      ip = i+1;
      if(ip >= TEX_WIDTH) ip = 0;
      float deltaS = inv * (normalMap[j][ip][2] - normalMap[j][i][2]);
      float deltaT = inv * (normalMap[jp][i][2] - normalMap[j][i][2]);
      r = 127.5 + deltaS * fGrad;
      if(r < 0.0) r = 0.0;
      if(r > 255.0) r = 255.0;
      g = 127.5 + deltaT * fGrad;
      if(g < 0.0) g = 0.0;
      if(g > 255.0) g = 255.0;
      normalMap[j][i][0] = (GLubyte)r;
      normalMap[j][i][1] = (GLubyte)g;
      if(inv == -1.0)   normalMap[j][i][2] = 255 - normalMap[j][i][2];
    }
  }
}

void setNormalMap(int n)
{
  glBindTexture(GL_TEXTURE_2D, texName[n]);//テクスチャをバインドする

glTexImage2D(GL_TEXTURE_2D,0,GL_RGB,TEX_WIDTH,TEX_HEIGHT,0,GL_RGB,GL_UNSIGNED_
BYTE,normalMap);
  glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T, GL_REPEAT);
  glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
  glBindTexture(GL_TEXTURE_2D, 0);//バインドを解除
}

void display(void)
{
  //時間計測
```

```
  static double time1, time2, drawTime, frame;
  if(ang <= 0.001) time1 = timeGetTime();

  //カラーバッファ,デプスバッファのクリア
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glLoadIdentity();//視点を変えるときはこの位置に必要

  if(cos(M_PI * view.theta /180.0) >= 0.0)//カメラ仰角度でﾋﾞｭｰｱｯﾌﾟﾍﾞｸﾄﾙ切替
    gluLookAt(view.pos[0], view.pos[1], view.pos[2], view.cnt[0], view.cnt[1],
view.cnt[2], 0.0, 1.0, 0.0);
  else
    gluLookAt(view.pos[0], view.pos[1], view.pos[2], view.cnt[0], view.cnt[1],
view.cnt[2], 0.0, -1.0, 0.0);
  //光源設定//'l'を押した後光源位置可変
  glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

  glActiveTexture(GL_TEXTURE0);
  glBindTexture(GL_TEXTURE_2D, texName[0]);
  glActiveTexture(GL_TEXTURE1);
  glBindTexture(GL_TEXTURE_2D, texName[1]);

  // シェーダ・プログラムの適用
  glUseProgram(shaderProg);
  //描画
  //fragment shaderのユニフォーム変数texのインデックスを取得
  GLint texNormalLoc = glGetUniformLocation(shaderProg, "normalMap");
  glUniform1i(texNormalLoc, 0);//GL_TEXTURE0を適用
  draw0();
  glUniform1i(texNormalLoc, 1);//GL_TEXTURE1を適用
  draw1();
  // シェーダ・プログラムの適用を解除
  glUseProgram(0);
  drawFloor(10.0, 10.0, 10, 10);
  //影
  drawShadow();

//中略
  //終了
  glutSwapBuffers();
}

//-------------------------------------------------------------------------
--
//ﾊﾞﾝﾌﾟﾏｯﾋﾟﾝｸﾞ用基本立体
//球のバンプ・マッピング用(他は省略)
void drawBumpSphere(double radius, int nSlice, int nStack, int nRepeatS, int
nRepeatT)
{
  int i, j;
  double s, t0, t1, r0, r1, th0, th1, phi;
  double p[2][3], tnt[3];

  GLuint tangentLoc = glGetAttribLocation(shaderProg, "tangent");

  for(j = 0; j < nStack; j++)
  {
    //j=0は北極点(x=0, y=0, z=radius)
    //2つのt座標
    t0 = (double)j / (double)nStack;
    t1 = (double)(j+1) / (double)nStack;
    //これらの天頂角
    th0 = M_PI * t0;
    th1 = M_PI * t1;
    //x-y平面に投影した半径
    r0 = radius * sin(th0);
```

```
    r1 = radius * sin(th1);
    //頂点z座標
    p[0][2] = radius * cos(th0);
    p[1][2] = radius * cos(th1);
    //接線ベクトルのｚ成分
    tnt[2] = 0.0;

    //南極点を0,北極点を1とするt座標
    t0 = (1.0 - t0) * nRepeatT;
    t1 = (1.0 - t1) * nRepeatT;

    glBegin(GL_QUAD_STRIP);
    for(i = 0; i <= nSlice; i++)
    {
      //s座標
      s = (double)i / (double)nSlice;
      phi = -M_PI + 2.0 * M_PI * s;
      //頂点のxy座標(i=0を真後ろ)
      p[0][0] = r0 * cos(phi);//x座標
      p[0][1] = r0 * sin(phi);//y座標
      p[1][0] = r1 * cos(phi);//x座標
      p[1][1] = r1 * sin(phi);//y座標
      //接線ベクトルのxy成分
      tnt[0] =-sin(phi);
      tnt[1] = cos(phi);

      s *= nRepeatS;

      glVertexAttrib3dv(tangentLoc, tnt);
      glTexCoord2d(s, t0);//テクスチャ座標
      glNormal3dv(p[0]);//法線ベクトル,正規化すれば頂点座標に同じ
      glVertex3dv(p[0]);//頂点座標

      glTexCoord2d(s, t1);//テクスチャ座標
      glNormal3dv(p[1]);//法線ベクトル,正規化すれば頂点座標に同じ
      glVertex3dv(p[1]);//頂点座標
    }
    glEnd();
  }
}
```