映り込み用のフレームバッファオブジェクトの準備

```
/*
** フレームバッファオブジェクトのサイズ
*/
#define FBOWIDTH   1024      // フレームバッファオブジェクトの幅
#define FBOHEIGHT  1024      // フレームバッファオブジェクトの高さ
…

  // カラーバッファ用のテクスチャを用意する
  GLuint cb;
  glGenTextures(1, &cb);
  glBindTexture(GL_TEXTURE_2D, cb);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, FBOWIDTH, FBOHEIGHT, 0, GL_RGBA, GL_UNSIGNED_BYTE, 0);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

  // デプスバッファ用のレンダーバッファを用意する
  GLuint rb;
  glGenRenderbuffers(1, &rb);
  glBindRenderbuffer(GL_RENDERBUFFER, rb);
  glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH_COMPONENT, FBOWIDTH, FBOHEIGHT);

  // 映り込み用のフレームバッファオブジェクトを作成する
  GLuint fb;
  glGenFramebuffers(1, &fb);
  glBindFramebuffer(GL_FRAMEBUFFER, fb);
  glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, cb, 0);
  glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_RENDERBUFFER, rb);
```

シャドウマッピング用のフレームバッファオブジェクトの準備

```
  // デプスバッファ用のテクスチャを用意する
  GLuint db;
  glGenTextures(1, &db);
  glBindTexture(GL_TEXTURE_2D, db);
  glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, FBOWIDTH, FBOHEIGHT, 0, GL_DEPTH_COMPONENT, GL_UNSIGNED_BYTE, 0);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
  glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

```
// 書き込むポリゴンのテクスチャ座標値のＲとテクスチャとの比較を行うようにする
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, GL_COMPARE_REF_TO_TEXTURE);
// もしＲの値がテクスチャの値以下なら真（つまり日向）
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_FUNC, GL_LEQUAL);

// シャドウマッピング用のフレームバッファオブジェクトを作成する
GLuint sb;
glGenFramebuffers(1, &sb);
glBindFramebuffer(GL_FRAMEBUFFER, sb);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, db, 0);
// カラーバッファは無いので描かない
glDrawBuffer(GL_NONE);
// カラーバッファは無いので読まない
glReadBuffer(GL_NONE);
```

シャドウマッピング用の変換行列の準備

```
// シーン全体を収める投影変換行列
GgMatrix mq = ggPerspective(0.3f, 1.0f, 5.0f, 9.0f);

// 光源を視点にした視野変換行列
GgMatrix ml = ggLookat(pl[0], pl[1], pl[2], 0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f);

// シャドウマッピング用の変換行列
GgMatrix ms = ggTranslate(0.5f, 0.5f, 0.5f) * ggScale(0.5f, 0.5f, 0.5f) * mq * ml;
```

シャドウマップの作成

```
  // ビューポートの保存と設定
  ground.saveViewport();
  glViewport(0, 0, FBOWIDTH, FBOHEIGHT);

  // フレームバッファオブジェクトを結合する
  glBindFramebufferEXT(GL_FRAMEBUFFER, sb);

  // デプスバッファをクリア
  glClear(GL_DEPTH_BUFFER_BIT);

  // シャドウマップの作成
  normal.use();
  normal.setLightPosition(pl);
  normal.loadMatrix(mq, ml * mm * tb.get());
```

```
    // フレームバッファオブジェクトを結合する
    glBindFramebuffer(GL_FRAMEBUFFER, fb);

    …
```

シャドウマップを参照して描画

```
    // 床面の描画
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, cb);
    glActiveTexture(GL_TEXTURE1);
    glBindTexture(GL_TEXTURE_2D, db);
    ground.use();
    ground.loadShadowMatrix(ms.rotateX(-1.5707963f));
    ground.loadMatrix(mp, mv.rotateX(-1.5707963f));
    rectangle->draw();
```

バーテックスシェーダ

```
// テクスチャ座標
out vec2 ctex;
out vec4 dtex;

void main(void)
{
  …
  ctex = pv.xy;     // ウィンドウ全体に矩形を表示するので頂点の座標をそのままテクスチャ座標に使う
  dtex = ms * pv;   // シャドウマップは頂点に視点から見たときの変換行列をかけた座標を使って参照する

  gl_Position = mc * pv;
}
```

フラグメントシェーダ

```
void main(void)
{
  // 市松模様
  vec4 a = mix(c1, c2, mod(floor(ctex.x * 4.0) + floor(ctex.y * 4.0), 2.0));
  vec2 p = gl_FragCoord.xy / vp.zw;
  vec4 c = texture(cmap, p);

  fc = (iamb + idiff * textureProj(dmap, dtex)) * a + ispec + kspec * c;
}
```

映り込みをぼかす

```
// 映り込みのぼけ具合
const float sigma = 0.005;

…

void main(void)
{
  // 市松模様
  vec4 a = mix(c1, c2, mod(floor(ctex.x * 4.0) + floor(ctex.y * 4.0), 2.0));
  vec2 p = gl_FragCoord.xy / vp.zw;
  vec4 c = texture(cmap, p);

  for (int i = 0; i < 16; ++i) c += texture(cmap, p + rn[i] * sigma);
  fc = (iamb + idiff * textureProj(dmap, dtex)) * a + ispec + kspec * c / 16.0;
}
```