Train2Game Game Developer - Portfolio Project Marking Criteria

Each Portfolio Project will be judged against a set of criteria which are based on the main aspects of the overall task. Each of these criteria will be assigned a grading band from those shown below:

- **Unsatisfactory**
  Essentially, a part of the assignment has been totally missed out (for example, if one of the criteria is to describe or implement NPC behaviour but this is omitted), or no evidence is provided that time has been spent working on that part.

- **Needs Improving**
  Key parts don't quite meet what is expected, or may be incomplete or not completed in the right way. However, there is evidence in the work that it is understood what is required and an attempt has been made at least. With possibly minor improvements or revision this would be able to pass, although a significant amount of work would likely be required in order to produce a higher quality of work.

- **Satisfactory**
  This is essentially a simple "pass". The work covers what is needed to pass but there is a lot of room for improvement. The work would probably show a good start and, with improvement, could be easily upgraded to a **Clear Pass** or **Excellent**.

- **Clear Pass**
  Indicates that the work has been completed well and, although there may be room for improvement, the work is of significant enough merit to be above the bare minimum needed to pass.

- **Excellent**
  This indicates either a high quality (there would not be anything that can be done to improve that part of the assignment) or high completion (there is nothing left to do to complete that part of the assignment).

To achieve an overall pass for a Portfolio Project, each of the criteria must achieve a **Satisfactory**, **Clear Pass**, or **Excellent** grading. A single grading of **Unsatisfactory** or **Needs Improving** against one or more of the criteria will result in the Portfolio Project needing to be reattempted.

A breakdown of the criteria for each of the Portfolio Projects follows.

Marking Criteria

Game Developer

Developer Section 2B - Games Design/Technical Design Document

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Presentation Style, layout and format of the document and spelling, punctuation and grammar. **33%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Communication How well ideas and concepts are communicated and how the language adheres to industry-relevant terminology. How the use of flowcharts assists with explanation and communication of ideas and concepts. **33%**	No original ideas or concepts communicated, or the language is too difficult to properly understand what is trying to be communicated. No flowchart or diagrams.	Although it is evident that ideas and concepts are being communicated, they are difficult to properly understand. The language used may be inappropriate or immature, inconsistent or convoluted with too much un-necessary information. Inappropriate or irrelevant diagram(s) or flowchart(s) included.	Generally well communicated, although there is occasionally some ambiguity over what is trying to be communicated or it is unclear what is being said. Any included diagram(s)/flowchart(s) omit some information or are unclear.	Ideas and concepts are well communicated with very little refinement or rewording required. Flowchart(s) and/or diagram(s) are included and complete.	Ideas and concepts communicated in a clear and concise way using relevant industry terminology where appropriate. Accompanying diagram(s)/flow chart(s) provide are appropriate, clear and compliment the report well.
Technical considerations Investigation into technical issues relating to the implementation of the design. **34%**	No consideration of any technical issues.	Mention of some technical issues but largely irrelevant, incomplete or flawed in premise.	Some basic key issues have been considered. Little or no consideration of any of the mentioned issues may be resolved	A range of issues have been considered with some view to how they may be overcome	Comprehensive review of a range of technical issues with possible suggested solutions
Grade: / 10					

Developer Section 2C - Pathfinding

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the program fulfils the requirements. **20%**	Program unfinished or fails to run.	An attempt has been made to implement pathfinding, however the program fails to pathfind between two points reliably.	Program can pathfinding between two points, but is fairly “crude” in it’s demonstration. The program is limited in scope and does not demonstrate any additional functionality.	Program can pathfind between two points with some additional features, such as collision avoidance or variable path costs.	Program pathfinds between two points. Evidence of collision detection and/or avoidance. Variable path costs implemented.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands how the pathfinding in their program works. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain what the program does but little or no understanding is evident of how it works. More needs to be written about how the pathfinding algorithm works and how it has been implemented in their program.	Explanation is given as to how the program works and it is evident that the student understands the basics of their algorithm. Some confusion or omission of steps in the algorithm is evident.	A good piece of documentation that explains the workings of the program and how their pathfinding algorithm works.	Comprehensive documentation that compliments the program and fully explains the functionality, as well as demonstrating the student's full understanding of the workings of their pathfinding algorithm.
Grade: / 10					

Developer Section 2C - Sprite Blitting

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the program fulfils the requirements. **20%**	Program fails to run or does not demonstrate any successful blitting of sprites.	An attempt has been made to combine a sprite with a background, although there may be visual errors in the resulting image. Limited or no additional features considered.	Program achieves blitting between images in memory. Consideration shown for some additional functionality, such as using double buffering, or different depth buffers, although these show limitations in their implementation.	Program demonstrates a solid implementation of blitting and makes use of multiple additional features.	Program shows an excellent implementation of blitting with a range of implementations, such as multiple depth buffers, colour keying and alpha blending. Some animation is also included with some consideration given to compression.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands how the sprite blitting works. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain what the program does but little or no understanding is evident of how it works. More needs to be written about how sprite blitting works and how it has been implemented in their program.	Explanation is given as to how the program works and it is evident that the student understands the basics of sprite blitting.	A good piece of documentation that explains the workings of the program and how their pathfinding algorithm works.	Comprehensive documentation that compliments the program and fully explains the functionality, as well as demonstrating the student's full understanding of the workings of their pathfinding algorithm.
Grade: / 10					

Developer Section 2C - Data Serialisation

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the program fulfils the requirements. **20%**	Program fails to run or does not show evidence that data has been saved or can be loaded.	Program shows an attempt to save data but the data may not be able to be read accurately. Data storage or loading may is limited to a single format and the code may show that it would be difficult to adapt for different formats	Program demonstrates saving and reading of data with few errors in re-loading of data. Limitation is shown in the scope of the system with regards to reusability and how often it is used within the program.	A well-implemented system that saves and reads data with accuracy in different formats.	An excellent system has been implemented that allows for accurate saving and reading of data in a range of formats. Evidence that extra formats could be added with little additional work required.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands how data serialisation works in their program. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain what the program does but little or no understanding is evident of how it works. More needs to be written about how data serialisation works and how it has been implemented in their program.	Explanation is given as to how the program works and it is evident that the student understands the basics of what data serialisation is.	A good piece of documentation that explains the workings of the program and how their data serialisation implementation works.	Comprehensive documentation that compliments the program and fully explains the functionality, as well as demonstrating the student's full understanding of the workings of their data serialisation system.
Grade: / 10					

Developer Section 2C - Procedural Generation

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the program fulfils the requirements. **20%**	Program fails to run or there is no evidence that there is any form of procedural generation included in the program.	Some implementation of pre-defined procedural generation is evident although dynamic generation while the program running is not.	Data is successfully generated within the program, although the pseudo-random generator may produce seemingly non-random results.	Program uses an algorithm to dynamically generate of data values while the program is running.	Program uses procedural generation in an effective way, implementing a random number generator and using fractals. At least some consideration has been given to the optimisation of the algorithm used to generate data.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands how procedural generation works in their program. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain what the program does but little or no understanding is evident of how it works. More needs to be written about how procedural generation works and how it has been implemented in their program.	Explanation is given as to how the program works and it is evident that the student understands the basics of what procedural generation is.	A good piece of documentation that explains the workings of the program and how their procedural generation works.	Comprehensive documentation that compliments the program and fully explains the functionality, as well as demonstrating the student's full understanding of the workings of procedural generation. Optimisation of procedural generation algorithms is also discussed.
Grade: / 10					

Developer Section 2C - Memory Management

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the program fulfils the requirements. **20%**	Program fails to run or there is little or no evidence or memory management.	Although the program attempts to deal with the allocation of resources, there are noticeable errors in the produced solution.	The program allocates and deallocates memory whilst the program is running. Memory leaks are not handled and fragmentation has not been dealt with.	The manager attempts to implement a way to deal with fragmentation that occurs over time in the program or is able to "detect" leaks that may occur.	The manager allocates and deallocates resources using an efficient choice for a search algorithm. A solution is implemented to deal with fragmentation and memory leaks with a way to trace back a memory leak to the source.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands how procedural generation works in their program. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain what the program does but little or no understanding is evident of how it works. More needs to be written about memory management, its implementation and how it is evident in their program.	Explanation is given as to how the program works and it is evident that the student understands the basics of memory management.	A good piece of documentation that explains the workings of the program and how their memory manager works.	Comprehensive documentation that compliments the program and fully explains the functionality, as well as demonstrating the student's full understanding of the workings of memory management. The student also discusses fragmentation and ways in which it can be dealt with.
Grade: / 10					

Developer Section 3A – Working in 3D

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the program fulfils the requirements. **20%**	Game world fails to run or has too little functionality to meet any of the requirements.	Some features have been added to the world although large omissions noticeable and implemented features are unfinished.	World shows progress although there is still a distinct lack of required features. Some interaction implemented, such as movement.	Project is largely complete with most features implemented, although some may still require completion. World is interactive and implemented features show signs of polish.	Project is complete with all features implemented and well polished. Some creativity shown with additional features implemented, or some attempt made to string the features together to make a game of some sort.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands the features implemented in their game world. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain how the world works although documentation lacks substance necessary to demonstrate understanding. Omission of readme file or other key parts of the documentation used to explain what is implemented.	Explanation is given as to how the game world functions and it is evident that the student understands the concepts of 3D games programming.	A good piece of documentation that explains the workings of the game world and the features within it.	Comprehensive documentation that compliments the program and fully explains the functionality, as well as demonstrating the student's full understanding of the 3D programming features implemented.
Grade: / 10					

Developer Section 3B – Working in 3D Part 2

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the game fulfils the requirements. **20%**	Game fails to run or has too little functionality to meet any of the requirements.	Part 1 has been attempted although large omissions noticeable. Little or no attempt has been made at implementing other parts. Gameplay is present but game is incomplete.	Parts 1 and 2 mostly implemented with some attempt made at other parts. Game shows structure and is playable, although lacks polish.	Project shows a good deal of completion with parts 1 – 3 largely complete and progress made on other parts. Game shows some polish and can be largely completed.	All parts implemented and the game is well polished and playable from start to finish.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands the features implemented in their game. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain how the game and the features work although documentation lacks substance necessary to demonstrate understanding. Omission of readme file or other key parts of the documentation used to explain what is implemented.	Explanation is given as to how the game and the features function and it is evident that the student understands some additional concepts of 3D games programming.	A good piece of documentation that explains the workings of the game and the features within it and how it was developed with mention of bugs or issues that remain.	Comprehensive documentation that compliments the game and fully explains the implemented features as well as how the game developed and explaining any issues that may still remain in the program.
Grade: / 10					

Developer Section 3C – Deep Sea Diver

	Unsatisfactory **2** *Little or no evidence of an attempt. Significant rework required in order to achieve a pass.*	**Needs improving** **4** *Work shows evidence of understanding but needs some revision in order to pass*	**Satisfactory** **6** *A pass, although the work would benefit significantly from improvements.*	**Clear pass** **8** *A sound piece of work with very few flaws*	**Excellent** **10** *A high standard piece of work with no significant flaws.*
Code Presentation Indentation, layout and commenting of code. Use of (appropriate) naming conventions. **20%**	No comments, little or no indentation, code is hard to follow, variables are unintuitive named.	Variables are named but may be unclear as to what they are supposed to represent. Code layout and commenting is evident but may be inconsistent or unclear.	Some attention has been paid to the layout of the code but the code would benefit from being easier to read. Naming conventions and indentation is inconsistent or needs improving. Comments are reasonable but could be improved to explain the code better.	A good attempt at presenting the code but some inconsistencies in the names of variables and indentations. More comments may be beneficial or may need more elaboration.	Code layout is clear, consistent, properly indented and well commented. Variables are named consistently and are and prefixed according to industry standard naming conventions.
Programming quality How well the code has been written in order to make the program run, with particular focus on the use of OO programming and the use of accessors and mutators. **20%**	Program fails to run.	Little or no evidence of any OO programming, accessors/mutators either unused or used wrongly. Code is un-necessarily repeated. Program suffers from poor coding.	Some evidence of code re-use and accessors and mutators have been implemented for variable access. The program runs fairly well but the code is inefficiently written.	Well written code. Code is mostly reused where possible. Variables are appropriately protected and accessed through accessors and mutators. Program performance is acceptable.	The code shows adherence to object-orientated programming principles with appropriate usage of accessors, mutators and constant variables. The program runs well and some attempt has been made to optimise the code.
Program functionality How the game fulfils the requirements. **20%**	Game fails to run or shows no expansion on previous work	Game shows some evidence of additional features although these are limited in number and lack originality and scope and may not work as intended.	Game shows some originality and a variety of added features, although these lack polish and their functionality is limited. Some additional work may need to be done to make the added features work as intended.	A wide range of features are implemented within the game and the game has been polished to integrate these added features. Steps have been taken to customise the look and feel of the game to add originality.	Game is notably different from the starting project and incorporates all features mentioned in the brief. Features work well and the game is well polished.

Documentation Presentation Style, layout and format of the document and spelling, punctuation and grammar. **20%**	Poor layout and grammar, inconsistent formatting and style and unintuitive to follow.	Some attempt made to format the document, although coherence is sometimes hindered by lack of a proper structure. Grammar and/or punctuation needs improvement.	Some presentation formatting is evident and the document is clear to follow. Occasional errors in grammar and/or punctuation.	A well laid out document with only minor flaws, or inconsistency, in formatting. Only minor and occasional errors in spelling and punctuation.	Professionally prepared document with a consistent layout and style. No significant errors in punctuation and grammar.
Evidence of understanding through Documentation How the documentation shows that the student understands the features implemented in their game. **20%**	No documentation submitted or documentation is irrelevant to the requirements.	An attempt to explain how the game and the features work although documentation lacks substance necessary to demonstrate understanding. Omission of readme file or other key parts of the documentation used to explain what is implemented.	Explanation is given as to how the game and the features function and it is evident that the student understands some additional concepts of 3D games programming.	A good piece of documentation that explains the workings of the game and the features within it and how it was developed with mention of bugs or issues that remain.	Comprehensive documentation that compliments the game and fully explains the implemented features as well as how the game developed and explaining any issues that may still remain in the program.
Grade: / 10					