Mark Scheme

Marks for your assessment will be allocated based on a matrix marking scheme (available in the project brief) that is used in a centre weighted assessment of the holistic mark across all categories. You should use this as a check list during time you work on this project and **submit this self-evidence marking sheet** with the completed work.

The three categories are equally weighted and are:

1. *Programming, structure and design*
2. *Scene Graph Manipulation and Interaction*
3. *Scene Creation and Enrichment*

The full marking scheme is provided below and you should use this as the template for your **self-evidence**.

Submission:

You must submit the following items through blackboard:

1. A .zip file containing your final visual studio solution directory – this is the entire directory and must include everything to run the code. Please use the following filename convention: **hughes_chris_assignment2.zip – obviously replace my name with your name!**
2. The root of the zip file must also contain a pdf version of this mark scheme including your self-assessment with the name: **hughes_chris_markscheme.pdf** . This means for each of the four categories you must:
    a. Highlight one column (only 1) which you believe matches the equivalent level of the work you have completed.
    b. A (500 word maximum) justification for your mark – This will explain what you have done to meet the highlighted grade column and may reference your source code (filename and line number ranges) and justify how your work is of an equivalent level to the column – basically tell us what you did to get the marks.
    c. At the end of the mark scheme is a box where you must list all of the files you have created or modified from the template.

Version: 21st February 2023

Programming, Structure and Design

0-19%	20-39%	40-59%	60-79%	80-100%
Programming, structure and design	Inconsistency in naming styles and conventions Poor use of formatting (white space etc) No generic code reuse, with the majority of functionality bespoke to the task. Correct application and configuration of MS VCs C++ project environment variables. No coherent structure to the programme code Poor/weak functional design with little consideration of roles and responsibilities.	Some use of accepted C++ file structure for classes and functions Inconsistent formatting of code Elements of generic functionality, allowing some reuse, but inconsistently applied Use of bespoke classes and structures to handle and manage scene and interaction	Well defined and decomposed code structure with all classes adhering to C++ file structure and file based decomposition of functions. Constant and clear formatting of all code throughout Generic code used in all cases, where possible. Some use of toolkit classes in a specialised form to achieve functional goals Appropriate use of non-OO code for specific elements. Simulation and animation handled by own (non-toolkit) systems, bespoke to the application domain	Evidence of planned and optimised class based structure that encapsulates multi-functional, generic , reusable structures in all cases. Creation and application of own generic dynamic data structures to manage run-time data. Decomposition of programme features to generic control pipelines for each feature set with isolated specific functionality managed within top level structures Refined class structure, based on toolkit specialisation, for all functional goals Some use of advanced dynamic data handling systems (eg stl) for some functional goals. Use of own file I/O functionality to save and restore state of application but not the configuration of the scene Simulation based on bespoke components unique for each instance of use

I believe I have used accepted file structures; however, I also have inconsistent formatting of code. I have created lots of helper classes but I do realise that some variables and functions should be protected and structured more consistently.

Scene Graph Manipulation and Interaction

0-19%	20-39%	40-59%	60-79%	80-100%
Scene Graph Manipulation and Interaction	Use of a simple, flat, scene structure, without decoration, to maintain the scene description Definition of own tools for describing the loaded model data structure that do not utilise the provided toolkit.	Definition and application of tools to query loaded data structure to identify specific node types by name. Scene comprised of multiple instances and decomposed features of the loaded models within a single coherent scene structure Development of scene structure with additional nodes for managing space and state in addition to the loaded models. Functional wrapper created for the control of a single traffic light working within the time managed SG environment and implemented as a scene decoration dynamic/interactive features enacted by bespoke tools outside of the provided toolkit	Implementation and application of features to encapsulate model aspects as functional components within a bespoke environmental management context Bespoke functional wrapper for the control of a collection of traffic lights forming a single junction with lights operating as a single entity maintaining sequence, enacted within the SG time constrained system as a decoration object Use of own structures/features to define a unique animation path for the car and enact continual playback without consideration of junction controls. Some dynamic/interactive features enacted within the scene graph toolkit as generic components with high level application specific configuration	Implementation and application of features to encapsulate all model aspects as functional components within a bespoke environmental management context Extended call-back decorator classes to report motion events to animation control system Refined hierarchic class structure to unify control callback nodes within a single generic interface Provision of scene structures and mechanisms to enable runtime addition of artefacts to the scene Inclusion of invisible control artefact to detect motion/presence and control response Use of standard toolkit features to define and control animation Multiple forms of navigation based on user input Animation control points derived from road structure Most dynamic/interactive features enacted within the scene graph toolkit as generic components with high-level application specific configuration

I have made the facardes more bespoke for my purpose. I added in the function of using the traffic system for traffic lights so as to keep track of them in cars collision logic. The collision logic with other cars works and I used a line to quad type of collision detection and I do believe it is a very advanced

form of collision logic for this assignment. However, this doesn't work completely because when the slower car moves forward, the car before it recognises that it is not colliding anymore so it goes back to its speed, this makes it go into the next car until it checks for collisions again.

The traffic light collision doesn't work as well as I wanted as when a car goes through the T Junction, it works but after it recognises that it is aligned with the previous traffic light. Had I had more time I would have fixed this using the same collision detection as the car to car but I didn't have the time to implement it.

Scene Creation and Enrichment

	0-19%	20-39%	40-59%	60-79%	80-100%
Scene Creation and Enrichment	Models loaded and positioned with appropriate scale, but without decomposed arrangement	Construction of scene from models provided with decomposed arrangement to define a unique road way. Single car static ally located within the model Provision of a single traffic light with light sequence control	Extended construction of scene from models provided with decomposed arrangement to define a unique road way. Inclusion of additional static geometric structures appropriate to the scenario Single car model following pre-defined animation path with consideration of the light state at junctions Provision of a multiple traffic lights with coherent junction sequence control	Refined and extensive road structure Refined application of static 3d geometric models to develop rich environment Provision of environmental features to show effects (fog, day/night, etc) with simple timed progression Inclusion of dynamic entities (eg animated/ simple simulation control) within the scene Interactive 3D entities within the scene controlled by indirect interaction (ie keyboard)	Extensive and refined 3d static model User controllable environmental features to show effects (fog, day/night, etc) with simple timed progression Interactive 3D entities within the scene controlled by direct interaction (ie mouse) (1 or 2 examples)

I have created my own model of the car in raaAssignment2.cpp with random colours assigned when being created in raaCarFacarde.cpp. I have added a green ground that does not overlap with the original roads. If I had more time I would have implemented a tiling system that added random scene enrichments to empty parts of the scene. You can see remnants of this code in raaAssignment2.cpp

List of files created or modified from the template (please include the full path from your directory root):

- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaCarFacarde.cpp
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaCarFacarde.h
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaAssignment2.cpp
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaCollisionTarget.h
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaTrafficSystem.cpp
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaTrafficSystem.h
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaTrafficLightFacarde.cpp
- raaOSGAssignment-template\raaAssignment2\raaAssignment2\raaTrafficLightFacarde.h