Durham University

COMP3751 Interactive Media, Gaming, and VR-AR Technologies Summative Assignment, Submodule: Virtual & Augmented Reality

Important, please read first!

- The assignment should be submitted via Blackboard; the deadline is May 8^{th}, 2pm. All deadlines can be found in SharePoint.
- Your software will be tested and should work with Python 3.11.1. You can use additional Python libraries to achieve any auxiliary functionality (e.g., NumPy), but the requested algorithms/methods asked in problems 1 - 5 should be implemented by you, as evidenced in the source code that you will submit, not via existing libraries that implement that functionality.
- Please submit a compressed archive (.zip) with: (a) all your source code, original/modified engine source code files and 3D model. I should be able to run your software just by running `python3 render.py` on terminal and see the two requested sequences on screen one after the other. Include a readme file with instructions on how to run your program and what external resources you require if this is not by calling `python3 render.py`. (b) A .pdf report no longer than 6 pages including images (max $\sim$2500 words). At the top of the first page of the document identify yourself using your CIS username. (c) A short video demonstrating the full sequence succession (compressed to $<$100MB, you can use any online video compression platform to compress it). The video should show the movement in real time as indicated by the tracking dataset, i.e., each sequence should last $\sim$ 27 seconds. When running your software though, it is ok if the rendering takes more time - software renderers are slow. You can speed-up the video to make it last 27 seconds.
- The marks available for correct implementations/answers to each question are indicated. Partial credit will be given for good attempts. The level of achievement (good/very good/excellent/etc.) for each marking criterion is determined based on the marking and classification conventions published in the university core regulations. The Virtual & Augmented Reality submodule is only assessed by this coursework (50% of the module mark). A FAQ section in Blackboard will be updated with questions as they arise. Please note that the engine suffers from some lighting-related rendering artefacts that you can ignore.

As a VR engineer you have been tasked to extend a prototype 3D graphics engine capable of rendering simple images. You are provided with a basic 3D engine [1] (**improved version that you should use** is available in Blackboard), capable of rendering simple 3D objects using orthographic projection only (i.e., no perspective projection, depths are projected on the screen without taking their depth -distance from camera- into account instead). The engine can perform simple shading too, based on vertex colour interpolation. The engine outputs a single framebuffer on the disk.
Your assignment is to extend the 3D engine, to handle perspective projection & object transformations, tracking, physics, and post-processing. Your task list is below. You should demonstrate your development in a demo scene by **rendering a VR headset (3D model is provided) rotating mid-air in the middle of the viewport (based on provided tracking data of a real headset rotating) while other headsets slide on the floor (3D model is provided) at the bottom of the scene, decelerate due to friction and collide with each other.**

Before proceeding, please read LaValle's relevant chapters in the free book available in Blackboard, attend all relevant lectures and read [2] carefully (Please note the typo in the order of operations in eq. 8 & 10).

PROBLEM 1, RENDERING - 20 MARKS:

The provided engine currently only produces static renders. Add the following features to the rendering engine by updating its source code as required:

1. Enable real time output of the frame buffer on the screen, instead of output to the disk. You could use PIL / Matplotlib /OpenCV / etc. to show the buffer on the screen. (5 mark)
2. Implement perspective projection instead of orthographic projection. You will need to extend some elements of the engine to handle homogeneous coordinates. (10 marks)
3. Implement the basic transformation matrices, in particular add the ability to translate, rotate, and scale objects. (5 marks)

PROBLEM 2, TRACKING: HANDLING POSE DATA - 10 MARKS:

In the Blackboard coursework folder, you can find a sample dataset (6959 records) acquired from a VR headset's IMU. The headset was sequentially rotated from 0 degrees, to +90 degrees and then to -90 degrees around the X, Y and Z axes (Z is up due to the way the IMU was soldered to that particular headset). IMU observations were recorded at a rate of 256Hz: Time in seconds, Tri-axial velocity (rotational rate) in deg/s, tri-axial acceleration in g (m/s^2), and tri-axial magnetometer flux readings in Gauss (G).

1. Read and import the provided (.csv) dataset (5 marks). You can use a CSV reader, but it might be easier to write a bit of custom read code since you know the format of the dataset. Convert rotational rate to radians/sec (1 mark).
2. Implement *your* methods to (i) convert Euler angle readings (radians) to quaternions (1 mark), (ii) calculate Euler angles from a quaternion (1 mark), (iii) convert a quaternion to its conjugate (inverse rotation) (1 mark), and (iv) calculate the quaternion product of quaternion a and b (1 mark).

PROBLEM 3, TRACKING: POSE CALCULATION - 20 MARKS:

1. Implement a dead reckoning filter (using only the gyroscope-measured rotational rate): Calculate current orientation by using a previously determined orientation (consider the initial orientation q[0] to be the identity quaternion [1,0,0,0]) and re-evaluate that orientation based on an estimated speed over the elapsed time (5 marks).
2. Add gravity-based tilt correction by including accelerometer information in the computation: Transform acceleration measurements into the global frame (2 marks). Calculate the tilt axis (2 marks) and find the angle ϕ between the *up* vector and the vector obtained from the accelerometer (2 marks). Use the complementary filter to fuse the gyroscope estimation and the accelerometer estimation (4 marks). Upon firing up the engine, the first sequence should show the headset in the middle of the viewport rotating based on the fused gyroscope & accelerometer data.
3. Try a few different alpha values (e.g., 0.01, 0.1, ...), investigate and comment on their effect on drift compensation in your report. Implement any other processing of the accelerometer values that you consider important / necessary and discuss this in the report. (5 marks)

PROBLEM 4, PHYSICS - 10 MARKS:

1. Implement simple distance-based collision detection between the objects. Use spheres of an appropriate radius as bounding regions to perform the calculation. For your demo scene, arrange the headsets on the floor in such a way that a few collide and change direction. (5 marks)
2. Simulate friction to render the gradual slowing down of moving objects having an initial velocity on the floor in the rendering engine. No need to use actual forces / accurate physics simulation, as long as the simulated effect looks plausible. (5 marks)

PROBLEM 5 - POST-PROCESSING - 15 MARKS:

1. Choose **one** effect from this list: *bloom, motion blur, depth-of-field*, and implement it in the engine by doing your own research on how the effect works and is implemented in a rendering engine. (10 marks)
2. Implement colour support in the engine, enabling the headsets being different colours in the final output. (5 marks)

PROBLEM 6, RESEARCH REPORT & VIDEO - 25 MARKS:

The main purpose of the research report is to include evidence of testing, provide images/answers to the questions below, and for you to comment on anything else that you consider important or were asked in the problems above.

1. Produce static renders for the report, demonstrating that your tracking and transformation matrices work. **Do not forget to attach a short video containing two sequences, demonstrating the full tracked motion of the headset, one for problem 3.1 and one for problem 3.2 as visual proof that your implementation runs/works.** (5 marks)
2. What are the advantages and disadvantages of using simple dead reckoning vs including the accelerometer? Support your claims with data from your simulation. (7 marks)
3. Comment on the performance of the methods in point 2 above, i.e., comment on the expected number of calculations for each method. (2 marks)
4. Could positional tracking be implemented from the data provided? If so, how? Comment on the expected accuracy of positional tracking based on the IMU data. What are some potential limitations of this approach? (2 marks)
5. Comment on collision detection using spheres. How could you improve collision detection and what would the effects of those improvements be on performance? Support your arguments with data for the particular 3D model of the headset that you were provided with. (4 marks)
6. Discuss the perceptual repercussions/effect of the post-processing effect that you chose to implement on the human visual system. (5 marks)

References

[1] CANN, E. RenderPy. `https://github.com/ecann/RenderPy`, 2023. [Online; accessed 9-Jan-2023].

[2] LAVALLE, S. M., YERSHOVA, A., KATSEV, M., AND ANTONOV, M. Head tracking for the oculus rift. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (2014), IEEE, pp. 187–194.