Faculty of Technology – Course work Specification 2017/18

Module name:	**Object Oriented Design and Development with C++**		
Module code:	**IMAT2605**		
Title of the Assignment:	**2D Physics Puzzler**		
This coursework item is: (delete as appropriate)	Summative		
This summative coursework will be marked anonymously	No		
The learning outcomes that are assessed by this coursework are: 1. Be able to critically evaluate how and when different OO techniques and design patterns should be used in order to solve problems typically found in software development 2. Be able to effectively communicate OO designs through UML static class diagrams and a set of documented C++ classes 3. Be able to analyse the correctness and performance of C++ code using appropriate software tools 4. Be able to synthetize a C++ OO software solution for a real-time simulation or game problem			
This coursework is: (delete as appropriate)	Individual		
This coursework constitutes 50% **to the overall module mark.**			
Date Set:	**2nd October 2017**		
Date & Time Due:	**26th of April 2018 4pm**		
The 'normal' coursework return date for this work is: 24th of May 2018. *If for any reason this is not forthcoming by the due date your module leader will let you know why and when it can be expected. The Head of Studies (headofstudies-tec@dmu.ac.uk) should be informed of any issues relating to the return of marked coursework and feedback.*			
When completed you are required to submit your coursework to: 1. Report (PDF document) to be submitted via a Turnitin link on blackboard. 2. Documented software to be submitted via a submission link on blackboard. Instructions on how to do this are posted on blackboard. 3. Video demonstrating the functionality/gameplay of your coursework.			
Late submission of coursework policy: Late submissions will be processed in accordance with current University regulations which state: *"the time period during which a student may submit a piece of work late without authorisation and have the work capped at 40% if passed is **14 calendar days**. Work submitted unauthorised more than 14 calendar days after the original submission date will receive a mark of 0%. These regulations apply to a student's first attempt at coursework. Work submitted late without authorisation which constitutes reassessment of a previously failed piece of coursework will always receive a mark of 0%."*			
Academic Offences and Bad Academic Practices: These include plagiarism, cheating, collusion, copying work and reuse of your own work, poor referencing or the passing off of somebody else's ideas as your own. If you are in any doubt about what constitutes an academic offence or bad academic practice you must check with your tutor. Further information is available at: http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-offences.aspx and http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-practice.aspx			

Tasks to be undertaken:

You are to design, build and test a 2D puzzle game which makes some use of real-time physics.
Specification of game mechanics:

- Static graphical background
  - A picture which forms the background of the game
- Moving game objects
  - Some kind of game object (actor) which:
    - Undergoes realistic motion
    - Collides with other game objects
- Implementation of defined rules for winning and/or losing
- A head-up-display (HUD)
  - HUD can be simple text or a more advanced graphical element
  - Could show any aspect of game information

Deliverables to be submitted for assessment:

- A short report of no more than 4 sides of A4 explaining the design decisions of your software, and the testing regime used with relevant test data.
- A visual studio project containing the game source code and any unit tests.
- HTML or PDF (via LaTeX) documentation report on the code produced using Doxygen. This should be included with your software submission
- A video explaining the game mechanic and demonstrating the level of functionality achieved. You may also use this as an opportunity to highlight any areas of code you are particularly proud of.

How the work will be marked:

Broadly your mark will be based on how much of the specification you have completed and how you have achieved it. Good OO design and good code standards all improve your mark. You should make use of the techniques you have been taught to improve the efficiency, maintainability and reusability of your code.
Note: You may use a physics engine (I would recommend Box2D or Bullet) however, this will alter how the work is marked. Using an engine is likely to make the OO design considerably trickier.

The mark will be assessed using the following mark scheme:

- 0 – 20%
  Game mechanics have not been achieved. Little or no work has been done in implementation or design. Ignored code standards. No documentation. No report.
- 20 – 40%
  Games mechanics have partially been achieved, a significant portion of the mechanics have not been implemented. Design has not followed OO principles and lacks thought. No testing. Poor code standards. No documentation. Cursory report lacking any detail.
- 40 – 50%
  The fundamental game mechanics have been implemented, albeit with some bugs. Design considers OO principles in main aspects. Efficiency of the code has not been considered. Testing lacks rigour. Poor code standards. Some

attempt at documentation. Basic report lacking in detail.

- 50 – 60%
  The majority of the game mechanics have been implemented, albeit with some bugs. Design considers OO principles in the main aspects, achieving a design with some abstractions and reusability. Efficiency of the code has not been considered. Testing is weak in places. Code standards considered in some places. Core code is documented. Report gives decent overview of the design decisions made.
- 60 – 70%
  Nearly all of the game mechanics have been implemented and the game is largely bug free. The physics is realistic and the game plays well. Design considers OO principles throughout achieving a design with good abstractions, reusability and to some degree elegance perhaps using patterns. Efficiency of the code has been considered in key aspects of the game. Testing is thorough. Code standards are very good and all code is documented.
- 70 – 85%
  Some extensions have been well implemented in addition to the prescribed mechanics. Design considers OO principles throughout achieving a design with good abstractions and reusability, arriving at an elegant design making use of design patterns. Efficiency of the code has been considered throughout the game. Testing is thorough. Code standards are very good and all code is documented.
- 85 – 100%
  A game has been implemented which goes significantly beyond the simple mechanics to produce an impressive game at near commercial quality. Design considers OO principles throughout achieving a design with good abstractions and reusability, arriving at an elegant design making use of patterns and interface classes. Efficiency of the code has been considered throughout the game and a profiling report is included. Testing is thorough. Code standards are very good and all code is documented.

It should be clear that these criteria are subjective and open to interpretation. To ensure a high mark you need to achieve good functionality with high quality code. Strong functionality will not make up for poor hacked together code. High quality code will not make up for a lack of functionality.

A sample of up to 10% of the work will be assessed with a viva. This will not form part of your assessment but will explore any concerns the module team have about academic offences and bad academic practice. Ensure the work is your own – you may be selected for a viva and have to explain all aspects of your code. If you cannot explain your code you will receive a mark of zero and may be referred to the academic offences officer.

Module leader/tutor name:	**Simon Coupland**
Contact details:	**simonc@dmu.ac.uk**