School of Computing

Module Coordinator Other lecturers	Dr. Jiacheng Tan <jiacheng.tan@port.ac.uk>
Date Issued	TBC
Code	GACV/M30242
Title	Graphics and Computer Vision

Schedule and Deliverables

Item	Value	Format	Deadline	Lat deadline Ecf deadline
Coursework	50%	A single .zip file containing all source code and documents	2021-01-15 23:59 [GMT]	2021-01-29 23:59
Final Examination	50%	Paper-based, closed book examination.	Timetabled assessment period: 2021-01-25 to 2021-02-05	

Notes and Advice

- The Extenuating Circumstances procedure is there to support you if you have had any circumstances (problems) that have been serious or significant enough to prevent you from attending, completing or submitting an assessment on time.
- ASDAC are available to any students who disclose a disability or require additional support for their academic studies with a good set of resources on the ASDAC moodle site
- The University takes plagiarism seriously. Please ensure you adhere to the plagiarism guidelines. And watch the video on Plagiarism
- Any material included in your coursework should be fully cited and referenced in **APA 7** format. Detailed advice on referencing is available from the library
- Any material submitted that does not meet format or submission guidelines, or falls outside of the submission deadline could be subject to a cap on your overall result or disqualification entirely.
- If you need additional assistance, you can ask your personal tutor, student engagement officer ana.baker@port.ac.uk , academic tutor xia.han@port.ac.uk or your lecturers.

M30242 Graphics and Computer Vision 2020-21

Coursework

This assessment accounts for 50% of the overall assessment of the module. It consists of two tasks that assess the learning outcomes 3 and 4.

This assignment document contains two appendices.

Warnings:

1. The work you submit must be of your own. You may reference the numerous examples on the web, but you must NOT copy any of the examples. Failure to comply with this requirement may result in plagiarism procedures being taken.
2. The programs you submitted must be compatible with the software and hardware settings on standard University lab PCs without requiring installation of any additional libraries or updates. It is your responsibility to make sure all the files are intact and compatible with version of the software provided by the IT Service of the university (see https://servicedesk.port.ac.uk/ for detailed information)

Submission

1. Submit all the deliverables of both Task 1 and 2 in the form of ***a single zip file named with your student number*** through the ***Coursework Submission*** folder on the unit base on Moodle.
2. Submission on Moodle will open a week before the deadline and close at 23:59 on the deadline day.
3. The submission must be anonymous. Do **NOT** write your name on any artefact.
4. Emailed work will **NOT** be accepted.

Task One (50%)

Warning: For this task, you are NOT allowed to use any WebGL library that has built-in functions for creating, drawing, texturing geometric primitives such as spheres, cubes, and so on. You need to generate the vertex data of such objects, i.e., vertex coordinates, texture coordinates and normals, and perform the relevant operations, such as texture mapping, lighting and shading calculations, by yourself.

Specification

In this task, you are required to create an animation. The scene consists of the planet earth in the middle with a satellite orbiting around it along a circular orbit within the horizontal plane. The scene is illuminated from top-right by a *directional* light that is at a 60-degree angle with the horizontal plane if viewed in front view.

The earth model is a sphere of radius 10 and mapped with an earth image. The earth rotates slowly around its own vertical axis. An image of the earth is provided for texturing the sphere.

The satellite consists of a main body of a cube of size of 2x2x2 and two "solar panels" that are attached to the two opposite sides of the main body through two connection "rods". The rods are cuboids of size 0.2x0.2x0.5 and *golden* in colour. The solar panels are *blueish* thin rectangular objects of 1x2 in size. For simplicity, we assume that the panels always face upwards. One side of the cube representing the main body of the satellite has a *black* colour, which will constantly face the earth while orbiting the earth. A *golden* antenna *dish* of a diameter of 4 is attached to the black side by a *golden* rod of 0.2x0.2x0.4. The antenna will face the earth.

The animation will be interactive: You should be able to control the radius of the circular orbit (with the left and right arrow keys) and the speed (up and down arrow keys) of the satellite at runtime. You should have full viewport/scene-navigation control: translations along x- (shift plus mouse drag), y- (alt plus mouse drag) and z-direction (mouse wheel) and rotations around x- and y-axes (mouse drags). The translation controls should be independent of the rotation controls.

Your application should work with standard browsers on University lab PCs without requiring any special set-up or configuration of software or hardware. Firefox browser is preferred because textures may not work properly in Google Chrome if its security policy prevents loading texture files locally. You should extensively test the animation controls to ensure that any control action will not cause the system to freeze, crash, or any scene objects to disappear or behave in a strange way.

Deliverables

1. The source code of the entire WebGL application and any necessary supporting files such as libraries and textures. The program code should be suitably commented and come with necessary instruction for using it.

2. An electronic copy of a short report (no more than 1000 words) that documents the design/implementation decisions, difficulties or problems (if any) and evidence and/or conclusions of test and evaluation of the application against the specification.

Task Two (50%)

Application Scenario and Conditions

To control the traffic at the entrance of a narrow tunnel, a computer vision system is used to intercept oversized or speeding vehicles. Any vehicle that exceeds 2.5m in width and/or 30 miles per hour in speed will be diverted or stopped by traffic lights or police officers upon receiving the warnings from the system. Fire engines (assuming that they are mainly red in colour and have a width/length ratio of approximately 1:3) are the only vehicles exempted from the control.

The system consists of a video camera fixed right above the centre of the lane of a straight road and is 7m off the ground. Its optical axis is 30 degrees off the horizon and pointing to the lane along the direction of traffic. A diagram showing the camera configuration is given in the Appendix. The resolution of the sensor of the camera is 640X480 pixels. For simplicity, it is assumed that the pixels of the camera sensor are square and each sensor pixel is equivalent to 0.042 degrees in view angle. The camera grabs a video frame (an image) of the lane at 0.1s intervals. It is further assumed that each frame contains only one vehicle. The output frames from the camera will be the only input to the vision system. You should not make any assumptions about the identity of the vehicle contained in a frame before processing the frame.

Requirements

1. *Functions or Functional Blocks*. Design and implement in Matlab a computer vision application that fully meets the specification. Your application should contain the following basic functions or functional blocks: Size detection, Speed detection, Colour detection

2. *Outputs and Information*. Your program should output the results of the key processing steps, for example smoothed images, colour blobs, centres, outlines or bounding box, and document them in the report (see *Requirement 4*) in the form of screenshots.

Your program must output the following values or information in the command line window:

```
Car width:       meters
Car length:      meters
Car width/length ratio:
Car colour:
Car speed:       mph
Car is speeding (Y/N):
Car is oversized (Y/N)
Car is fire engine (Y/N):
```

3. *Test and Verification*. Include the following script (name it as **`Test.m`**) in your application and use it to test your application. The coursework assessors will run the script to verify the intermediate and final outputs of the application.

```
clear,
close all,
img1=imread("xxx.jpg"),
img2=imread("xxx.jpg"),
my_application(img1,img2)
```

Speeding detection could be tested using image pairs *001.jpg* vs *002.jpg, 001.jpg* vs *003.jpg , 001.jpg* vs *004.jpg*, and so on by assuming that the images are taken at 0.1s interval in each case. Fire engine and oversized vehicle should be tested using the relevant images.

4. *Documentation and Report*. Prepare a report of no more than 1000 words that documents the system design, testing and evaluation. The report should

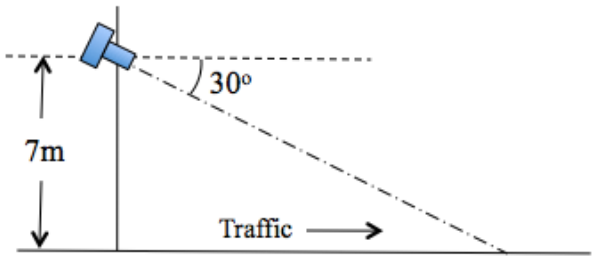
- include discussions of the necessary conditions of and/or assumptions about the application that may help to justify your design decisions;
- provide a concise flow diagram of the entire system to show the important processing steps;
- justify each processing step by discussing or showing the features you want to extract, the method used for extracting the features, and the screenshots of the input and output of the processing step;
- document your test against the application scenarios by providing the inputs, e.g., the names of image files, and outputs, e.g., the screenshot of the speed/width/colour of a vehicle and any command-line messages.

Deliverables

1. Program code and supporting files. The program code should be in M-file format and suitably commented and with instructions to use/run it.

2. A copy of the coursework report that meets *Requirement 4*.

Appendix 1 Camera system configuration



Appendix 2 Marking Scheme

Task One (50%)		
Object models Accurate models of each elements of the scene.	**15**	
Surface attributes Colours and/or texture mapping.	**6**	
Lighting Use of light and any justification	**6**	
Animation and interactive parameter control Overall animation, satellite orbit & speed control	**12**	
Scene navigation control Translation & rotation controls	**6**	
Test & evaluation Report	**5**	
Task Two (50%)		
Correct detection of traffic • Speeding detection (30mph). • Oversize vehicle detection (2.5m). • Fire engine detection.	15	
Feature selection and detection • Selection of appropriate features. • Correct detection methods/procedures & appropriate parameters. • Quality of extracted features.	10	
Testing & Evaluation • Testing script • Evidence & documentation (screenshots and discussions) of test against traffic scenarios	10	
Report • Discussions of any conditions/assumptions • Flow diagram and justification (For each processing step: feature & detection-method selection, screenshots of inputs and outputs) • Documentation and discussion of test against traffic scenarios & evaluation against requirements	15	