CPT205 Computer Graphics (2020-21)

Assessment 3 – Discussion Questions and 3D Modelling Project

Yuxuan.Ren

1823678

Information and Computing Science

2021.1.7

PART I. Discussion Questions

Question 1. Briefly describe the graphics pipeline, identify pipeline bottlenecks, and discuss possible ways to tackle the bottlenecks for enhancing the performance from both software (e.g. algorithms) and hardware perspectives.

Q1 Answer:

Graphic pipeline:

Graphics pipeline is a series of operations to process vertices (which define the objects) to obtain pixels, which then form an image on screen. It contains three main steps: Geometry processing, Rasterization and Fragment processing. After modelling, we get vertices. The aim of geometry processing is to carry out coordinate transformations, to compute a colour for each vertex, to assemble sets of vertices into primitives (Primitive assembly) and to decide which objects can be seen on screen (Clipping). In the second step (rasterization), a set of fragments for each primitive will be given, which can be thought as a potential pixel that carries information, including colour and location, depth. Finally, Fragment processing takes in the fragments generated by the rasterizer and updates the pixels in the frame buffer (hidden-surface removal, antialiasing and texture mapping are done at this stage).

Pipeline bottlenecks:

Different types of bottlenecks occur during different stages.

1) The transfer-limited bottleneck. Data transferring from application to GPU cannot take place in time because of the amount of data and inadequate format [1].
2) The transform-limited bottleneck. It is caused by too many vertices and too much computation per vertex [1].
3) The Texture-limited bottleneck. The main issues for this bottleneck are low texture memory, intensive texture filtering and poor cache utilization [2].
4) The framebuffer-limited bottleneck. It is caused by lots of read and write operations on the framebuffer [1].

Possible ways (corresponding to the bottlenecks' order):

1) Minimize data sent per vertex [1]. For example, it is better to use GLfloat than GLdouble if possible. Also, for hardware, it is better to increase the bus bandwidth.
2) Decrease the costs over a large number of primitives by maximizing calls and minimizing each primitive [1]. In addition, when setting the light mode, it is better to use directional lights which cause fewer computation [1]. Also, using level of detail to decrease computation for farther objects [2]. For hardware, improving the GPU vertex processing power can help make some progress [1].
3) First, the appropriate resolutions should be chosen to relieve memory pressure. Second, the right filtering modes should be chosen so that there will be fewer samples [1].
4) After a screen clear operation, do CPU-related work instead of filling the graphics pipeline with more graphics command immediately [1].

Question 2. Discuss applications of computer graphics incorporating artificial intelligence. This should cover techniques, key issues and possible solutions with directions for future development.

Q2 Answer:

One application is that artificial intelligence helps to make computer graphics more well-performed. Companies evaluates their pipelines and replace the components with machine learning components to achieve better performance. For example, using machine learning algorithms to scale, denoise and replace objects [3]. There is another interesting application called "Image and

Video Stylization" which uses machine learning to deal with an image or a video and change their texture or something else to convert them into a particular style (a style from an artist or other sources) [3].
Another application is that artificial intelligence and computer graphics can be used together to generate rich dynamic virtual environments with complex human–object interactions which is needed by VR, AR and robotic [4]. A common way to do this is to learn poses, conditions on specific actions from static interaction snapshots [4][5] and then use computer graphics to create visual contents [5]. But the key issue is that static interaction snapshots could have bottlenecks when representing and creating dynamic scene which changes with time [4]. The interactions with objects cause sequential transformations which is very difficult to formalize [4]. To solve this problem, there is a way to train a model based on Recurrent Neural Network (RNN) using the videos of everyday activities [4]. This method will have the ability to predict the interaction and plan motions for a real robotic agent [4]. In one experiment [4], when the experimenter picked up the bottle and made a gesture to pour water, the cup (a robotic agent) which is far away from the bottle immediately rushed to that position. In my opinion, if we apply this application to robots, especially the domestic robots, we are closer to the idea of smart home.

References

[1] 2003. [Online]. Available: https://www.researchgate.net/publication/255651009_OpenGL-Performance_and_Bottlenecks. [Accessed: 05- Jan- 2021].
[2] Nvidia.com, 2003. [Online]. Available: https://www.nvidia.com/docs/IO/8230/GDC2003_OGL_Performance.pdf. [Accessed: 05- Jan- 2021].
[3] A. Agrawal, "Application of Machine Learning to Computer Graphics", *IEEE Computer Graphics and Applications*, vol. 38, no. 4, pp. 93-96, 2018. Available: 10.1109/mcg.2018.042731662.
[4] H. Wang et al., "Learning a Generative Model for Multi-Step Human-Object Interactions from Videos", *Computer Graphics Forum*, vol. 38, no. 2, pp. 367-378, 2019. Available: 10.1111/cgf.13644.
[5] M. Savva, A. Chang, P. Hanrahan, M. Fisher and M. Nießner, "PiGraphs", *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 1-12, 2016. Available: 10.1145/2897824.2925867.

PART II. 3D Modelling Project

Written Report [You may have headings/sections and sub-headings/subsections where you would like to.]

Idea and introduction:

The idea of this project is to build a dream world. Everything can happen in a dream. To introduce my work, I have to mention a great idea (in my opinion) I had when I was young. It is to study in dream. Or make it more specific, to review what I learned during daytime in dream. As the proverb goes, what you dream at night is reflective of what you think during the day (although Floyd has some more outstanding opinions).

The little boy learned the knowledge of solar system yesterday. He loves the universe and dreams of one day can travel around the solar system. This fancy dream comes true in a real dream. He can observe the solar system and every planet from every angle. This beautiful world doesn't disappear until the alarm rings (6 0'clock, he needs to go to school). But since it is too cold outside, he has two choices now: 1) go to school like a good student. 2) continue his dream.

Keyboard:

(One colour for a function or several related functions)

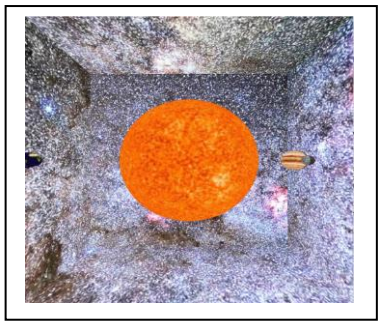
Both upper case and lower case are accepted

Q, E	Zoom in the first scene (dream world)
A, D, S, W	Change the camera position in the first scene
Z	Stop the planets
R	Focus on one specific planet (Planet mode)
F	Stop focusing on one specific planet (cancel planet mode)
L	Use wire instead of solid models and polygons
T	Clock on, move to the second scene (real world)
Y, N	Choose to go to school or go to bed (can only be clicked when the prompt text appears)
O	Open the door on the second scene

U, H, J, K	Change the camera position in the second scene
ESC	Quit

Brief instruction section:

1. You will firstly see this scene:

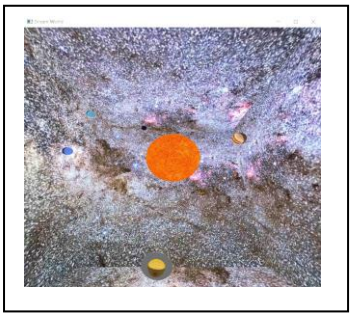


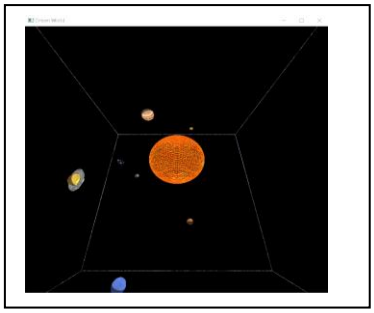
This is the solar system in a bright space (the texture is beautiful but a bit bright)

Now you can use 'W', 'S', 'A', 'D' to change the camera position (pan) and 'Q', 'E' to make it closer and farther (zoom). But no matter how the position changes, the camera is always looking at the sun.

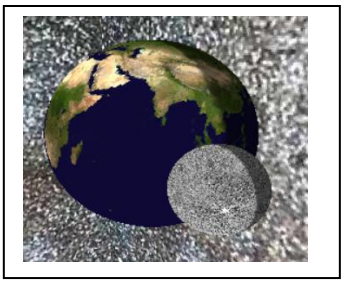
At this time, you have three choices.

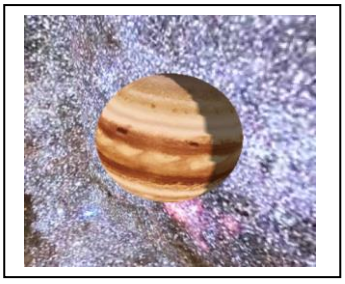
2. The first choice is to press 'L'. In the left picture, it is too beautiful and dazzling to let you ignore the planets. Press 'L' can let you see the planets more clearly. I call it 'Focus mode'.

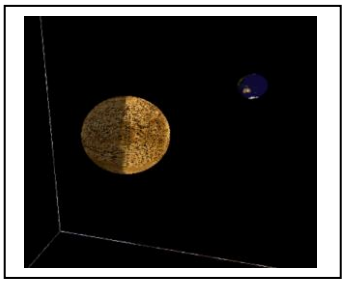




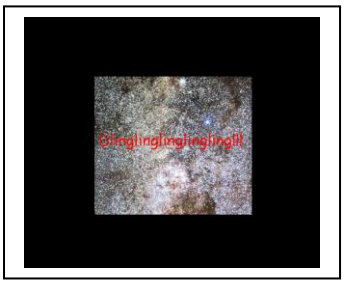
3. The second choice is to press 'Z'. It will stop the planets. Then you can press 'R' to change the camera position to see every planet. I call this 'Planet mode'. You can also use 'W', 'S', 'A', 'D', 'Q', 'E' under this mode (the same function as before). But remember to press 'F' to cancel this mode (to initialize the camera position). It is also a good choice to combine 'Planet mode' with 'Focus mode'. If you want to let the planets move again, you should first press 'F' to cancel the 'Planet mode', then press 'Z'.

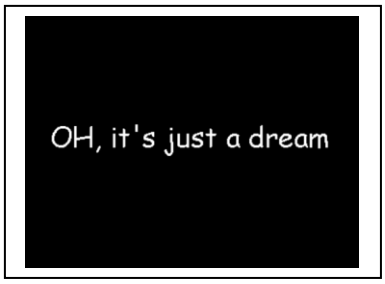


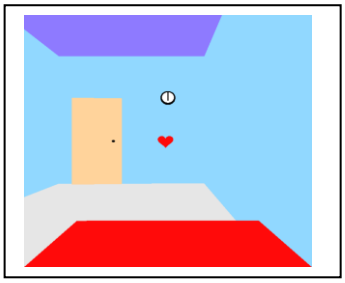
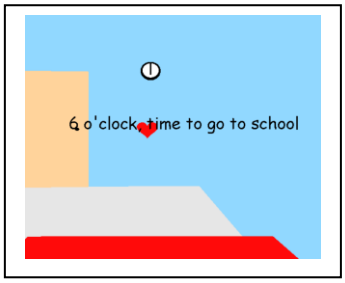


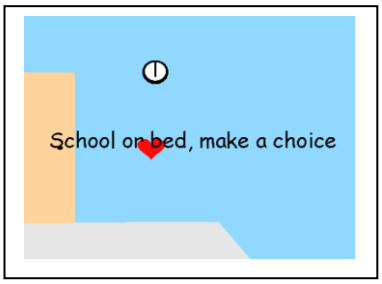


4. The third choice is to press 'T'. According to the story of this program, it lets the alarm clock ring. With "Dlinglinglinglingling!!!" on screen, the space fades away. After some seconds, the real world appears. There will be some hints in front of you.



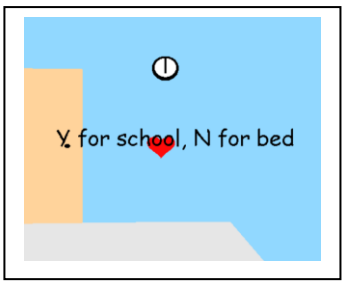



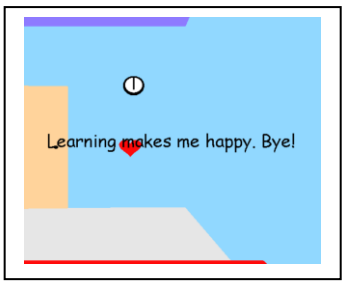


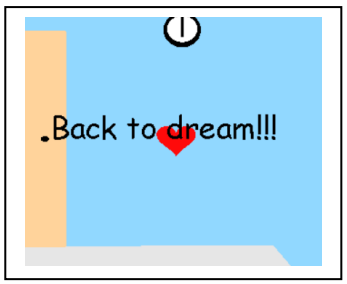
5. Then you will meet with another choice. From the hints, you will know that you need to decide whether to go to school or not.

   Press 'Y' means you will go to school, then "Learning makes me happy. Bye!" will be shown on the screen and after a few seconds, the program exits.

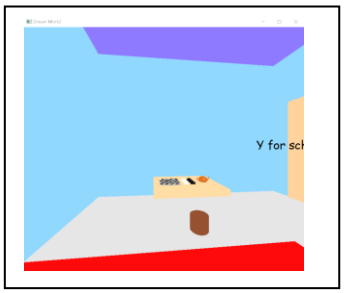
   Press 'N' means you won't go to school, then "Back to dream!!!" will be shown on the screen and after a few seconds, you will go back to the first scene (dream world). Loop again and again if you press 'N' every time.

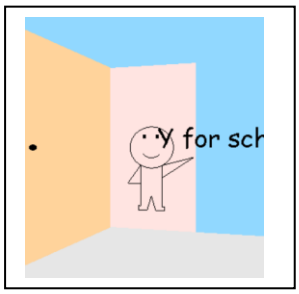






6. But you can also ignore the hints, just look around by pressing 'U', 'J', 'H', 'K'. You can also press 'O' to open the door.





The components and graphic techniques:

1. The first scene (Dream world):

   This dream world is the demonstration of the solar system. At first, I just think I should make an ideal scene which I cannot see in daily life since it is dream. I get the idea of solar system from lab 9.

   The boundary is a sky box, which can always be seen in some games. It is like making a paper dice with six different pictures. Everything is in this 'dice'. The photo was taken at a photo exhibition by myself last month. It is the real, gorgeous sights outside our Earth. Then I use it to get six texture materials. The truth is that the overall landscape is beautiful excepts for some edges. It is because the two sides of some edges are from different places.

   I use different textures for sun, every planets and moon. The first scene can be seen as a high-density use of textures. The methods for read images and bind the texture are from lab 13. I got all the textures I want on the internet. A problem I found before is that I should always remember to disable the texture before I need to set a colour using glColor3f.

   In the beginning, I really want to draw the solar system in proportion. I even searched for the real data. Since the sun is especially huge, I properly change their sizes.

   The solar system can be seen as an excellent example for hierarchical modelling. 1) The planets move around the sun at different speed. 2) The planets spin in their own speed. 3) The satellites spin in their own speed (In my program, only moon) 4) The satellites move around their planets. Thus, some models can share same transformations, which can make the code simpler and make the relationship between each model more clearly.

   The lighting of the sun is a little tricky. I use two spot lights at the corner of the sun. Each of them has a cut-off of 90 degree, which means the sun light can reach every corner in this world. I just need to ensure that I disable the lighting when I drawing other parts.

   The placement of camera spends me a lot of time. At first, I wanted to let it go anywhere. But soon, I found that it was really puzzling and was hard to get to a target position. Therefore, I just fix the look-at point of gluLookAt. The camera is put on a circle. The three parameters for camera movement are the radius of the circle, the angle between X-axis and Z-axis and the angle between Y-axis and X-Z plane. The camera position can be expressed as (abs(fltRadius * cos(fltY_XZAngle / 180 * PI)) * sin(fltXZAngle / 180 * PI), fltRadius * sin(fltY_XZAngle / 180 * PI), abs(fltRadius * cos(fltY_XZAngle / 180 * PI)) * cos(fltXZAngle / 180 * PI)). 'Q' and 'E' changes the radius, 'W' and 'S' changes the angle between Y-axis and X-Z plane, 'A' and 'D' changes the angle between X-axis and Z-axis. For the angle between Y-axis and X-Z plane, it is

only able between -90 and 90. When the 'Planet mode' is on, the look-at point changes to the position of the planet (can be calculated using the obit radius and the orbital angle). At this time, the camera position must add an offset since the look-at point is no longer the original point (the sun).

The 'Focus mode' just changes the solid objects to wire objects. At first, I use it to adjust the asteroid belt around the Saturn. Then an idea came to my mind: if I convert everything into wire objects, since my background colour is black, the planets and the sun will be the only thing left in this scene. Without sky box, we can be more focus on the planets.

2. The second scene (Real world):

   If the first scene is seen as an application of texture, lighting and materials, the second scene is just a combination of simple primitives. These two scenes provide a sharp contrast (a way to emphasize the importance of texture, lighting and material). One reason for making this scene so simple is that I want to highlight the beauty of dreams. Another reason is that I think the first scene is the main part of this program, the second scene is only used to add story (give two different endings).

   The table and the contents on it are also an example of hierarchical modelling. The chessboard texture is from lab 11. The door can be opened. I add a boy, which is from my assessment 1, to make it funnier.

   The red cube under the camera is a bed. I want to make it look like a bed in Minecraft, which is simple enough.

   The clock is a little tricky. It is a texture I made before for the assessment 2.2. In that texture, just some of the elements in the matrix are black. When I choose a circle area in the middle of the texture coordinate, it really looks like 6 o'clock.

   In my design, the second scene is just 4000-unit away from the first scene. Since the projection function in my project is gluPerspective (100, 1.0, 0.1, 2000.0) (The far parameter is only 2000), you cannot see the second scene when camera is in the first scene. I use one camera for each scene. When certain conditions are met, use the appropriate camera.

3. The words on the screen:

   I put all the words in front of the camera. In the second scene, they are also on a circle, so that they can move a little (I set the distance between the camera and the words to a value not very big in order to make sure that they won't always in front of the camera) when the camera rotates.

timeCounter is a variable I used to count the time. It increases per frame, so this variable can also be seen as frame counter. No matter what it is, it creates time difference for me so that I can display different words at different times.

4. Keyboard:

   Finally, I should talk about the keyboard. This program is much more complexed than the ones before, so for each press, I need to judge some other conditions. clockOn is a parameter which means that the clock is ringing. It can also be seen as a trigger to switch to the second scene. So clockOn == false means the first scene is displayed now and clockOn == true means the second scene should be displayed. It should be a condition together with the judgement of the key in order not to make some changes in the second scene when you are in the first scene, vice versa.

   There is another thing that needs to be concerned. If you press 'N' in the second scene, the screen will go back to the first scene, and if nothing changes, at this moment, parameters and object positions are not as same as when the program has just been opened (for my design, you will see nothing). The best choice is to initialize all the parameters again.

Conclusion:

In conclusion, this program simulates a common situation in our study life. We have sweet and beautiful dreams sometimes. But nearly everyone meets the problem about getting up early for class. At this moment, some people complain about how little sleep they get (especially in winter), but many of them have no choice but to go to school. Using this program, I commemorate this unforgettable experience. I hope I will always remember this feeling even when I'm no longer in school.

There are still three things I can do to improve this program. 1) Make the school scene, so that when I press 'Y' to go to school, the story will be more amusing instead of just showing a sentence. 2) I mainly focus on the camera moving, animation and story design. Some models can be more complexed. 3) The solar system is not the only thing in the space (for example, comets), with some further learning, maybe I can make the first scene better. Many people have a few knowledges about space, they just know that astronauts are great, but fewer knowledge about real outer space. This kind of programs can provide a way to broaden their horizon.