COMP3170 Assignment 1 – UFO

Topics covered:

- 2D mesh construction
- HSB and RGB colour spaces
- 2D transformations: rotation, translation, scale
- Animation
- Scene graph
- Coordinate frames
- 2D camera: view and projection matrices, aspect
- Instancing

Task Description

Your task is to implement a scene with a night-time city skyline and with a background starfield (Figure 1). A UFO flies over the scene, controlled by the keyboard.

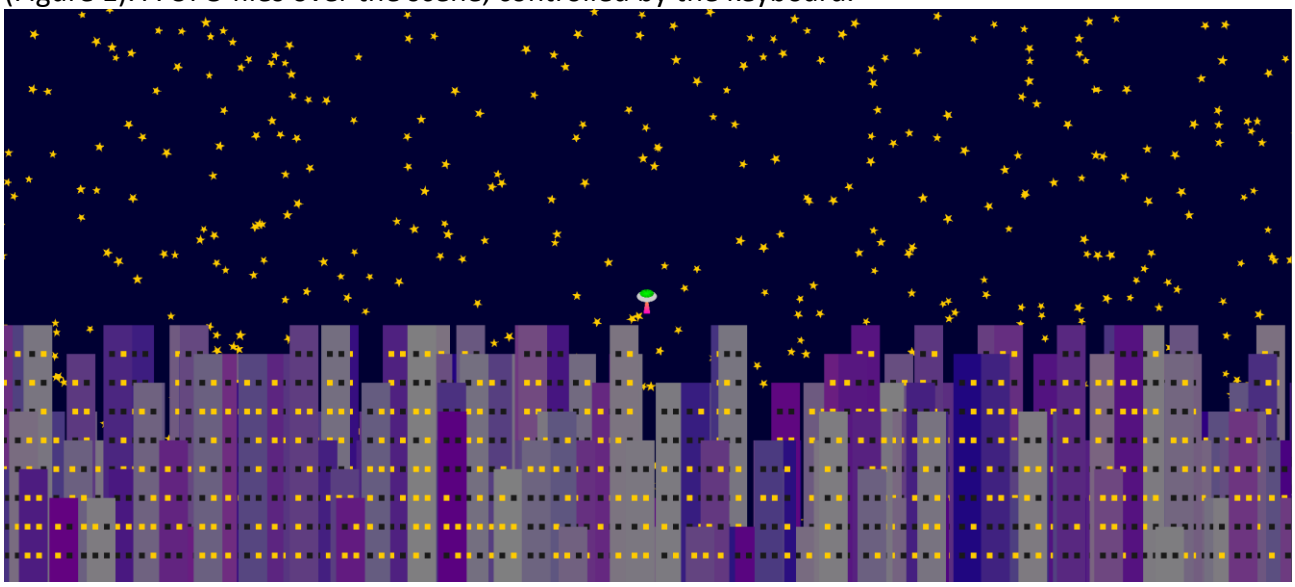

Figure 1: A UFO flying over a night-time city.

Framework

For this assignment, you will need the Eclipse project containing the LWJGL and JOML libraries that we have been using in the workshop classes. See the Week 1 workshop for instructions to download and install this project. Make sure to pull the latest version of the library from the repository before beginning the project.

A basic framework for the assignment has been provided in GitHub Classroom:

https://classroom.github.com/a/tN_DQDUD

Accept the assignment to clone it into your own Github account.
It contains the files:

- Assignment1.java – The "bare-bones" driver class for the project.
- Scene.java – an example of a Scene class.

- UFO.java – an example of a class using the SceneObject class.
- simple_vertex.glsl / simple_fragment.glsl – a basic, simple shader.

To complete the assignment, you will need to edit these files and add further classes (and shaders) of your own.

Components

You are required to complete each of the components below. Each component contributes a percentage towards your Completeness mark, as described below. Note that not all components are of equal difficulty.

Note: In the spec, some specific numbers (e.g. the colour and size of the UFO, etc) are not specified. You are free to choose whatever values you feel appropriate for these as long as they illustrate the behaviour required. However, remember to use named constants in your code (with comments to indicate units) to allow these values to be easily modified. Clarity marks will be deducted for using 'magic numbers' (i.e., embedded numerical constants given without explanation).

General Requirements

You should use the SceneGraph implementation provided in the comp3170.lwjgl project to organise the objects in your scene.

Throughout this document, we will refer to "world units". A world unit is defined as a single unit of world space. For example, an object with the following Model to World Matrix:

$$\begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Has a scale of 2 (along the x and y axis) in world units, and is 2 world units right and 3 world units up from the world origin. Please contact us if you are still unsure what this means – it can be confusing!

Building – Mesh (5%)

Create a building mesh with a rectangular shape. The building must be at least 10 world units tall. At the start of every 5 units up, the building must feature two square windows, as illustrated in Figure 2.

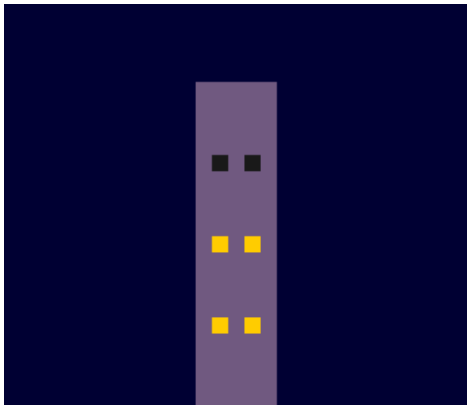

Figure 2: A single building, 20 world units tall (with windows every 5 units up – so at locations 5, 10 and 15).

Building – Colour (5%)

Building-window colours should alternate between yellow and black. For the body of the building, set the hue to a colour of your choice, with a randomly chosen saturation value from 0% to 100%, and a brightness of 50% in HSB space, as shown in Figure 3.

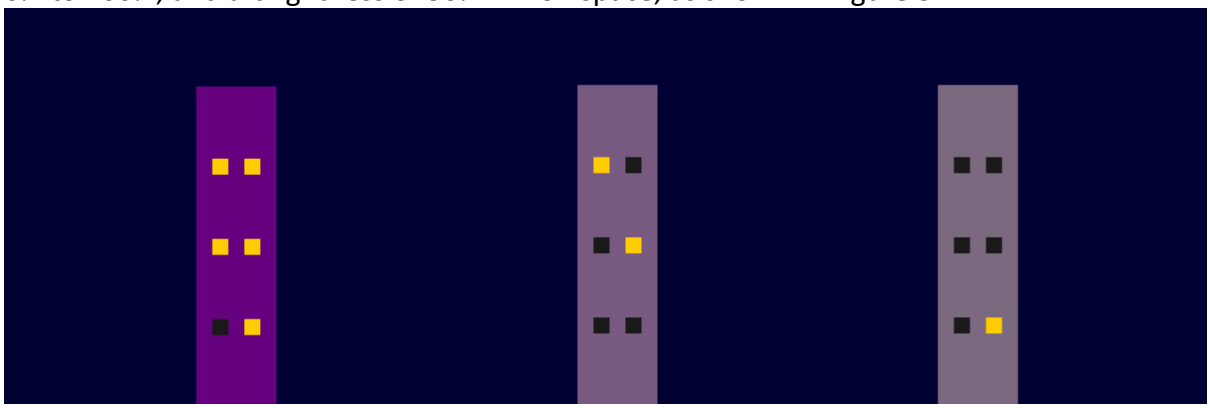

Figure 3: Various saturation of buildings from purple to grey.

Building – Height/Floors (5%)

Vary the building height between 10 and 50 world units, with extra windows added accordingly (every 5 units).

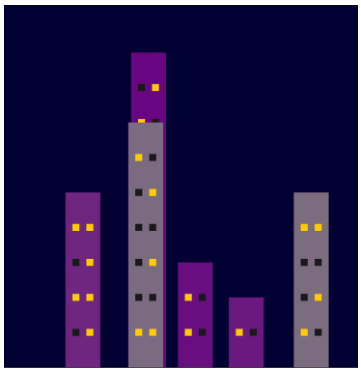

Figure 4: A collection of buildings of different heights.

Skyline (5%)

Clear the background to a dark colour. Randomly position 500 buildings along the bottom of a 500 x 100 area of world units, as shown in Figure 5.

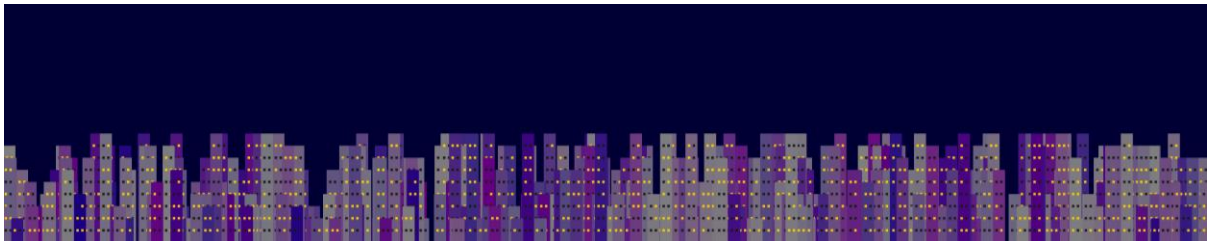

Figure 5: A skyline of buildings. Image captured from a window sized 2000 x 400 pixels, representing a 500 x 100 area of world units.

Starfield (5%)

Randomly position 1000 stars around the 500 x 100 area of world units, as shown in Figure 6. Each star should have exactly five points. Stars should be of varying size and rotation.

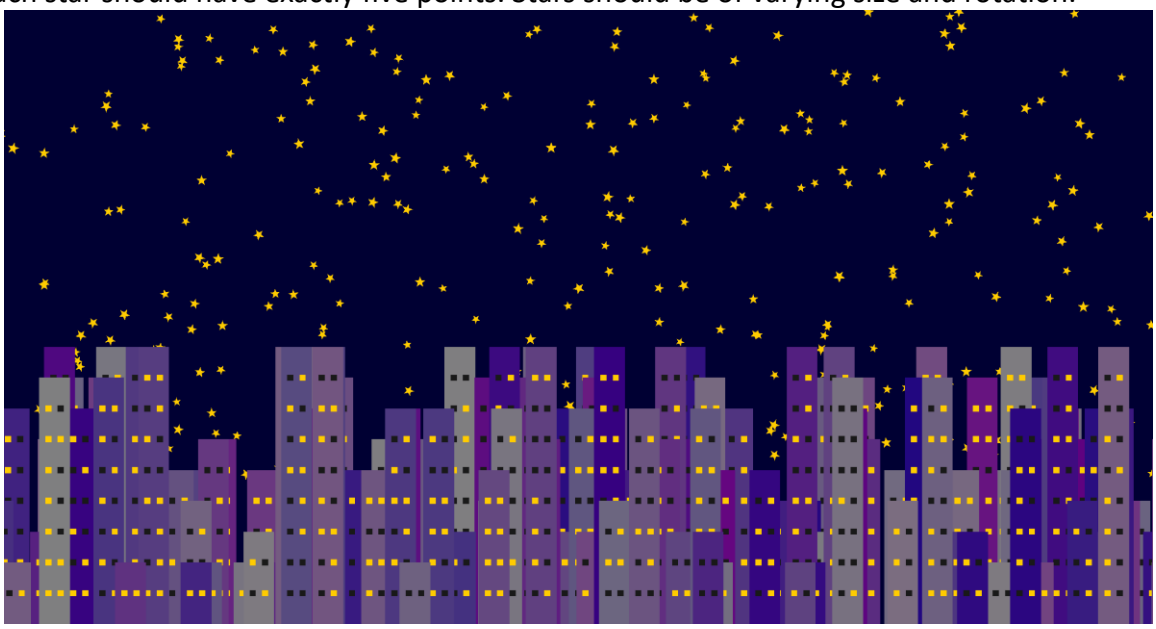

Figure 6: A field of stars rendering behind the skyline, of varying sizes and rotations.

UFO – Mesh (10%)

Create a mesh of a UFO with a frame (grey in colour, circular shape), a cockpit (a colour of your choice, circular shape) and a tractor beam (a cone, colour of your choice), as shown in Figure 7.

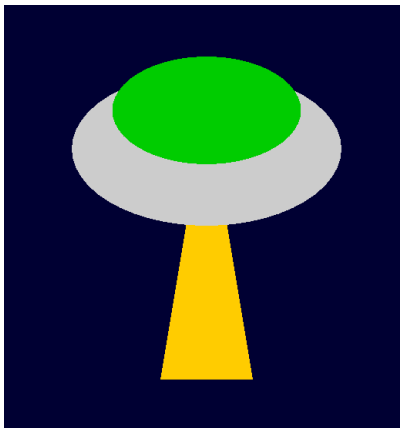

Figure 7: A UFO with cockpit and tractor beam.

UFO – Vertex colouring (5%)

Use vertex colouring to have the UFO cockpit's colouring change between sides, and the tractor beam to change colour from top to bottom, as shown in Figure 8.

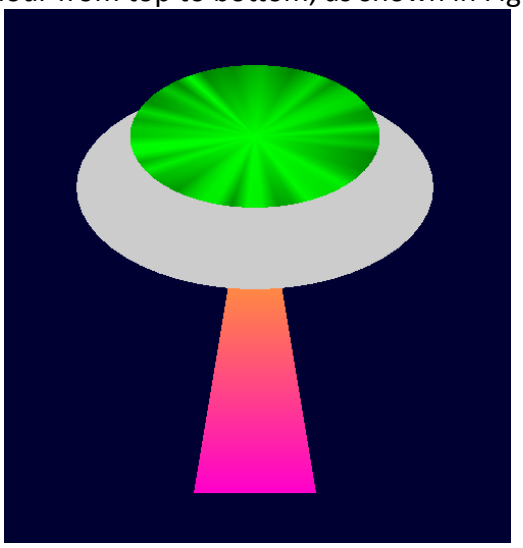

Figure 8: A UFO with cockpit and tractor beam.

UFO – Animation/Movement (5%)

The UFO moves up, down, left and right in world space at a set speed, adhering to the following keyboard input:

W – Up
S – Down
A – Left
D – Right

See the video on iLearn for a demonstration of this transformation.

UFO – Tractor beam Animation (5%)

The tractor beam can rotate in an arc of 60 degrees left and right, and be scaled along its y axis. The scale of the tractor beam must not go below 0. This rotation and scaling adheres to the following keyboard input:

Up arrow – Shrink tractor beam
Down arrow – Grow tractor beam
Left arrow – Rotate left
Right arrow – Rotate right

See the video on iLearn for a demonstration of these transformations.

World camera (10%)

Create a camera that follows the UFO. The camera should remain stationary, except for if the UFO's mesh reaches the left or right quarters of the view, in which case the camera should move left and right with it. The camera should remain aligned with the world X/Y coordinates and should not rotate. See figure 9, and the video on iLearn, for a demonstration of this.

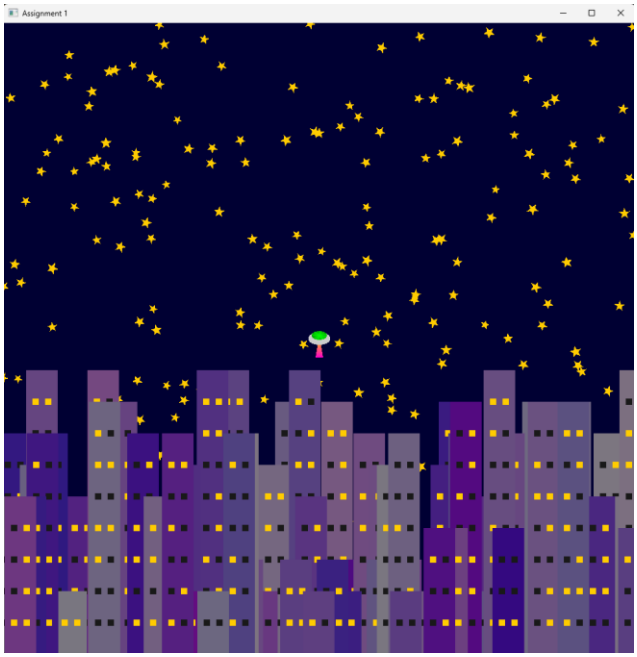

Figure 9: A camera centred on the UFO.

World camera – resizing (10%)

The view volume of the camera should be adjusted proportional to the size of the window, with scaling along the y axis.

A 1000x1000 window should show a 100x100 area of world space. Making the window's width larger (or smallewr) should reveal more (or less) of the world without changing the screen size of objects displayed. Making the window taller (or shorter) should scale objects. See Figure 10, and the video on iLearn, for a demonstration of this.

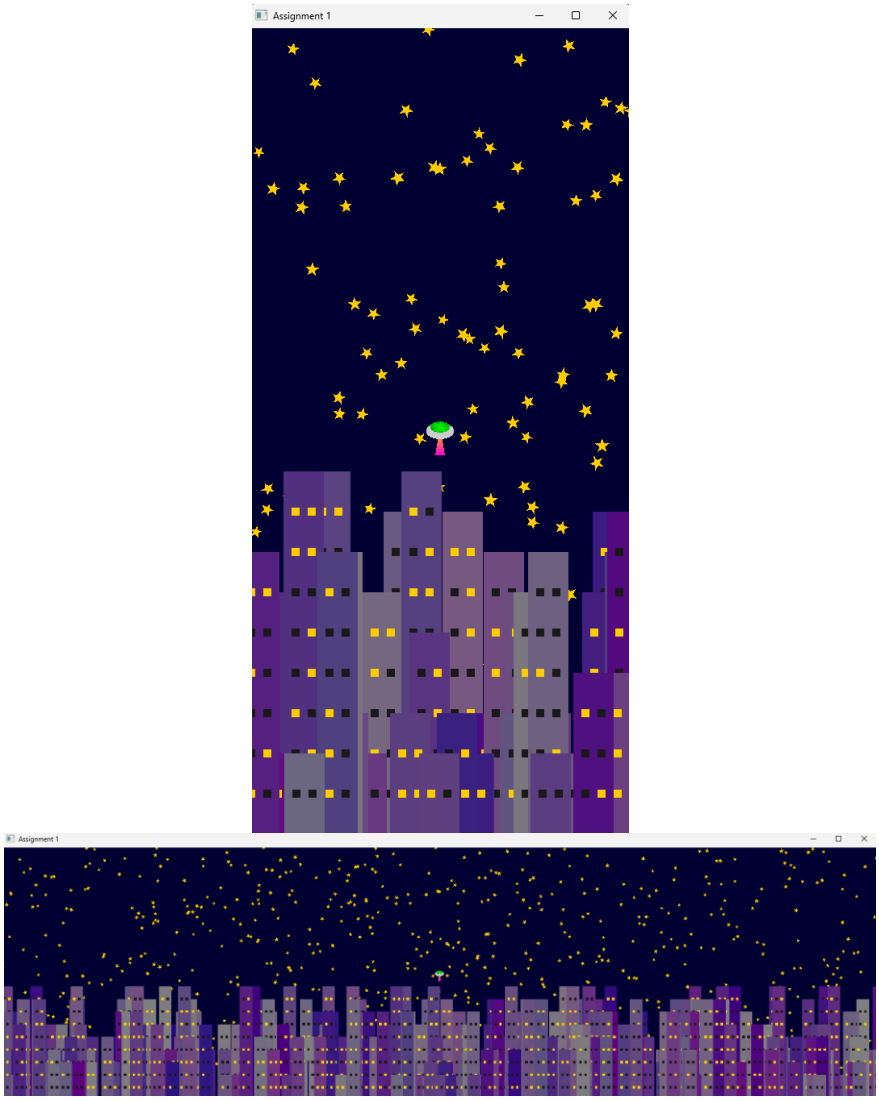

Figure 10: Two images showing the screen at different dimensions, with corresponding scaling.

Distinction and HD level tasks

The above tasks are enough for you to earn a Credit (with credit-level effort/skill applied). The below tasks are more challenging, and should only be attempted by students aiming for Distinction and HD marks. We recommend completing the above tasks first before attempting these.

Instancing (10%)

Implement the starfield using instancing so all the stars are drawn in a single draw call.

Parallax scrolling (5%)

As the UFO moves left and right, the starfield will slowly move in the opposite direction, giving a sense of depth. See the video on iLearn for a demonstration of this.

Ripples on tractor beam (5%)

The tractor beam features ripples of another colour running along its length, as shown in Figure 11 (note: this is taken from another sample solution of the project, with some variance in approaches and colourings).

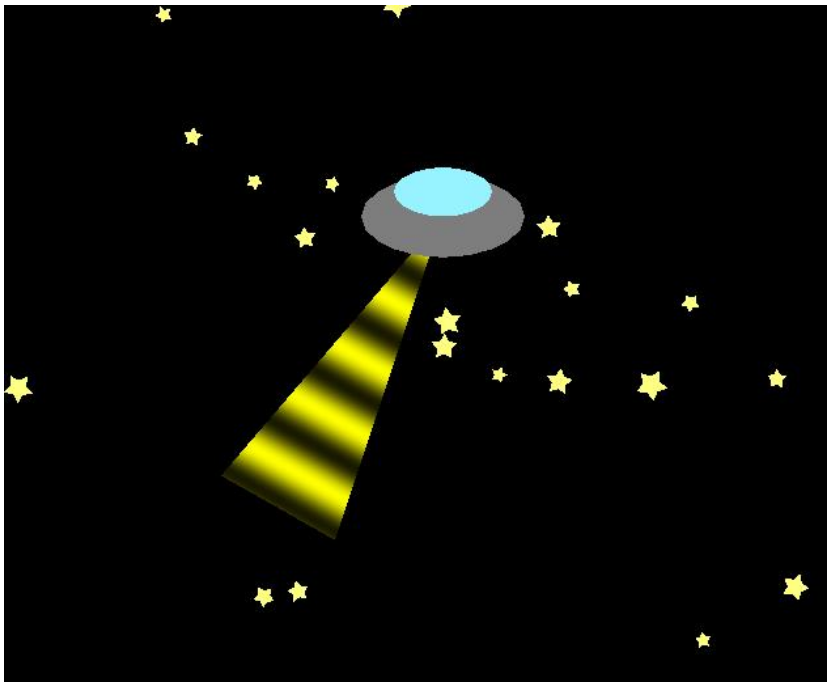

Figure 11: The UFO with ripples along its tractor beam.

Animated ripples on tractor beam (5%)

The tractor beam's ripples are animated, moving continuously down to its base or upwards to its frame. See the video on iLearn for a demonstration of this (note: this is taken from another sample solution of the project, with some variance in approaches and colourings).

Mouse-guided Tractor Beam (5%)

The rotation and length of the tractor beam is determined by the mouse's position relative to it, so it will always rotate towards and stretch to meet the mouse. See the video on iLearn for a demonstration of this.

Documentation

In addition to your code, you should submit a PDF report following the Word template provided. The report should include:

- A completed table indicating the features you have attempted.
- An illustration of the scene graph used in your project.
- Illustrations of all the meshes used in your project, drawn to scale in model coordinates, including:
    - the origin
    - the x and y axes

    - the coordinates of each vertex
    - the triangles that make the mesh
- A diagram illustrating the world camera calculations, including:
    - The viewport rectangle
    - The mapping from view (camera centric) coordinates to NDC
    - The mapping from NDC to viewport (pixel) coordinates

Please use a ruler when drawing, and ensure your drawings are clear. Marks may be deducted for messy or unclear drawings.

Submission

Your **Java project** will be submitted using Github Classroom. Your most recent commit to the repository before the assignment deadline will be marked. Practice good version control habits of regular commits with clear and meaningful commit messages.

Your **report** will be submitted using iLearn as a PDF.

Marks

Your marks will be calculated using three components (according to the rubric below):

- **Correctness:** Whether your code is correctly implemented.
- **Clarity:** Whether your code is easy to understand.
- **Documentation:** Whether your report contains all the required elements.

This mark will be averaged with your **Completeness** mark, which refers to the total value of the components you have attempted. Each component is worth between 5% and 10%, as detailed in the Report Template.

Your final mark will be determined using the following formula:

AVERAGE(**Completeness**, SUM(**Correctness** *60% + **Clarity** * 20% + **Documentation*** 20%))

So, for example if you attempt 80% of the features above, with perfect correctness (100%), slightly sloppy code (70%) and some minor sloppiness in the document (80%), your final mark would be:

AVERAGE (80% * (60% * 100% + 20% * 70% + 20% * 80%))
AVERAGE(80% * (60% + 14% + 16%))
AVERAGE(80% * 90%)
= 85%

On the other hand, if you only attempt 50% of the features above, to the same level of quality (90%) your final mark would be:

AVERAGE(50% * 90%) = 70%

Rubric

Grade	Correctness (60%)	Clarity (20%)	Documentation (20%)
HD (100)	Excellent work. Code is free from any apparent errors. Problems are solved in a suitable fashion. Contains no irrelevant code.	Good consistent style. Well structured & commented code. Appropriate division into classes and methods, to make implementation clear.	All sections are complete and accurately represent the code. All diagrams are neat, clear, and well annotated.
D (80)	Very good work. Code has minor errors which do not significantly affect performance. Contains no irrelevant code.	Code is readable with no significant code-smell. Code architecture is adequate but could be improved.	All sections attempted with minor sloppiness or missing detail. No discrepancies between documentation and code.
CR (70)	Good work. Code has one or two minor errors that affect performance. Problems may be solved in ways that are convoluted or otherwise show lack of understanding. Contains some copied code that is not relevant to the problem.	Code is readable but has some code-smell that needs to be addressed. Code architecture is adequate but could be improved.	All sections attempted with minor sloppiness or missing detail. Minor discrepancies between documentation and code.
P (60)	Poor. Code is functional but contains major flaws. Contains large passages of copied code that are not relevant to the problem.	Significant issues with code quality. Inconsistent application of style. Poor readability with code-smell issues. Code architecture could be improved.	All sections attempted with significant sloppiness and missing detail. Minor discrepancies between documentation and code.
F (0-40)	Code compiles and runs, but major elements are not functional.	Significant issues with code quality. Inconsistent application of style. Poor readability with code-smell issues. Messy code architecture with significant encapsulation violations.	Some aspects incomplete. Diagrams unclear and badly drawn. Does not make use of graph paper. Coordinate systems not properly annotated. Major discrepancies between documentation and code.