Università
Ca'Foscari
Venezia

ADVANCED ALGORITHMS AND PROGRAMMING METHODS

CM 470-2

ASSIGNMENT 3

Student:

Giosuè Zannini 873810

Academic year 2021/2022

Problem

In this assignment I had to continue the implementation of the second assignment. Parallelize the rendering pipeline. Decide on what level to introduce concurrency (fragment, scanline, triangle, object) in order to maximize throughput (and minimize contention). And explain reasons and synchronization requirements. Have your library create a user-specified number of worker-threads and distribute the load among them.

Resolution

To solve this task I have done a lot decisions:

- I have included the screen coordinates in the vertex object;
- the method that I used to check if a point is into the triangle is the edge function, a simple method to know if a point (x,y) is below or above of the segment created by the two input parameter vertices checking that the result is ≥ 0 and then recall it also in the other edges;
- the method that I used to calculate the depth for each point is using the plan equation, in fact I can think a triangle as a plan passing between the three vertices;
- after calculating vertices triangles in screen coordinates, I order them in clockwise order;
- I add the methods to manage the textures and other properties;
- for the problem "Primitives outside the range are dropped", in our project is very simple its management, in fact I used two nested for loop to elaborate only the pixel that are into the screen dimensions;
- all data types in my project are double (even coordinates) to maintain a high level of accuracy;
- I used a specific file for all the constants and I decided on this strategy to make the project more dynamic, parametric and simpler variable;
- I used the character '.' for the empty cells, I can modify it using the PRINTED_CHARACTER_INVALID_CELL constant in the ValueProperties.h file;

- I have decided to implement concurrency at the level of the triangles, because in this way I have a bit of concurrency and I can split the jobs in easy way. So each thread computes all phases till the print on the screen for each triangle that they have.

I used many files in this project:

- Main.cpp: this file is used to create the complete logic execution plan, in fact in the first part there are all variable definition and initialization and in the last part there is the definition of the Parallelization object and the method to do the rendering.
- OperationRasterizer.h: this file is used to define the object class that contains all required functions to solve this assignment and in fact the class used inside represents the rasterizer object.
- Triangle.h: this file is used to define the object class that contains the methods to create the triangle object with an array of three vertex class objects representing the three real vertices.
- Vertex.h: this file is used to define the object class that contains the methods to represent the vertex object and this object has many fields as input coordinates, normalized coordinates, u and v parameter and the last are screen coordinates that I added because perhaps they are useful in the next assignment otherwise I will remove them.
- ValueProperties.h: this file is used to maintain organized all costants and their value in a single file to make the project more dynamic, parametric and simpler variable, moreover here there is N_THREAD that identify how many threads the program should use to do the rendering.
- ReadFile.h: this file is used to create all useful functions to read and verify the correctness of the input file that rappresents the object, composed of only many triangles, that I want to draw in the screen.

- Thread.h: this file is used to define the object class that contains the method to create threads, split the job and to make a rendering.

When the program is executed as first thing all necessary variables and objects are initialized as screen matrix, zbuffer, projection matrix, projection array and so on. The last object that I added is mutex_matrix and it maintains the concurrency on z_buffer. After I created the Operation class object with Target_t=char, also setting their fields passing the references of screen matrix, zbuffer, projection matrix. Then using the readfile methods I create a dynamic vector of triangles using data inside the .obj input file and in each for loop I use the size method of the vector class to get the correct number of triangles that are describe in the obj input file. If the file isn't in the correct format or the number of parameters are wrong the program print in output the error description and it will stop the execution. After this phase I create the Parallelization object that takes : OperationRasterizer object, vector of triangles and the number of thread. During the initialization this object decides if the number of thread are defined by user is correct or not (for example if number of thread is greater than number of triangles the system set the number of thread equal to number of triangles, otherwise will be a lot of threads without a job, and so on) and distributes the job (triangles) in fairness way among all threads. As last step this object calls the method rendering that creates all threads and each of them computes the input coordinates and normal coordinates using the view matrix that the prof sent by email. Then for each triangle calculate the normalized vertex coordinates, and the relative screen vertex coordinates and for each point check it using the edge function if it is into the triangle and in case calculate the depth using the plan equation. Finally compare it with the zbuffer and in case it is smaller update zbuffer so to make visible only the near object and using the shader to calculate the first decimal of the depth to save it in the screen matrix, at the end the main thread print the screen matrix. I used the mutex into z_buffer because only in this phase the threads can modify the same cell. In the other phases each thread

only reads the variables. Moreover I used another mutex called m_print into class Parallelization, because the print isn't a safe function for the concurrency, then to avoid problem during the print I used this mutex. As last thing I added the time to make the job for each thread and the complete time.

NB: In file "Example of output.txt" there is an example of output