DEPARTMENT OF COMPUTER SCIENCE

ASSESSMENT DESCRIPTION 2013/14 (EXAM TESTS AND COURSEWORK)

MODULE DETAILS:

Module Number:	08101	Semester:	1
Module Title:	Programming 1		
Lecturer:	RSM		

COURSEWORK DETAILS:

Assessment Number:	1	of	1		
Title of Assessment:	Hyperspace Cheese Battle Development				
Format:	Program	Report	Demonstration		
Method of Working:	Individual				
Workload Guidance:	Typically, you should expect to spend between	20	and	35	hours on this assessment
Length of Submission:	This assessment should be **no** more than: ***(over length submissions will be penalised as per University policy – see below)***	1500 **words** *(excluding diagrams, appendices, references, code)*			

PUBLICATION:

Date of issue:	15th November

SUBMISSION:

ONE copy of this assessment should be handed in via:	E-Bridge		If Other (state method)	
Time and date for submission:	**Time**	9:00	**Date**	10th December
If **multiple hand–ins** please provide details*:*				
Will submission be scanned via TurnitIn?	No	If this requires a separate TurnItIn submission, please provide instructions:		
*Late submissions **will be** penalised as per university policy*				

The assessment must be submitted **no later** than the time and date shown above, unless an extension has been authorised on a *Request for an Extension for an Assessment* form which is available from the Departmental Office (RB-308) or http://intra.net.dcs.hull.ac.uk/student/exam/Advice%20regarding%20resits%20in%20modules%20passed%20by%20compe/Forms/AllItems.aspx.
If Turnitin is taking a long time to produce its analysis you must still submit your work, albeit initially without the Turnitin analysis. A delay at Turnitin cannot be used to excuse a late submission.

MARKING:

Marking will be by:	Student Name

COURSEWORK COVERSHEET:

BEFORE submission, you **must** ensure you complete the **correct** departmental ACW cover sheet (if required) and attach it to your work. The coversheets are available from: http://intra.net.dcs.hull.ac.uk/student/ACW%20Cover%20Sheets/Forms/AllItems.aspx	NO coversheet required

ASSESSMENT:

The assessment is marked out of:	100	and is worth	40	% of the module marks
N.B If multiple hand-ins please indicate the marks and % apportioned to each stage above (i.e. Stage 1 – 50, Stage 2 – 50). It is these marks that will be presented to the exam board.				

ASSESSMENT STRATEGY AND LEARNING OUTCOMES:

The overall assessment strategy is designed to evaluate the student's achievement of the module learning outcomes, and is subdivided as follows:

LO	Learning Outcome	Method of Assessment *{e.g. report, demo}*
1	*Intellectual Skills: Identify problems which are amenable to computer based solutions and suggest how such a solution can be structured and deployed.*	Software Demonstration
2	*Intellectual Skills: Analyse simple data processing problems and express solutions in an object oriented programming language.*	Software Demonstration
3	*Practical Subject Specific Skills: Apply knowledge of the syntax and semantics of a specific programming language.*	Software Demonstration
4	*Practical Subject Specific Skills: Make use of appropriate tools to create and deploy simple programs in a specific programming language.*	Code Review
5	*Transferable Skills: Understand how problems can be systematically analysed, and solutions planned.*	Code Review Documentation Submission

Assessment Criteria	Contributes to Learning Outcome	Mark
Program works correctly	1,2,3,4,5,	40%
Evidence of Good Design	1,2,4,5	10%
Appropriate Program Enhancements	1,2,3,4	20%
Appropriate Documentation	2,5,6	10%
Evidence of Appropriate Testing	1,5	10%
Appropriate Layout and Identifier Selection	4,5	10%

FEEDBACK

Feedback will be given via:	Verbal (via demonstration)	Feedback will be given via:	Feedback Sheet
Exemption (staff to explain why)			
Feedback will be provided no later than 4 'semester weeks' after the submission date.			

This assessment is set in the context of the learning outcomes for the module and does not by itself constitute a definitive specification of the assessment. If you are in any doubt as to the relationship between what you have been asked to do and the module content you should take this matter up with the member of staff who set the assessment as soon as possible.

You are advised to read the **NOTES** regarding late penalties, over-length assignments, unfair means and quality assurance in your student handbook, also available on the department's student intranet at:

- http://intra.net.dcs.hull.ac.uk/student/ug/Handbooks/Forms/AllItems.aspx (for undergraduate students)
- http://intra.net.dcs.hull.ac.uk/student/pgt/Student%20Handbook/Forms/AllItems.aspx (for postgraduate taught students).

In particular, please be aware that:

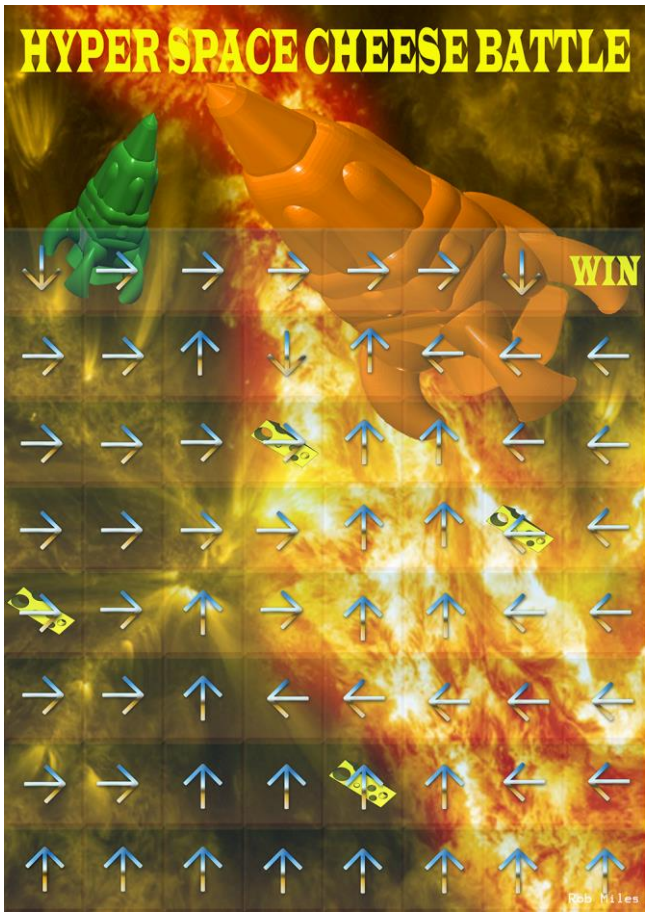
- Your work will be awarded zero if submitted more than 7 days after the published deadline.
- The overlength penalty applies to your written report (which includes bullet points, and lists of text you have disguided as a table. It does not include contents page, graphs, data tables and appendices). Your mark will be awarded zero if you exceed the word count by more than 10%.

Please be reminded that you are responsible for reading the University Code of Practice on the use of Unfair means (http://student.hull.ac.uk/handbook/academic/unfair.html) and must understand that unfair means is defined as any conduct by a candidate which may gain an illegitimate advantage or benefit for him/herself or another which may create a disadvantage or loss for another. You must therefore be certain that the work you are submitting contains no section copied in whole or in part from any other source unless where explicitly acknowledged by means of proper citation. In addition, **please note** that if one student gives their solution to another student who submits it as their own work, **BOTH** students are breaking the unfair means regulations, and will be investigated.

In case of any subsequent dispute, query, or appeal regarding your coursework, you are reminded that it is your responsibility, not the Department's, to produce the assignment in question.

The Game of "Hyperspace Cheese Battle"

Hyperspace Cheese Battle is a game of intergalactic conflict and racing. And cheese. Between 2 and 4 players can play. Each player controls a rocket which they move through space. Moves are controlled by the use of a multi-faceted random value indicating system, otherwise known as a dice. Players roll the dice to move their rockets onto space quadrants, otherwise known as squares. Certain parts of space are infused with "Cheese Power" which can be used by the advanced technology in the ships to perform special actions.



Rocket Moves

Each player takes it in turn to throw one dice. They start off the board at the bottom left hand corner and move their rocket onto the board with their first throw of the dice. At each turn they must then move their rocket that number of squares over the board in the direction of the arrow on that square. If their throw would take them off the board their rocket is not moved.

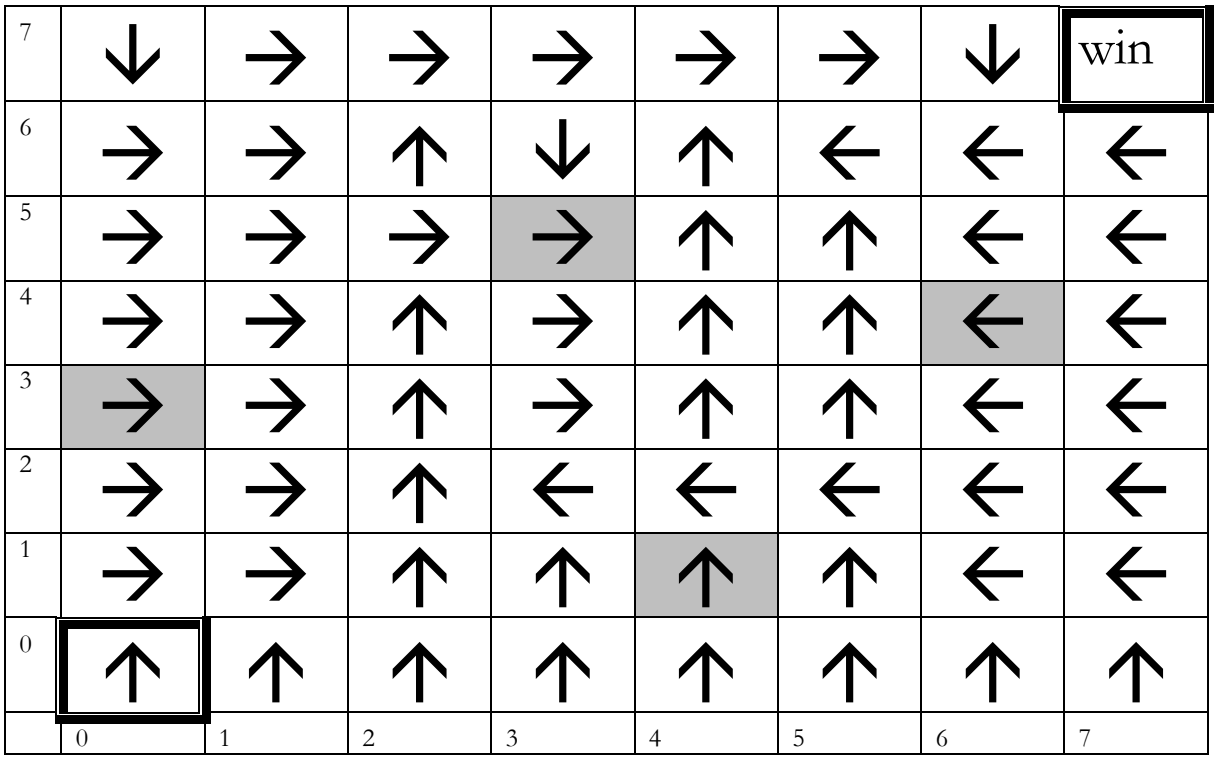
If they land on a cheese square they can perform one of two actions:

- Absorb the cheese power from the square and use it to refuel their engines, allowing them to roll their dice again for an extra move.
- Absorb the "Cheese Power" from the square and create a "Cheese Deathray" which they can fire at any another player. This causes the engines of that rocket to explode, sending it back to the bottom row of the board. The player being sent to the bottom can choose any unoccupied square on the bottom row.

Play continues until a player manages to travel to the square at the top right hand corner. The first player to do this is the winner of the game.

The laws of space and time do not allow a rocket to occupy the same hyperspace as another. If a move would cause a rocket to land on an occupied square the rocket is moved to the next free square in the direction of the arrow on the occupied square. If that square is also occupied the rocket is moved in the direction of the arrow of the next occupied square until an empty square is found.

Sample Gameplay



Boad coordinates showing Cheese Squares

Arnold (A), Betty (B), Cedric(C) and Damian(D) all sit down for a game of “Space Cheese Battle”.

1. A rolls a 2, follows the arrow on (0,0) and is placed on square (0,2).
2. B rolls a 2, follows the arrow on (0,0) and discovers that square (0,2) is occupied. She follows the arrow on (0,2) and moves onto square (1,2)
3. C rolls a 3, follows the arrow on (0,0) and lands on square (0,3). This is a Cheese Power square. C decides to roll the dice again and rolls a 4, landing on square (4,3).
1. D rolls a 3, follows the arrow on (0,0) and lands on square (0,3). This is a Cheese Power square.He decides to explode the engines of C’s rocket, and C decides to put their rocket on square (6,0).

This shows how the game works.

Game Program

You are going to write a program which will allow two to four players play a game between themselves. They can move their rockets on their own board but the game will make all the moves for them. The players can continue to take turns until somebody reaches the end of the game. It doesn’t need to actually draw the board; the program will display messages such as:

```
Jim has thrown a 6 and moves on to square 6,3
```

The position of the player on the board should be expressed as a pair of coordinates as shown on the diagram above.

If a player must make a choice the program should ask them to enter their choice of action, for example:

```
Jim has landed on a Cheese Power Square.

Does Jim want to roll again or explode the engines of another rocket?
Enter t (throw) or e (explode):
```

You will have to design the user interaction so that the players can play the game in the manner described above. The user interface should reject invalid commands.

Assessed Coursework Submissions

You will be required to submit:

- A listing of your program which will play the game for up to four players. The program should ask how many players and accept their names. During the game it should address each player by their name when it is their turn.
- A user guide for program. This should tell a customer who has bought your game how to load and play it onto their computer. It should be no longer than two A4 pages.
- A Test Report that contains a description of the tests that you performed on the program to prove that it works. This should describe how you have tested what happens when a player lands on the different kinds of squares.

All the software and documentation submissions must be made using the eBridge system.

Program Demonstration

As part of the assessment of this work you will be required to demonstrate your solution in the Fenner Computer Suite in Week 11 of this semester. The timetable for demonstrations will distributed via email and also published on the 08101 Sharepoint site.

Ensure that you turn up in good time for your demonstration. At the appointed time you should have your program ready to run and a listing of your code displayed on the monitor screen. The demonstration will take no more than 10 minutes and will involve a test run of the game and an examination of your source code and documentation. A demonstration marking sheet is attached to this document.

Minimum Specification

The minimum specification for a program that is regarded as a working “Hyperspace Cheese Battle” game is as follows:

- Correct acceptance of number of players and their names.
- The computer should produce a random dice throw for each player and report the result of their move. If the player lands on a “Cheese Power” square the computer must ask the player whether they want to throw the dice again or explode the engines of another player.
- The program should detect when the winning player has reached the end of the game.
- The program should allow the user to begin another game when one is completed.

For each of these behaviours you should describe how you have tested them in your test report.

Testing

For the purpose of testing it should be possible to manually enter the dice throw values during a game. When your program starts it should ask the player if it is being used in test mode. When in test mode the program should ask for the value that the player has thrown:

```
Enter the dice value: 6
Jim has thrown a 6 and moves on to square 6,3
```

When you demonstrate the program a test sequence will be used that will exercise all of the behaviours of the game. This sequence will be made available to you before your demonstration.

Suggested Enhancements

Additional marks are available for enhancements to the program. Some possible enhancements are as follows, along with the extra marks that you will gain if you add them.

Note that you must get the minimum specification game working before you can add enhancements. If you do not have the minimal game working you will not get any marks for enhancements that you have added.

- Six power. If a player throws a six they have the option to throw again. If they land on a cheese power square they must use the cheese power rather than have their extra throw. If they throw three sixes in a row their rocket engine explodes they are sent back to the bottom row. [10%]
- Computer players. The game should have the option of playing with computer players who are either “Angry Allan”, “Speedy Steve” or “Clever Trevor”. Angry Allan will always expode the engines of another player when he lands on a cheese power. He picks the player he wants to swap with at random. “Speedy Steve” will always throw the dice again and “Clever Trevor” will explode the engines of the player furthest ahead, or throw again if he is in the lead. [15%]
- Board display. The game should display the board and the player positions. The board can be drawn using a character based display or graphically using XNA. **If you wish to use XNA consult with Rob Miles before beginning the XNA development.** [15%]

If you have any other ideas for enhancements these can also be added. It is suggested that you discuss these with Rob Miles before implementing them.

Note that it is not possible to allocate more than 20% for enhancements to the game.

Rob Miles November 2013

Department of Computer Science Coursework Assessment Sheet

Module Number: 08101 Title: "Hyperspace Cheese Battle" Game

Student Name: _____

Program Works Correctly – 40%

Player number selection and name storage	
Player movement	
"Cheese Power" behaviour	
Game end detection and new game start	

Evidence of Good Design – 10%

Breakdown into appropriate methods	
Effective data storage	

Appropriate Program Enhancements – 20%

"Six Power" – 10%	
Computer Players – 15%	
Board Display – 15%	
XNA implementation – 20%	

Appropriate Documentation – 10%

User Manual	

Evidence of Appropriate Testing – 10%

Test Report	

Appropriate Layout and Identifier Selection – 10%

Appropriate identifier names	
Consistent and appropriate layout	

_____

_____

Date: _____ Student: _____ Marker: _____