Durham University

Computer Science

Summative Assignment

Module code and title	COMP3617 Virtual and Augmented Reality
Academic year	2022/23
Submodule title	N/A
Coursework title	VR Coursework
Coursework credits	10 credits
Lecturer	George Koulieris
Deadline*	Thursday, May 04, 2023 14:00
Hand in method	Ultra

Additional coursework files	3D Graphics engine code, VR headset 3D model, additional literature related to motion tracking, physics, and distortion correction. All files are available in the Blackboard coursework folder.
Required submission items and formats	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. Include a readme file with instructions on how to run your program and what external resources you require if this is not obvious. (b) A .pdf report no longer than 6 pages including images (max ~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 (compressed to <100MB).

*** This is the deadline for all submissions except where an approved extension is in place. Late submissions received within 5 working days of the deadline will be capped at 40%. Late submissions received later than 5 days after the deadline will received a mark of 0.**

COMP3617 Summative Assignment
Virtual & Augmented Reality

Important, please read first!

- The assignment should be submitted via Blackboard; the deadline is May 4th, 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 additional functionality (e.g., NumPy). However, the 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 these algorithms.
- 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 requested sequence on screen. Include a readme file with instructions on how to run your program and what external resources you require if this is not obvious. (b) A .pdf report no longer than 6 pages including images (max ∼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 (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., should last ∼ 27 seconds. When running your software though, it is ok if the rendering takes more time - software renderers are slow.
- 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 (pp 15-16): link.
- The Virtual & Augmented Reality module is only assessed by this coursework (100% of the module mark).
- A FAQ section in Blackboard will be updated with questions as they arise.

You work at a VR company, and you currently participate in the development of a prototype 3D graphics engine capable to render content for VR headsets. You have been tasked to work on the 3D engine to develop basic rendering, tracking, physics, and distortion pre-correction for VR. Your are provided with a rudimentary 3D engine [2] (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 distortion correction. Your task list is below. You should demonstrate your development in a demo scene by rendering a few VR headsets (3D model is provided) falling from the sky under the effect of gravity while the camera is yawing, pitching and rolling based on the provided (in Blackboard) real headset tracking dataset.

Before proceeding, please read LaValle's relevant chapters in the free book available in Blackboard, attend all relevant lectures and read [4] carefully (Please note the typo in the order of operations in eq. 8 & 10).

Additional helpful but not necessary resources can be found in Blackboard ([1, 3])

PROBLEM 1, RENDERING - 15 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. (1 mark)
2. Implement perspective projection instead of orthographic projection. You will need to extend some elements of the engine to handle homogeneous coordinates. (6 marks)
3. Implement the basic transformation matrices, in particular add the ability to translate, rotate, and scale objects. You should use these in the demo scene, e.g., rotating the headsets as they fall from the sky. (8 marks)

PROBLEM 2, TRACKING: HANDLING POSITIONAL DATA - 15 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) - the flux readings will not be used in this coursework.

1. Read and import the provided (.csv) dataset (6 marks).
2. Convert rotational rate to radians/sec and normalize the magnitude of both the accelerometer and magnetometer values taking special care of NaN divisions (5 marks).
3. 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: CAMERA POSE CALCULATION - 20 MARKS:

1. Implement a dead reckoning filter (using the gyroscope-measured rotational rate) with gravity-based tilt correction (using the accelerometer data). First calculate current position by using a previously determined position (starting at the identity quaternion) and re-evaluate that position based on an estimated speed over the elapsed time (5 marks). Consider the initial orientation q[0] to be the identity quaternion [1,0,0,0].
2. Then you will include 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 virtual camera should rotate based on the fused input data (gyroscope & accelerometer).
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, DISTORTION PRE-CORRECTION - 10 MARKS:

Account for the distortion introduced by a headset lenses by implementing distortion pre-correction. The formulas that you will use are based on a heavily simplified version of Brown's model, that only uses two coefficients, $c1$ and $c2$ for radial distortion, omitting the p-coefficients required for tangential distortion correction. The formulas below consider that world points which are inside the canonical viewing frustum have x and y coordinates ranging from -1 to 1. We will refer to these points using polar coordinates:

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \operatorname{atan2}(y, x)$$

Brown's simplified Forward radial transform; r_d is the distorted radius, r_u is the undistorted radius:

$$r_d = f(r_u) = r_u + c_1 r_u^3 + c_2 r_u^5$$

You will implement pincushion distortion pre-correction by warping the frame buffer appropriately. Assuming that the lens is radially symmetric, the pincushion distortion can be described as a stretching of the image that becomes increasingly severe away from the optical axis. To correct the pincushion distortion effect, you will need to apply an appropriate transformation so that the light emitted by the screen reaches the eye undistorted. Use the simplified radial distortion model detailed above. The actual parameters you use do not matter but the inverse distortion generated should be clearly visible.

PROBLEM 5, PHYSICS - 20 MARKS:

1. Implement simple Physics in the engine, simulating gravity acceleration and air resistance applied on the falling objects. See additional literature in the Blackboard coursework folder for formulas on calculating air resistance. Choose arbitrary values for the *drag coefficient* (e.g., 0.5), *air density* (e.g., $1.3kg/m^3$) and *area* (e.g., $0.2m^2$). (10 marks)
2. 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 objects in such a way so that a few collide and change direction. (10 marks)

PROBLEM 6, RESEARCH REPORT & VIDEO - 20 MARKS:

The main purposes 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, for different orientations of the camera, demonstrating that your tracking and transformation matrices work. Do not forget to attach a short video demonstrating the full tracked motion of the camera. (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. (5 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. Based on how you implemented distortion correction in this CPU-based software renderer, how could this technique be accelerated when implemented in a GPU-accelerated engine? Discuss all possible options. (2 marks)

References

[1] Brown, D. C. Decentering distortion of lenses. *Photogrammetric Engineering and Remote Sensing* (1966).

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

[3] De Villiers, J. P., Leuschner, F. W., and Geldenhuys, R. Centi-pixel accurate real-time inverse distortion correction. In *Optomechatronic Technologies 2008* (2008), vol. 7266, International Society for Optics and Photonics, p. 726611.

[4] 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.