Assignment 3: Triangles and Z-buffering
CS180 Fall 2023

Professor: Lingqi Yan

University of California, Santa Barbara

Assigned on Oct 20th, 2023 (Friday)

Due at 23:59 on Oct 26th, 2023 (Thursday)

Notes:

- Be sure to read the university's Academic Integrity policy.
- Any updates or correction will be posted on Canvas, so check there occasionally.
- You must do your own work independently.

---

1 Overview

In the last assignment, we drew a wire-frame triangle on the screen, which doesn't look so interesting. This time we will move one step forward – draw solid triangles on the screen, in other words, rasterize a triangle. Last time, after the view-port transformation, we called the `rasterize_wireframe(const Triangle& t)`, this time you need to implement and call the function `rasterize_triangle(const Triangle& t)`.
The general workflow inside this function is:

1. Convert NDC coordinates to screen space coordinates.
2. Create the 2D bounding box of the triangle.
3. Iterate through all the pixels (using their **integer** indices) that are inside this bounding box. Then, use the screen-space coordinate of the center of the pixel to check if the center point is inside the triangle.
4. If it is inside, then compare the **interpolated** depth value at its position to the corresponding value from the depth buffer.
5. If the current point is closer to the camera, set the pixel color and update the depth buffer.

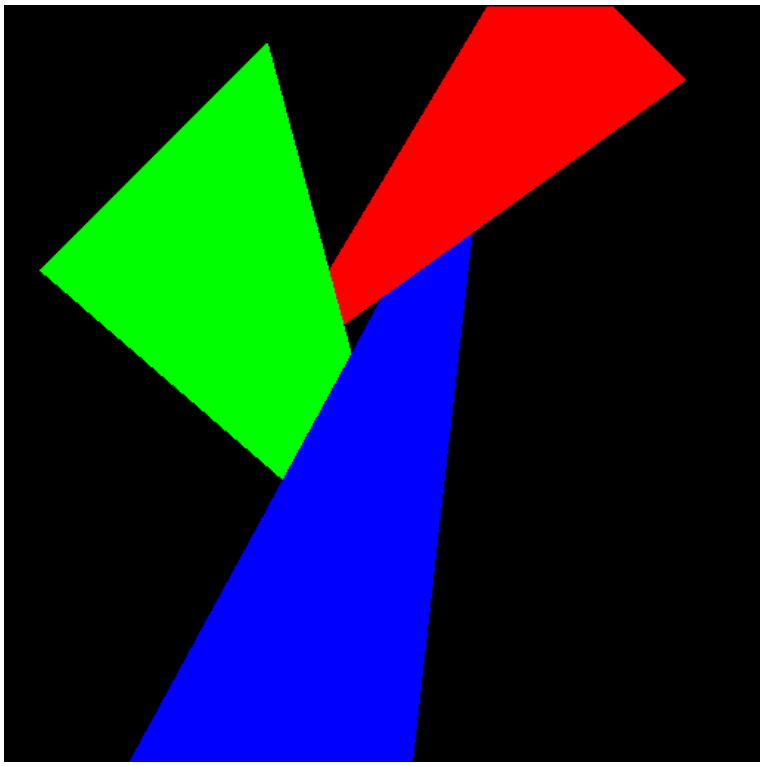
Functions that you need to edit:

- **rasterize_triangle(const Triangle& t):** Perform the triangle rasterization algorithm. It is located in the file `rasterizer.cpp`.
- **static bool insideTriangle(_p, _v):** Test if a point is inside a triangle. Return true if the point is inside the triangle, otherwise return false. This function needs to be called by `rasterize_triangle()` in `rasterizer.cpp`.
- **calculateSSCoords(const Vector3f* v_ndc):** Calculate the screen space coordinates of three vertices. The inputs are the NDC coordinates of three vertices.
- **get_model_matrix(float rotation_angle)**: For this assignment you don't need to handle any rotation, just return an identity matrix. It is located in the fille `main.cpp`.

- **get_projection_matrix(float eye_fov, float aspect_ratio, float zNear, float zFar):** This is also left empty in the file `main.cpp`. Copy-paste your implementation of this function from second assignment. Please make sure that the copied implementation is correct.

Since we only know the depth value at the three vertices of the triangle. For the pixels inside the triangle, we need to do the **interpolation** to get its depth value. We have handled this part for you, since it is the topic that has not been covered in the lecture yet. The interpolated depth value is saved in the variable **z_interpolated**. You can find this part of the code in the comments in the `rasterize_triangle(const Triangle& t)`.

Pay attention to how we initialize the depth buffer and the sign of your z values. In this assignment, you don't need to handle the rotation transformation, just return an identity matrix for the model transformation. If you do it correctly, you will see the output image like this:



You will get an incorrect result if the depth testing part is not implemented correctly.

2 Getting started

You will notice that the **get_projection_matrix()** function in **main.cpp** is left empty. Copy-paste your implementation from the **second assignment** to fill in this function.

2.1 Run your code

Open the terminal. Under your code folder, run following script

```
bash ./run.sh
```

You can ignore the warning information

```
rm: cannot remove 'res.png': No such file or directory
```

Sometimes there are some issues of directly running scripts. If it doesn't work, you can copy the command line inside the **run.sh** and run one-by-one.

3 Grading

1. (5 points) Submission is in the correct format, includes all necessary files. Code can compile and run.
2. (5 points) Implement the NDC coodinates to screen space coordinates correctly.
3. (10 points) Implement the inside triangle test correctly.
4. (10 points) Implement the z-buffer algorithm correctly. The triangles are drawn on screen in correct order.
5. (10 points) Implement the remaining part of rasterization correctly.
6. (Bonus 5 points) Anti-aliasing by super-sampling: You may notice that the result image is quite jaggy when we zoom in. We can solve this by super-sampling. Sample each pixel by 2×2 samples and compare the result before and after. You need to create a larger frame buffer to do this. One way to think about is that instead of considering only the center of the pixel, each pixel will now have 4 samples (if 2×2 is used). Color of the pixel will be the average of all the samples' colors. For this part, you can add codes at any place

but do not add/delete files. If it is correct, the generated **res.png** should be anti-aliased.

3.1 Submission

Please submit **ONLY ONE** zip file on Canvas containing your project and a brief report. Don't include any intermediate results such as compiled files, images, etc.. And please also don't include **external** folder in your zipped file. The zip file should be named as "NAME-PERM_Number-HW3.zip" and replace any space with underscore, e.g. "Songyin_Wu-000000-HW3.zip". After unzipping the zipped file, there should be only the files that originally contained (except **external** folder) and a report file without any additional folders and files. Uploading homework that doesn't meet the requirements will result in **1-5 points** deduction.

Do not add any additional input/output (usually there is no need to write any input/output), and try to avoid adding any code outside the given areas which look like

```
// ********************** TODO: xxx **********************


// *****************************************************
```

Sometimes they may lead to wrong results when grading. It is totally fine to add something when debugging but remember to remove them before uploading to Canvas.

The report only needs to include key parts of how to get the final answers for each question. No need to include too many details. It is usually less than one page.

4 Bonus Points

Starting from this homework, there will be some questions that are **optional**, which means the points you get here (we call them bonus points) won't be counted into your final score or on Canvas. The only usage of bonus points is that if you get A in this class (without the bonus score), it may help you get A+. We will leave comments if your implementation of bonus questions is correct on Canvas.