CS560/460 Spring 2018

**Assignment 4**

(Due on April 9, 2018 by 11:59pm)

**1. Texture interpolation and geometric warping**

* **Description:** Texture warping is a typical application of the graphical object manipulation. The basic operation is a triangle-based or quadrilateral-based transformation using either affine, or bilinear, or projective transformation. This assignment is designed to deform an image in a 2D space with interaction by using mouse to click and drag in real time. The warping consists of geometric shape warping and texture warping. This assignment is to enhance the concept of geometric warping, texture (image) operation, and interpolation.

## Your implementation: In this assignment, you will implement the warping by deforming corner vertices of a rectangle. Meanwhile, the texture should be mapped onto the deformed rectangle area.

## 

## flower

## Your implementation consists of:

## Load and display the image “flower.bmp”.

## Draw a rectangle (you can define the size by yourself)

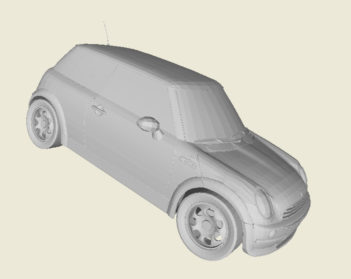
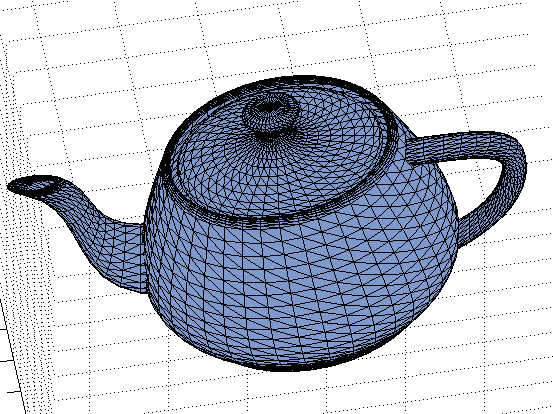
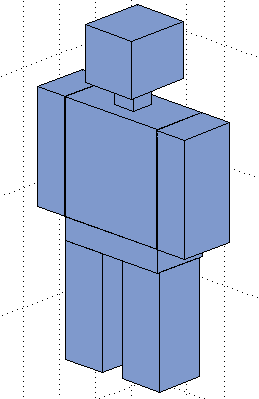
## Map the loaded image onto the rectangle area.

Implement following two functions:

## Place the mouse cursor on the *center* of the rectangle (blue dot), click to select the center vertex as an anchor vertex, then drag the anchor vertex to a certain place until the mouse button is released (red dot). During the procedure of deforming the rectangle, apply the texture mapping simultaneously. *(Or: you can do an alternative deformation: Place the mouse in the corner of the rectangle, and stretch the rectangle plus texture mapping).*

## Automatically rotate the image 360 degrees around z axis (display the image every 30 degrees)

* 1. Load a 3D model from a file (.OBJ format) (e.g., teapot, car, or robot). File name can be hardcoded. Use mouse to rotate and/or zoom the teapot model.

(4) **Extra points**: Deform the model surface by a mouse; Map a texture to an arbitrary patch of the teapot surface.

## *Extra points:* Draw a CUBE, map the texture on the CUBE (6 faces). The CUBE can be viewed from any angles controlled by mouse. The CUBE can be resized or deformed by moving its corner by the mouse.

## flower

***Note: The texture mapping is implemented using the inverse mapping approach (do not use OpenGL texture mapping function)***

**2. Hand-in**

* Your program package and the assignment report must be submitted through the blackboard on/before the due date.

## 3. Mark distribution

* Your assignment will be marked as follows

(1) 40%; (2) 30%; (3) 20%

(4) Extra 10%; (4) Extra 10%

Write-up 10%

**4. Hint**

(1) The texture coordinate computation can be realized by one of the following methods: Barycentric coordinate method, bilinear interpolation, and geometric interpolation.

(2) Any one of the four cases for texture mapping will be alright (moving points from blue to red)

(3)

texImage.bmBits stores the pixel by “**BGR**” (not “**RGB**”) with three BYTES

**(4)** texImage.bmBits is just an array of (unsigned) chars, so if you want to

have, say, your own array of pixel intensities, in your header include this:

unsigned char \*allPixels;

Inside of LoadImage, add the following:

int file\_size = sizeof(unsigned char)\*texImage.bmWidth\*texImage.bmHeight\*3;

unsigned char \*flatPixels = (unsigned char\*)texImage.bmBits;

allPixels = (unsigned char\*)malloc(file\_size);

for(int t = 0; t < file\_size; t++)

allPixels[t] = flatPixels[t];

Last, in OnDestroy, add:

free(allPixels);

to prevent memory leaks.

(5) The sample program just shows how to read image (in program), does not show the display of the image. It is your job to draw the image, using

"SetPixel(\*pDC, x, y, COLOR)" // slow

or

glRasterPos2i(x, y);

glDrawPixels(Width, Height, GL\_RGB, GL\_UNSIGNED\_BYTE, Texture); //fast

or:

glColor3ub(red,green,blue);

glVertex3i(x,y,0); //slow

**For example:**

void CCssample4View::Render\_A\_Point(int i, int j){

// “i, j” is the point in the target rectangle

CPoint c;

//barycentric interpolation or geometric interpolation

GeometricInterpolation(int i, int j);

// bi-linear interpolation //GeometricInterpolation((float)i/(float)width\_rectangle, //(float)j/(float)height\_rectangle);

// “c.x, c.y” is the corresponding pixel in the source image

int x = c.x;

int y = c.y;

int width = texImage.bmWidth;

int offset = x+width\*y;

int b = allPixels[3\*offset + 0];

int g = allPixels[3\*offset + 1];

int r = allPixels[3\*offset + 2];

glColor3ub(r, g, b);

glBegin(GL\_POINTS);

glVertex2d(i, j);

glEnd();

}