Computer Graphics

Konstantinos Chatziantoniou 8941
konstantic@ece.auth.gr

March 17, 2020
Assignment 1 - Triangle Filling

1 Code Structure

Demos:

- demoFlat.m Script without external parameters that demonstrates the triangle filling with flat coloring.
- demoGouraud.m Script without external parameters that demonstrates the Gouraud triangle filling.

Functions:

- triPaintFlat.m Given info about a triangle and an image, fills the triangle with flat color, overwriting the pixels inside the triangle.
- triPaintGouraud.m Given info about a triangle and an image, applies Gouraud filling, overwriting the pixels inside the triangle.
- colorInterp.m Calculates the color of paint on a line by interpolating between the color of the edges. Used by *triPaintGouraud.m*
- paintObject.m Given a set of triangles and info about color,depth returns an image with the triangles filled. Depending on the *painter* param, the filling is flat or Gouraud.

Both demos call the paintObject.m function, with the *painter* param "Flat" or "Gouraud" accordingly. The duck_hw1.mat has to be on the *Matlab Path*. The scripts save the image as jpg(FlatRes.jpg adn GouraudRes.jpg).

Information about the input and output of the functions can be found as comments in their files. The files triPaintFlat.m and triPaintGouraud.m contain extra helper functions.

2 Implementation explanation

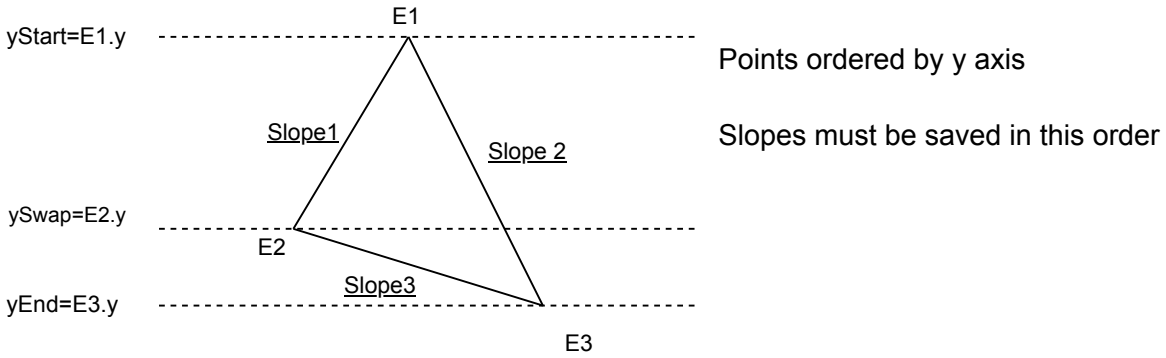
The whole program consists of 3 parts

- Finding internal points of triangle
- Color filling the points
- Applying the 2 above procedures for all the triangles

The first 2 bullet points are explained in the next subsections, whereas the third is explained in the *Assumptions* sections.

2.1 Algorithm For Internal Points

First the general case will be described, where there is not a horizontal vertex. The edges of the triangle are ordered and the slope of the vertices calculated. 2 variables are used to save the current x coordinates and 2 variables are used for the current slope. When the scanline reaches the second edge, the variables holding the slope is swapped.

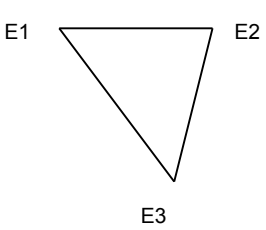


Algorithm 1: Finding points inside a triangle (General Case)

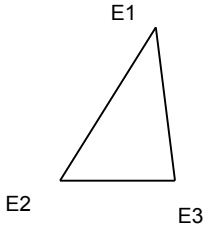
```
x1 = E1.x;
x2 = E1.x;
s1 = Slope1;
s2 = Slope2;
for scanLine = yStart:yEnd do
    if scanline reached ySwap then
        s1 = Slope3;
    end
    points = x1:x2;
    paint(points);
    x1 += s1;
    x2 += s2;
end
```

The above process does not cover 3 special cases:

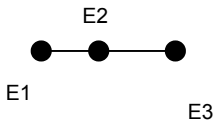
Top vertex horizontal



Bottom vertex horizontal



Just a horizontal line



The edges will always be sorted this way because they are sorted based on the x axis if they have the same y axis.

In order to be efficient, these cases will not be handled seperately. The handling will be embedded to the general algorithm.

- Top vertex horizontal: add $x1 = E2.x$ to the first check. This check also triggers for horizontal top vertex $scanline = E1.y = E2.y$ and $x1$ should be updated regardles of the slope. This doesn't affect the general case, since the x1 would already be equal to E2.x when the check condition is met.
- Bottom vertex horizontal: To the first check, add one more check for the third edge y coordinate. $x2$ should be updated regardles of the slope to x2 = E3.x.
- Only a horizontal line. This case is partially covered by the previous handling. We have to restore x1, altered by the first check to get the whole line.

Algorithm 2: Finding points inside a triangle (All cases covered)

```
x1 = E1.x;
x2 = E1.x;
s1 = Slope1;
s2 = Slope2;
for scanLine = yStart:yEnd do
    if scanline reached ySwap then
        s1 = Slope3;
        x1=E2.x;
        if scanLine == E3.y then
            x2 = E3.x;
            if scanLine == E1.y then
                x1 = E1.x;
            end
        end
    end
    points = x1:x2;
    paint(points);
    x1 += s1;
    x2 += s2;
end
```

2.2 Algorithm for Gouraud Filling

In order to linearly interpolate the color for each scan line, we will use the same process for the calculation of the x coordinate of the active point on the active vertex.

Each edge is assigned a color C_i.The color of the points on a vertex V_{ij} will be linearly interpolated from the C_i and C_j. The color of the active points of the scan line, wiil be linearly interpolated from the color of the active points on the active vertices.

Algorithm 3: Gauraud paint points inside a triangle (All cases covered)

```
x1 = E1.x;
x2 = E1.x;
s1 = Slope1;
s2 = Slope2;
c1 = C1;
c2 = C1;
g1 = Grad1;
g2 = Grad2;
for scanLine = yStart:yEnd do
    if scanline reached ySwap then
        s1 = Slope3;
        x1=E2.x;
        c1 = C2;
        g1 = Grad2;
        if scanLine == E3.y then
            x2 = E3.x;
            c2 = C3;
            if scanLine == E1.y then
                x1 = E1.x;
                c1 = C1;
            end
        end
    end
    points = x1:x2;
    colorInterp(c1,c2,x1,x2,points);
    x1 += s1;
    x2 += s2;
    c1 += g1;
    c2 += g2;
end
```

colorInterp function is trivial linear interpolation function. It is in *colorInterp.m* file.

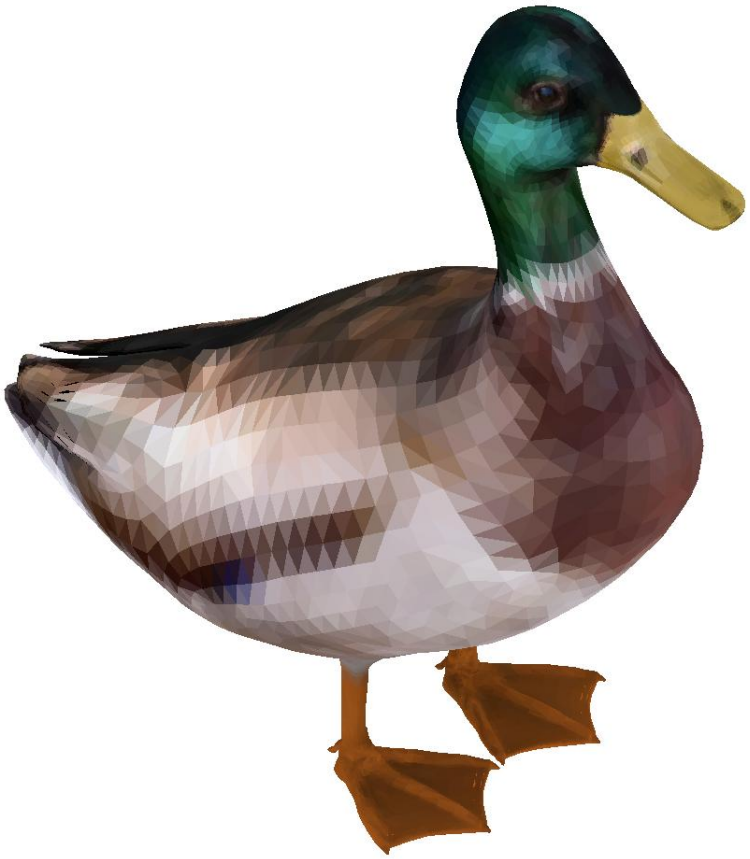
2.3 Assumptions

There are no extra assumptions for painting a single triangle. All cases are covered.

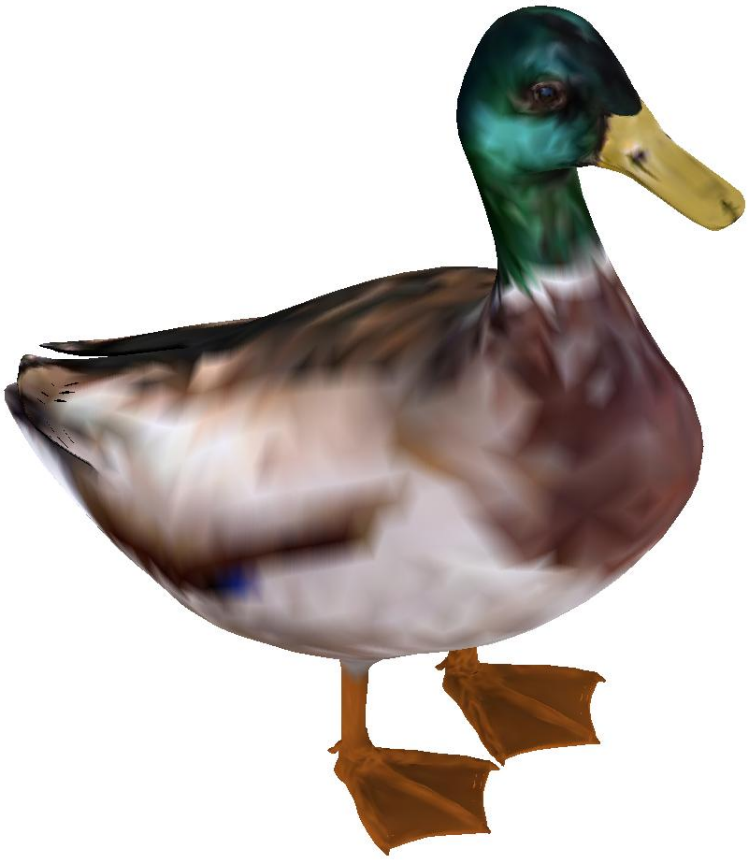
To paint the whole image, we start by filling the triangles with the biggest depth (far away) and then overwriting them by filling the nearest triangles. The depth of each triangle is calculated as the mean of the depth of the edges. We **assume** that there are no triangles intersecting each other. If they were, the above process wouldn't work correctly.

3 Results

Below are shown the images that the two demos produce. The flat version took approximately 0.8 seconds whereas the Gouraud version took 1.1 seconds on an i7 7500u (2.7GHz).



Flat filled duck



Gouraud filled duck