CS 580 – Discussion
HW2 & 3 Projection
Week 4

Lixing Liu

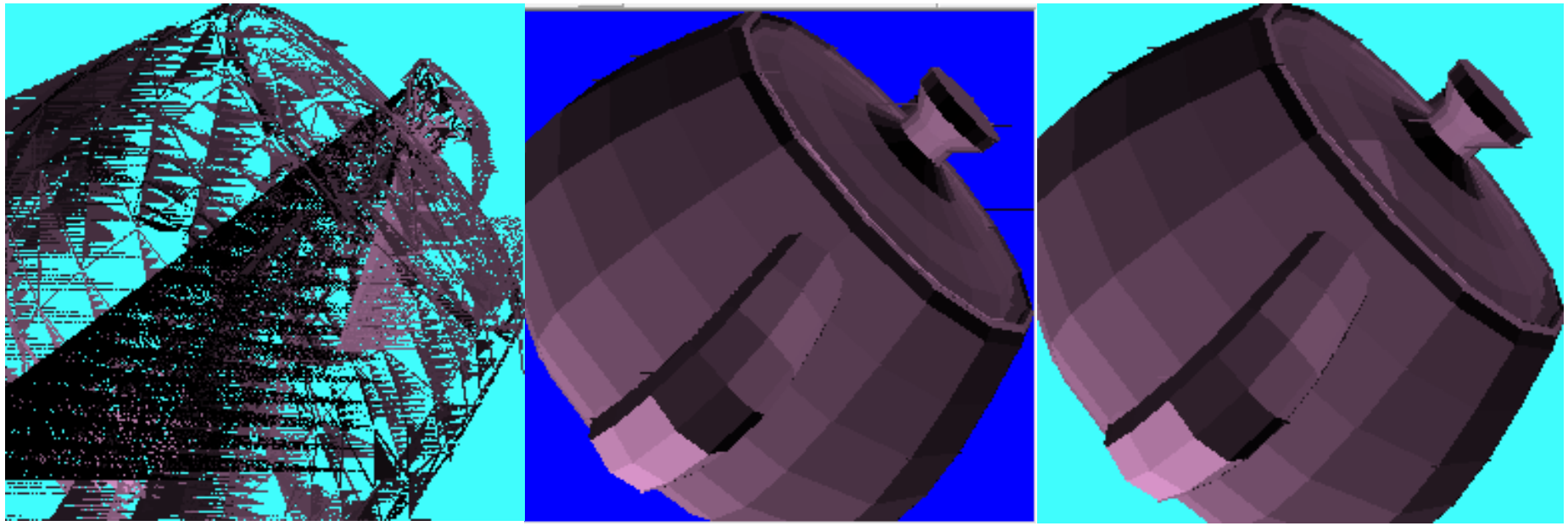
lixing.liu@usc.edu

Reminder: HW2

- Due Tue, Sep 11

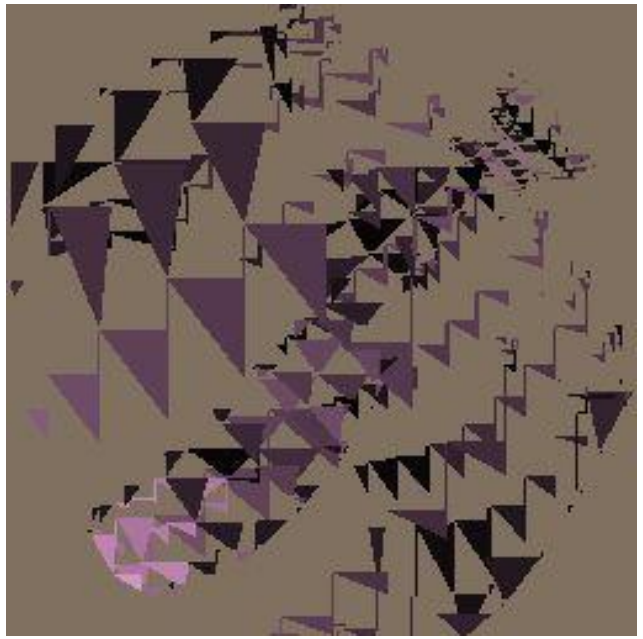
- Note: if you installed VS 2015 / 2017, you'll have .VC.db file (e.g., "CS580HW2.VC.**db**") instead of *.sdf file (e.g., "CS580HW2.sdf")
  * Please delete those large files before submitting your assignment to BB
  * The compressed package should be less than 1MB

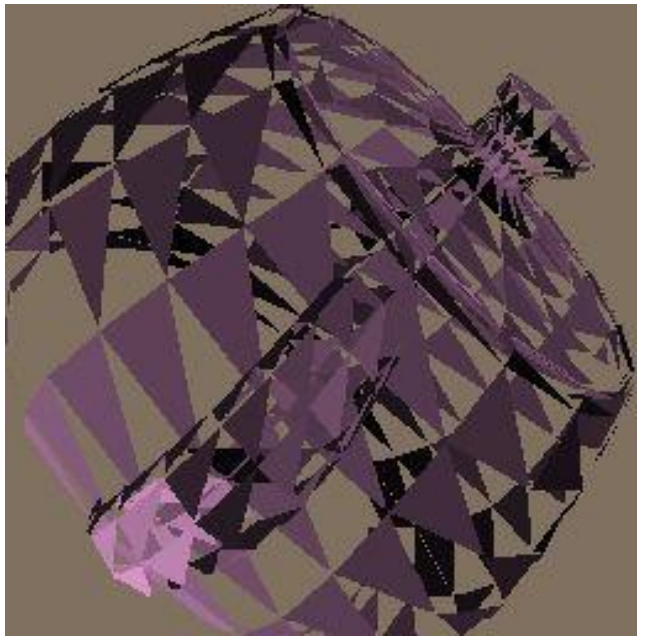
Possible issues with Scan-Line



Extra or missing horizontal/vertical lines are issues in the scan-line implementation

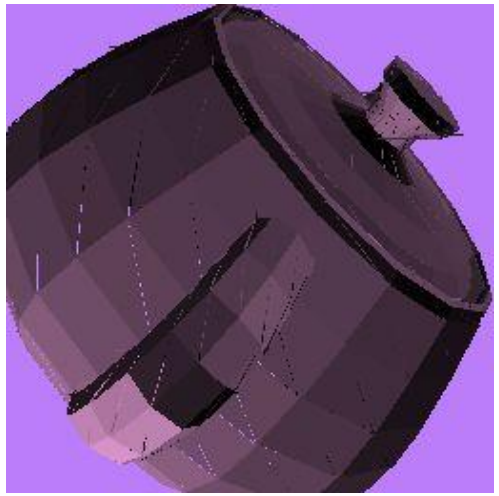
Possible issues with LEE





Missing triangles are due to not checking the LEE conditions properly
- All 3 LEE should have the same sign (**>0 OR <0**)

Possible issues with pixels falling on edges (both approaches)



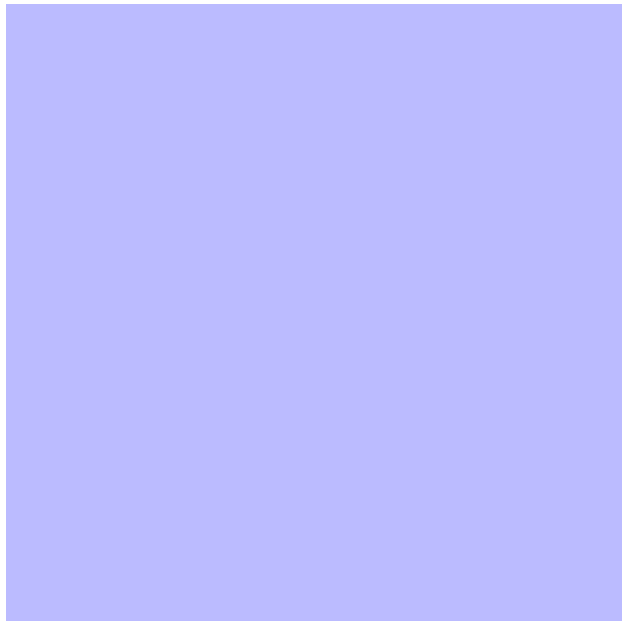
If you see triangle lines: the pixels falling on edges are not properly handled:

- Pick left or right edges and **include only one**
- If the edge is horizontal: pick top or bottom edges and **include only one**

=> C code: You can use the *ceil*/*floor* commands

Possible issues with Z-buffering

Z-buffer not checked



Wrong Z-buffer test /
Wrong Z initialization (initial Z should be **INT_MAX**)

Z-buffering

When to do Z-buffer checking?

Pixel-wise. Each time you write to pixel:

- Interpolate Z value at pixel (Zpix)
- Compare Zpix to the current Z-value in the Frame Buffer (Zfb)
- If Zpix < Zfb,
    - override existing values for pixel: color/alpha/Z
- Otherwise,
    - do nothing, skip pixel – the surface is occluded!

Note: Make sure you initialize the Z-buffer (to INT_MAX)!

Goals of HW 3

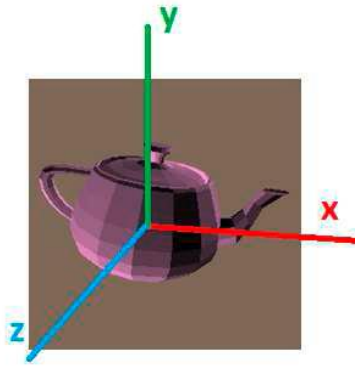
1. Change the viewpoint of the camera

Default view

Novel view

Goals of HW 3

2. Enable transformations on the teapot

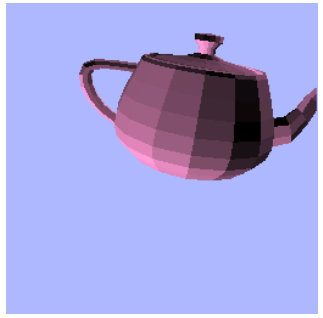


RotX = 30°

RotY = 30°

RotZ = 30°

Tr=[1 1 1]



Scale=[.5 .5 .5]

Environment setup

Open the CS580HW3.dsw file (and upgrade to .sln)

Copy and Paste your previous implementations in the proper sections in rend.cpp

Don’t modify the MFC APIs in application3.cpp or introduce any 3rd party APIs.

Goal 1: Camera view

screen **Xsp** perspective (NDC) **Xpi** image **Xiw** world **Xwm** model

Xwm object positions (per frame or per instance)
Xiw camera position and orientation (per frame)
Xpi camera FOV (focal length or zoom) (per frame)
Xsp mapping NDC image to frame-buffer (per frame)

HW2: vertex coordinates were in screen coordinate.
HW3: vertex coordinates are in model coordinate.
Note: To go from model coordinate to screen coordinate,
you need to multiply by **XspXpiXiwXwm**.

Goal 1: Camera view

The camera structure is defined in gz.h

```
typedef struct {
        GzMatrix        Xiw;      /* xform from world to image space */
        GzMatrix        Xpi;            /* perspective projection xform */
        GzCoord          position;       /* position of image plane origin */
        GzCoord          lookat;         /* position of look-at-point */
        GzCoord          worldup;        /* world up-vector (almost screen up) */
        float           FOV;            /* horizontal field of view */
} GzCamera;
```

Goal 1: Camera view

GzPutCamera:

- Set the camera parameters of render->camera
  - Position
  - Lookat
  - worldup
  - FOV

- Compute Xpi, Xiw

Goal 1: Camera view

In rend.cpp, you will need to implement:

GzPushMatrix // Pushes a matrix on the stack

GzPopMatrix // Pops a matrix from the stack

- Then you will need to push in that order:

Xsp*

Xpi*

Xiw*

Xwm

*When initializing the display: GzRender

Goal 1: Camera view – stack operations

GzPushMatrix:

- If the stack is empty
  Add the matrix
- Otherwise
  Multiply the new matrix by the top of the stack and push it into the stack
- Increment matlevel

GzPopMatrix:

- Decrement matlevel

Goal 1: Camera view – stack operations

You will need to apply the set of transformations to every vertex of every triangle in GzPutTriangle before rasterizing.

Note: Top of stack is Ximage[matlevel]

Warning: ignore triangles that are behind the view plane: skip any triangle with a negative screen-z vertex

Warning: Z-interpolation should be in screen space

Goal 2: Object transformations

Simply create the transformation matrices in rend.cpp:

- GzRotXMat // Rotation around X-axis
- GzRotYMat // Rotation around Y-axis
- GzRotZMat // Rotation around Z-axis
- GzTrxMat // Translation
- GzScaleMat // Scaling

- Warning: You need to convert the angles from degrees to radians:
  multiply by p/180

HW3 pitfalls

- Do not forget to set default camera
- Careful when implementing dot, cross products and matrix multiplications
- Convert angles from degrees to radians
- Do not forget to apply the stack to every vertex before passing on to the rasterizer
- Use homogeneous coordinates: careful when converting 4-D to 3-D vectors.

  $[x\ y\ z\ w]_T \Rightarrow [x/w,\ y/w,\ z/w]_T$
- ignore triangles that are behind the view plane: skip any triangle with a negative screen-z vertex
- Z-interpolation should be in screen space