COMPUTER SCIENCE
ASSESSMENT DESCRIPTION 2016/17

MODULE DETAILS:

Module Number:	08024	Trimester:	2
Module Title:	Simulation and Concurrency		
Lecturer:	Darren D McKie / Warren J Viant		

COURSEWORK DETAILS:

Assessment Number:	1	of	1		
Title of Assessment:	Software Portfoliio				
Format:	Program	Report	Demonstration		
Method of Working:	Individual				
Workload Guidance:	Typically, you should expect to spend between	100	and	125	hours on this assessment
Length of Submission:	This assessment should be **no** more than: *(over-length submissions **will be** penalised as per University policy)*	2000 **words** *(excluding diagrams, appendices, references, source code)*			

PUBLICATION:

Date of issue:	w/c 20/02/2017

SUBMISSION:

ONE copy of this assessment should be handed in via:	Canvas	If Other (state method)	
Time and date for submission:	**Time** 14:00	**Date**	See Below
If **multiple hand–ins** please provide details*:*	Source Code - Tuesday 9 May 2017 Report - Monday 15 May 2017 Demonstration - Between 22 & 26 May 2017		
Will submission be scanned via TurnitinUK?	Yes	If submission is via TurnitinUK students MUST only submit Word, RTF or PDF files. Students MUST NOT submit ZIP or other archive formats. Students are reminded they can **ONLY** submit **ONE** file and must ensure they upload the correct file.	

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:
http://www2.hull.ac.uk/student/registryservices/currentstudents/usefulforms.aspx

If submission is via TurnitinUK within Canvas staff must set resubmission as standard, allowing students to resubmit their work, though only the last assessment submitted will be marked and if submitted after the coursework deadline late penalties will be applied.

MARKING:

Marking will be by:	Student Name

ASSESSMENT:

The assessment is marked out of:	100	and is worth	100	% of the module marks
N.B If multiple hand-ins please indicate the marks and percentage 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	*Show evidence of a systematic and comprehensive understanding of concurrent and distributed architectures and implementation techniques*	Program, Report, Demo
2	*Show evidence of a systematic and comprehensive understanding of real-time physically based modeling algorithms and techniques. Critically evaluate, review and amend approaches used to improve these algorithms*	Program, Report, Demo
3	*Identify, select and implement concurrent and distributed applications in C++. Evaluate, review and amend approaches used to improve these applications*	Program, Report, Demo
4	*Implement real world simulation, applying techniques from mathematics and physics, for use in virtual environments and computer games*	Program, Report, Demo
5	*Apply selected mathematical techniques of vectors, matrices and numerical integration*	Program, Report, Demo

Assessment Criteria	Contributes to Learning Outcome	Mark
Quality of Product	1,2,3,4	20
Overall PBM design and quality of implementation and selection of algorithms	2,4,5	40
Overall system architecture and quality of concurrent implementation	1,3	40

FEEDBACK

Feedback will be given via:	Verbal (via demonstration)	Feedback will be given via:	Mark Sheet
Exemption			
Feedback will be provided no later than 4 'teaching 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, which is available on Canvas - https://canvas.hull.ac.uk/courses/17835/files/folder/Student-Handbooks-and-Guides.

In particular, please be aware that:

- Your work has a 10% penalty applied if submitted up to 24 hours late
- Your work has a 10% penalty applied and is capped to 40 (50 for level 7 modules) if submitted more than 24 hours late and up to and including 7 days after the deadline
- Your work will be awarded zero if submitted more than 7 days after the published deadline.
- The over-length penalty applies to your written report (which includes bullet points, and lists of text you have disguised 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://www2.hull.ac.uk/student/studenthandbook/academic/unfairmeans.aspx) 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.

08024 ACW “Gravity Wells”

The aim of the ACW is to research, design and implement a distributed, physically-based modelling demonstrator, for a series of gravity wells.

The world consists of a large, planar surface, surrounded by a circular wall. A large number of spheres (balls) are placed on the surface.

The simulation is run across a number of peers, each displaying a consistent view of the world. The simulation allows a user to apply a force to the balls.

Definitions

- *Region of influence*: A spherical region, centred on each peer, representing the maximum distance at which forces are applied to the balls.
- *Ownership*: A ball can only be owned by a single peer. Ownership can be transferred.
- *Contended*: A ball is contended if it has forces acting on it from more than one peer i.e. it exists within the region of influence of more than one peer.

Requirements

World

The world consists of a large, planar surface upon which are placed a large number of balls (ranging from 256 – 64000 balls, defined within a configuration file). The surface is surrounded by a tall circular wall.

The balls are initially arranged in a regular grid on the surface.

Simulation

Each ball has a mass assigned which is light, medium or heavy (you should define masses in real units and indicate this via the shading of the ball). Make sure that you get your units correct. A simulation should only use meters (m), kilogram (kg), etc. Balls move under the forces applied by the peers, they are effected by a gravity force that either attracts or repels them.

One of the two types of motion described below is to be implemented for the balls in your demonstrator:

- Objects are treated as particles with dimensions as far as the physics simulation is concerned; however, friction and elasticity should be applied. Graphically, the objects rotate in such a way as to mimic real balls when in contact with other objects and after collisions.

- Objects should be modelled as rigid bodies and therefore rotation should be fully modelled within the physics equations. **(Advanced)**

You will only be required to implement and demonstrate one of these types of motion, and more marks will be available for the advanced motion.

Collision detection and response is required for all objects that come into contact, e.g. balls, surfaces, soft deformable objects, etc.

All collisions will include both a frictional force and elasticity, which will be defined within a configuration file.

Advanced: Extending the concepts described above, you will also be able to define some of the balls to be soft deformable objects which are model by a spring-dashpot system or otherwise. These balls will be scattered around the plane and clearly identifiable. These soft balls will deform on impact when they collide with other objects (i.e. balls, floor, walls etc.).

Time

Time is to be calculated per simulation loop (delta_time) and used in the physics equations. Time can be slowed by a scaling factor of the real delta_time in order to witness particular events. The time scaling factor will be global, that is the scale time will be communicated between all peers.

The simulation loop will also be able to be paused globally, that is the pause of the simulation loop will be communicated between all peers. Note, that only the simulation loop will be paused, and so other components, e.g. networking and graphics will function as normal.

These features will be very useful when debugging your PBM.

Peers

The world is distributed across a number of peers. Each peer stores only partial knowledge of the world. Each peer owns a subset of the total balls within the simulation.

A peer-to-peer network infrastructure is required with a minimum of two peers.

Three or more peers can be implemented **(Advanced)**.

As a peer moves across the surface, it broadcasts its new position to all of its peers. When peers receive these broadcast messages, they are responsible for checking this updated position against their knowledge of the world. Any balls within the region of influence of their peer are noted. If the ball is outside of their own region of interest, then ownership of the ball is transferred to their peer. If the ball is within both their region of interest and that of their peer, then the ball is flagged as being *contended*.

When a ball is *contended*, peers who do not have ownership of the ball are responsible for passing their force data to the ball's owner. The force data is the value of all the forces applied to the ball, originating from that peer (i.e. their gravity well and collisions with their balls). The owner of the ball is responsible for collating this force data, and calculating a new position for the ball. The new position is then communicated back to the interested peers.

When a peer is added to the world it has no knowledge of any balls. Balls will gradually migrate to this new peer during the course of the simulation. Load balancing algorithms could be considered to improve the efficiency of the simulation.

Visualisation

The view of the world is through a number of peers. Each peer has a set field of view covering a small proportion of the total surface. The view will be a 3d perspective projection at approximately 30 degrees to the horizontal.

The view will include:

- A graphical representation of the region of influence, i.e. colouring the surface.
- A graphical representation of the centre of influence, e.g. a small sphere.
- All balls that are within the region of influence.

- The balls will be colour coded to show ownership and any contention, and will show that the balls are rolling. You will give an indication of mass either through colouring or texturing.
- You will take care in your visualisation to ensure that balls are sufficiently large on the screen to show the accurate motion and collision detection implemented in your system. You may wish to add a zoom control to help achieve this.

Note: all the balls have the same radius and the gravity well is always positioned at a height above the plane equal to the radius of a ball.

[**Advanced**] The height of the gravity well above the surface, can be altered using the M and N keys.

Head up display

Use the AntTweakBar to display the following. Note that some of these values will be changeable using AntTweekBar so that the changes are evident in the environment after a change has been made.

- Number of balls owned by the peer
- Number of balls currently contended
- Total number of balls in the system
- Magnitude of the force being applied
- The time step of each simulation loop averaged over one second (in seconds as a decimal value)
- The time scale (default = 1.0) **[changeable]**
- Magnitude of elasticity **[changeable]**
- Magnitude of frictional forces **[changeable]**
- Target frequency of the physics (in Hz), minimum 1 Hz **[changeable]**
- Target frequency of the graphics (in Hz), minimum 1 Hz **[changeable]**
- Target frequency of the networking (in Hz), minimum 1 Hz **[changeable]**

The frequency of the graphics, networking and physics are important for the marking process. Failure to display these correctly will result in a **significant mark penalty**.

Controls

Control of the world is through a number of peers. Each peer is able to affect the balls within their region of influence, by applying a positive or negative force at the point above the surface equal to the radius of the balls where their view direction vector intersects the surface.

The control scheme uses the mouse and keyboard:

- Left button, applies an attractor force. The longer the button is held the greater the force
- Right button, applies a repellor force. The longer the button is held the greater the force
- Left-Right button together, cancels the force
- Moving the mouse, moves the position of the peer in a plane parallel to the surface.
- R to reset the system
- P will pause the simulation loop only **(globally)**
- O, L to increase/decrease the frequency of the physics, minimum 1 Hz
- I, K to increase/decrease the frequency of the graphics, minimum 1 Hz
- U, J to increase/decrease the target frequency of the networking, minimum 1 Hz
- Y,H to increase/decrease the time scale, minimum 0.001, maximum 1.0 **(globally)**
- M,N to increase/decrease the height of the gravity well above the surface
- Camera controls:

- W, S, A, D for up, down, left, right respectively
- ↑, ↓ cursor keys for zoom in and zoom out respectively

Implementation

Networking

Only the Winsock 2 library is permitted.

This application is a peer-to-peer architecture. No server of any type should be used.

The TCP or UDP network protocols can be used to transfer the data between nodes.

The minimum requirement is two peers on separate physical PCs in 177.

A network simulation tool will be inserted between the peers to simulate network latency and packet loss. Your application must be able to cope with the following worst-case network quality of service:

Latency: 100ms ± 50ms

Packet loss: 20%

The IP address and ports should be stored in a configuration file.

Alternatively, the UDP broadcast mechanism can be used to identify peers on the network.

Threads

Only C++ 11 threads are permitted.

The major components within each application are required to operate asynchronously. The choice of threading architecture is not defined.

The i7 processors in lab 177 are considered the target platform. Threading is to be used to leverage the performance of these processors.

As part of the final assessment, you will be required to run various parts of your system at different target frequencies. For example, the graphics may be requested to be run at 30 frames per second (Hz), whilst the Simulation at 1000 Hz.

This ability of individual components to run asynchronously is a major part of this assessment. Failure to implement this correctly will greatly reduce your marks.

Process Affinity

In order to ease debugging and to simplify the assessment process, you will be required to use a very specific thread to processor mapping.

Core	**Application**
1	Visualization
2	Networking
3+	Simulation

You can use any number of threads in your applications provided you have sufficient to meet the processor mapping in the table above. For example, it is acceptable to use 4 networking threads on the server, provided that they are all located on core 2.

Graphics

Only simple graphics are required, but you may choose any graphics library you wish (e.g. OpenGL or DirectX).

Simulation

You are to implement all of the physics yourself. You may use the example code that has been provided to you in the lab work and through the directed web links in the lecture material. The use of any third party physics library MUST be referenced in your code and in your report, and will greatly limit your marks for the simulation section of this module due to you not implementing it yourself.

Research questions

The following are a few points you may wish to consider:

- What is the visual impact from the perspective of a peer on a non-owned, contended ball when the owner does not update the position fast enough?
- The simulation starts with one peer owning all the balls. As new peers are added into the simulation, how are the balls migrated to improve performance?
- What happens when a ball rolls out of the region of influence? Who is responsible for updating the position of that ball?

Report

The structure of the ACW report is as follows:

- System architecture, including where threads and networking have been used (1000 words max), plus UML diagrams
- How the motion physics has been implemented for the balls, and how the collision detection and response has been implemented between the balls and other objects (1000 words max)

Marks will be lost if the word limit is exceeded.

Mark scheme

A detailed mark scheme will be provided.