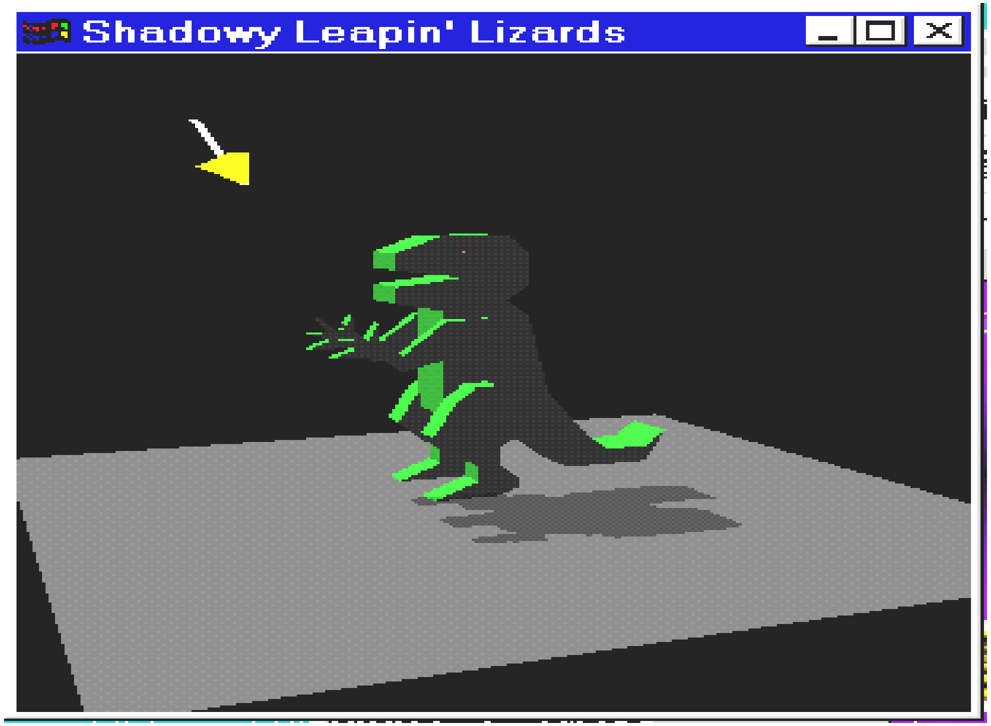
Shadows

Idea

- If :object can be seen from light position-lightened object
- Else: object is in the shadow



Physical nature of shadows

- Umbra
  - Area of the shadowed object that is not visible from any part of the light source
- Penumbra
  - Area of the shadowed object that can receive some portion of light

Physical nature of shadows

- Umbra
  - Area of the shadowed object that is not visible from any part of the light source
- Penumbra
  - Area of the shadowed object that can receive some portion of light

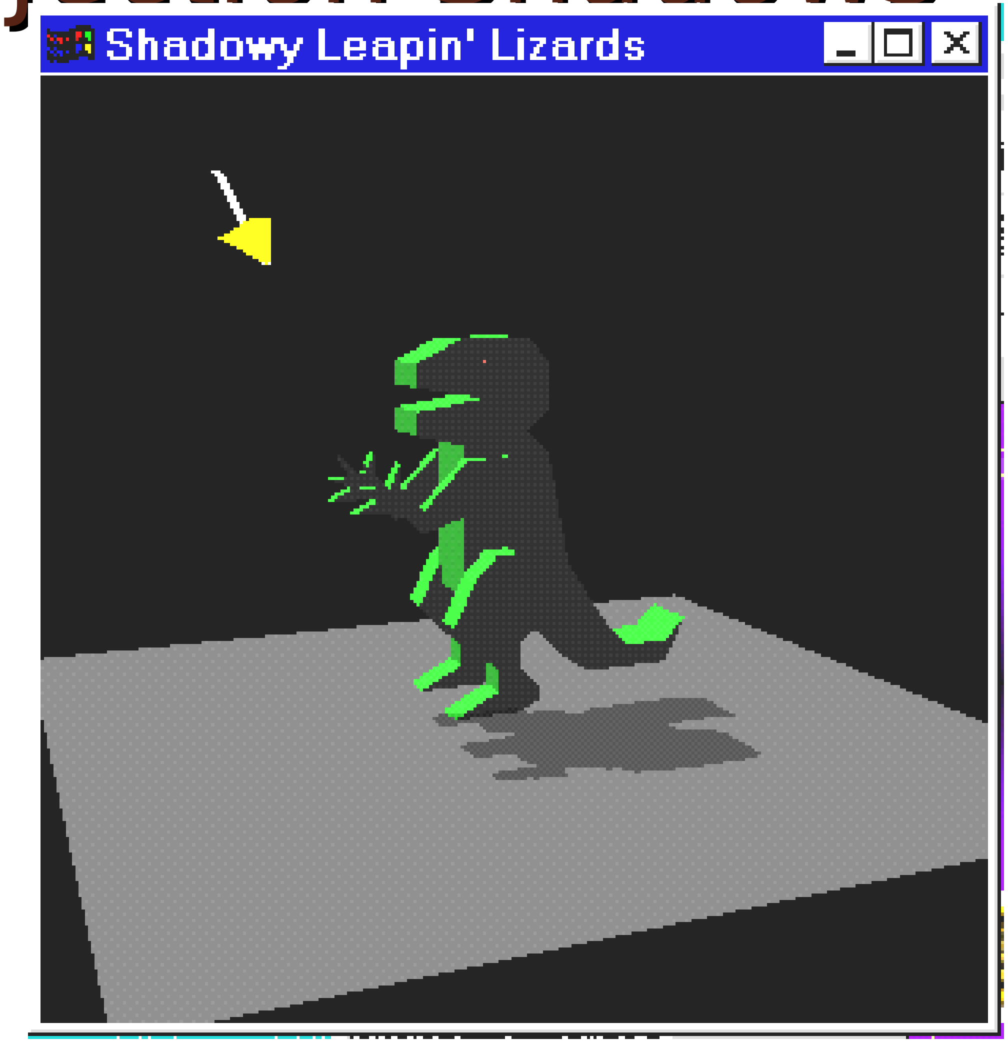
Ways to Implement

- Projection Shadows
  - Shadow is projected into the plane of the floor
- Shadow Volumes
  - “Shadow” volume projected by object
    from the light source.
- Shadow Maps
  - Shadow is created via testing whether pixel is visible from the light
- Creating Black Square Under Object ☺

Ways to Implement

- Projection Shadows (Planar shadows)
  - Shadow is projected into the plane of the floor
    - Object is projected into the plane of the floor
    - then rendered as a separate primitive
  - Applying this shadow is similar to decaling a polygon with another coplanar one
- Shadow Volumes
  - “Shadow” volume projected by object from the light source.
- Shadow Maps
  - Shadow is created via testing whether pixel is visible from the light
- Creating Black Square Under Object ☺

Projection Shadows



Shadow is projected into the plane of the floor.

Projection Shadows

- +
  - Easy to implement
  - Cross-platform way
- -
  - Difficult to use shadow onto anything other than flat surfaces
    - carefully cast the shadow onto the plane of each polygon face
    - cliping the result to the polygon's boundaries
    - Object splitting may be needed
  - There are limits to how well you can control the shadow's color

Complicated calculations

Minuses are not important for HW ☺

Projection Shadows

- Uses projection transformations
- Shadowing object is projected to some surface, related to shadowed object
  - involves applying a orthographic or perspective projection matrix to the modelview transform
- Visualized as separated primitive
- 2-tier shadow calculations
  - Matrix projection
    - applying an orthographic or perspective projection matrix to the modelview
  - Visualization of the object with proper color
    - rendering the projected object in the desired shadow color
- Ways:
  - Construction is done via a sequence of transforms
  - Construct a projection matrix directly

Projection Shadows

- Uses projection transformations
- Shadowing object is projected to some surface, related to shadowed object
  - involves applying a orthographic or perspective projection matrix to the modelview transform
- Visualized as separated primitive
- 2-tier shadow calculations
  - Matrix projection
    - applying an orthographic or perspective projection matrix to the modelview
  - Visualization of the object with proper color
    - rendering the projected object in the desired shadow color
- Ways:
  - Construction is done via a sequence of transforms
  - Construct a projection matrix directly

Render an object that has a shadow cast from a directional light on the *z* axis down onto the *x*, *y* plane:

- Render the scene, including the shadowing object in the usual way.
- Set the modelview matrix to identity, then call *glScalef1.f, 0.f, 1.f(1.f, 0.f, 1.f)*
- Make the rest of the transformation calls necessary to position and orient the shadowing object
- Set the OpenGL state necessary to create the correct shadow color
- Render the shadowing object
  - Second render
  - The transform flattens it into the object's shadow
- More: http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node192.html

Render the Shadow

```
 /* Render 50% black shadow color on top
of whatever
    the floor appearance is. */
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,
     GL_ONE_MINUS_SRC_ALPHA);
glDisable(GL_LIGHTING);  /* Force the 50%
black. */
glColor4f(0.0, 0.0, 0.0, 0.5);

glPushMatrix();
   /* Project the shadow. */
   glMultMatrixf((GLfloat *) floorShadow);
   drawDinosaur();
glPopMatrix();
```

Projection Shadows

- Uses projection transformations
- Shadowing object is projected to some surface, related to shadowed object
  - involves applying a orthographic or perspective projection matrix to the modelview transform
- Visualized as separated primitive
- 2-tier shadow calculations
  - Matrix projection
    - applying an orthographic or perspective projection matrix to the modelview
  - Visualization of the object with proper color
    - rendering the projected object in the desired shadow color
- Ways:
  - Construction is done via a sequence of transforms
  - Construct a projection matrix directly

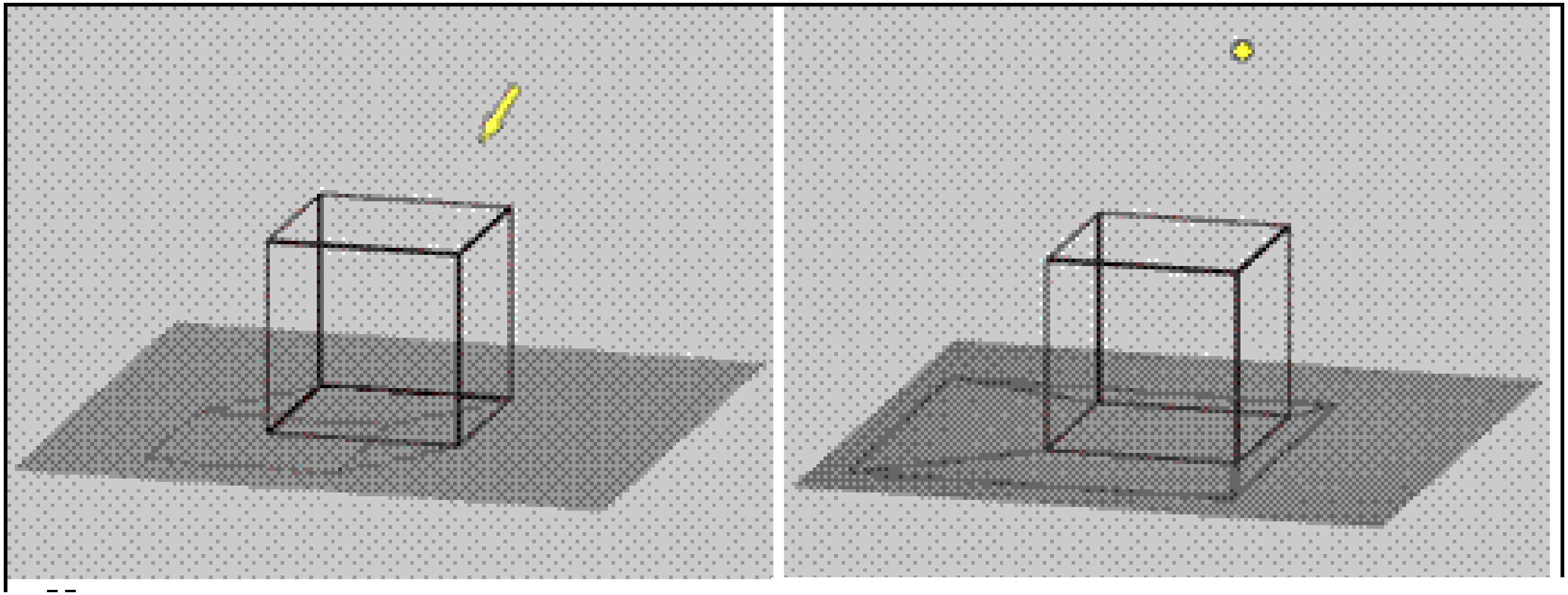
Render an object that has a shadow cast from a directional light on the *z* axis down onto the *x*, *y* plane:

- Render the scene, including the shadowing object in the usual way.
- Construct a projection matrix directly
- Set the OpenGL state necessary to create the correct shadow color
- Render the shadowing object
  - Second render
  - The transform flattens it into the object's shadow
- More: http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node192.html

Projection Matrix

- Arguments
  - Arbitrary plane in $Ax + By + Cz + D = 0$ form
  - Light position in homogeneous coordinates
    - GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
      - if light is directional - 0
        - Perpective shadow
      - 1 othervise
        - Ortho shadow
- The function concatenates the shadow matrix with the current

Projection Shadows



Shadow Ortho

Projection

Shadow

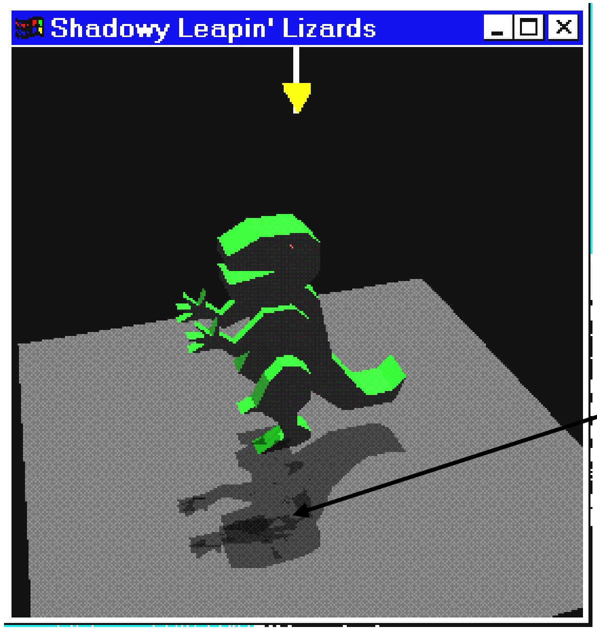
Perspective

Projection

```
void shadowMatrix(GLfloat shadowMat[4][4], GLfloat groundplane[4], GLfloat lightpos[4])
{   //  Find dot product between light position vector and ground plane normal. */
    float dot;
    float shadowMat[4][4];
    dot = ground[0] * light[0] +   //distance between light and plane
            ground[1] * light[1] +
            ground[2] * light[2] +
            ground[3] * light[3];
           shadowMat[0][0] = dot - light[0] * ground[0];
           shadowMat[1][0] = 0.0 - light[0] * ground[1];
           shadowMat[2][0] = 0.0 - light[0] * ground[2];
           shadowMat[3][0] = 0.0 - light[0] * ground[3];
           shadowMat[0][1] = 0.0 - light[1] * ground[0];
           shadowMat[1][1] = dot - light[1] * ground[1];
           shadowMat[2][1] = 0.0 - light[1] * ground[2];
           shadowMat[3][1] = 0.0 - light[1] * ground[3];
           shadowMat[0][2] = 0.0 - light[2] * ground[0];
           shadowMat[1][2] = 0.0 - light[2] * ground[1];
           shadowMat[2][2] = dot - light[2] * ground[2];
           shadowMat[3][2] = 0.0 - light[2] * ground[3];
           shadowMat[0][3] = 0.0 - light[3] * ground[0];
           shadowMat[1][3] = 0.0 - light[3] * ground[1];
           shadowMat[2][3] = 0.0 - light[3] * ground[2];
           shadowMat[3][3] = dot - light[3] * ground[3];
           glMultMatrixf((const GLfloat*)shadowMat);  //Concatination
}
```

Some Problems

Without stencil to avoid double blending of the shadow pixels:



Notice darks spots on the planar shadow.

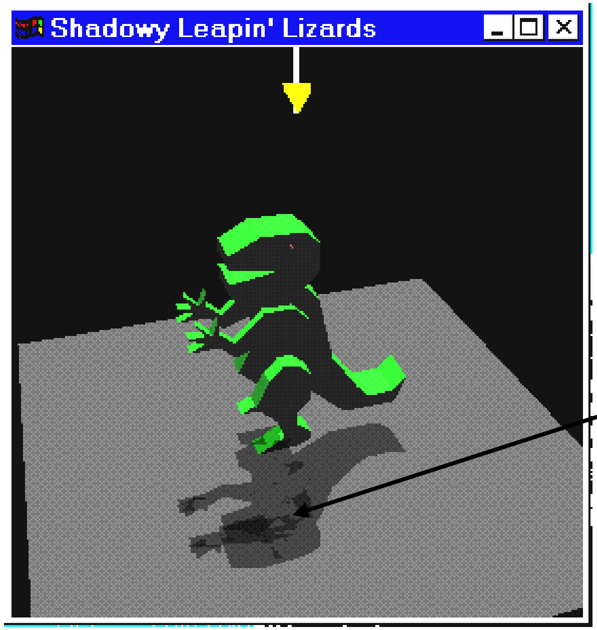
Solution: **Use S-buffer(In General)**

S-Buffer

- Per-pixel test, similar to depth buffering.
- Tests against value from stencil buffer; rejects fragment if stencil test fails.
- Distinct stencil operations performed when
  - Stencil test fails
  - Depth test fails
  - Depth test passes
- Provides fine grain control of pixel update
- glEnable/glDisable(GL_STENCIL_TEST);
- glClear(… | GL_STENCIL_BUFFER_BIT);
- …
- glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH | GLUT_STENCIL);

Some Problems

Without stencil to avoid double blending of the shadow pixels:



Notice darks spots on the planar shadow.

Solution: Clear stencil to zero. Draw floor with stencil of one. Draw shadow if stencil is one. If shadow's stencil test passes, set stencil to two. No double blendin

More information on

- Planar shadows: http://www.opengl.org/resources/code/samples/sig99/
- Shadow volumes: http://www.opengl.org/resources/code/samples/sig99/
- S-Buffer

http://ezekiel.vancouver.wsu.edu/~cs442/lectures/shad