DEPARTMENT OF COMPUTER SCIENCE

COURSEWORK ASSESSMENT DESCRIPTION

MODULE DETAILS:

Module Number:	08964	Semester:	2
Module Title:	Simulation and Concurrency		
Lecturer:	WJV / DPMW / DDM		

COURSEWORK DETAILS:

Coursework Assessment Number:	1	of	1		
Title of Assignment:	Galton Box Simulation				
Format:	Program	Report	Demonstration		
Method of Working:	Individual				
Workload Guidance:	Typically, you should expect to spend between	75	and	125	hours on this assessment
Length of Submission:	This assignment should be **no** more than:	1500 words			

PUBLICATION:

Date of issue:	Week 4

SUBMISSION:

ONE copy of this assignment should be handed in via:	E-Bridge	If Other (please state method)	
Time and date for submission:	**Intermediate Review** **Source Code** **Report** **Demonstration**	Week 9 during lab slot 09:30 Monday 11th May - Week 12 09:30 Monday 18th May - Week 13 Weeks 13 & 14	
If **multiple hand–ins** please provide details *(as appropriate):*			

The assignment should be handed in **no later** than the time and date shown above, unless an extension has been authorised on a *Request for an Extension for an Assessment* (Mit Circs) form which is available from the Office or http://www.student-admin.hull.ac.uk/downloads/Mitcircs.doc. The extension form, once authorised by the lecturer concerned, should be sent to Amanda Millson.

MARKING:

Marking will be by:	Student Name

BEFORE submission, each student must complete the **correct** departmental coursework cover sheet dependant upon whether the assignment is being marked by student number, student name, group number or group name. This is obtainable from the departmental student intranet at http://intra.net.dcs.hull.ac.uk/sites/home/student/ACW%20Cover%20Sheets/Forms/AllItems.aspx

ASSESSMENT:

The assignment is marked out of:	100	and is worth	100	% of the module marks

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	*Demonstrate research, selection and asessment of concurrent and distributed architectures and implementation techniques*	*Program, Report*
2	*Analyse real world problems and identify appropriate physically-based algorithms for their simulation*	*Program, Report*
3	*Implement distributed applications in C++*	*Program*
4	*Implement real world simulation, applying techniques from mathematics and physics, for use in virtual environments and computer games*	*Program*
5	*Use the mathematical techniques of vectors, matrices and numerical integration*	*Program*

Assessment Criteria	Contributes to Learning Outcome	Mark
Quality of system architecture	1	15
Quality of distribution implementation	1,3	25
Quality of completed product	1,3	10
Quality of Visualization and input/output	4	10
Performance of the collision detection algorithms	4,5	15
Performance of the collision response	4,5	10
Overall PBM design, implementation and selection of algorithms	2,4	15

FEEDBACK

Feedback will be given via:	Mark Sheet	Feedback will be given via:	N/A
Exemption (staff to explain why)			
Feedback will be provided no later than 20 working days 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/sites/home/student/default.aspx**. 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.

Assignment Details

08964 Assessment Description

Derek Wills, Warren Viant and Darren McKie

1.0 Aim

The aim of the assessment is to create a simulation of a Galton Box variation, distributed across two PC computers.

2.0 Configuration

The basic box used in this simulation is constructed from two transparent faces that sandwich a number of regularly placed pins (figure 1). Pins, diameter 2.5 cm, are placed in rows across the box, each pin separated by 10 cm. Alternate rows are displaced horizontally by 5 cm. The odd numbered rows (row 1 being the first row of pins from the top of the Galton Box) will have 10 pins, the even rows will have 9 pins. The vertical distance between rows is also 10 cm. The distance between the front and back plane is 20 cm, which is also the depth of each pin. There should be a total of 12 rows of pins.

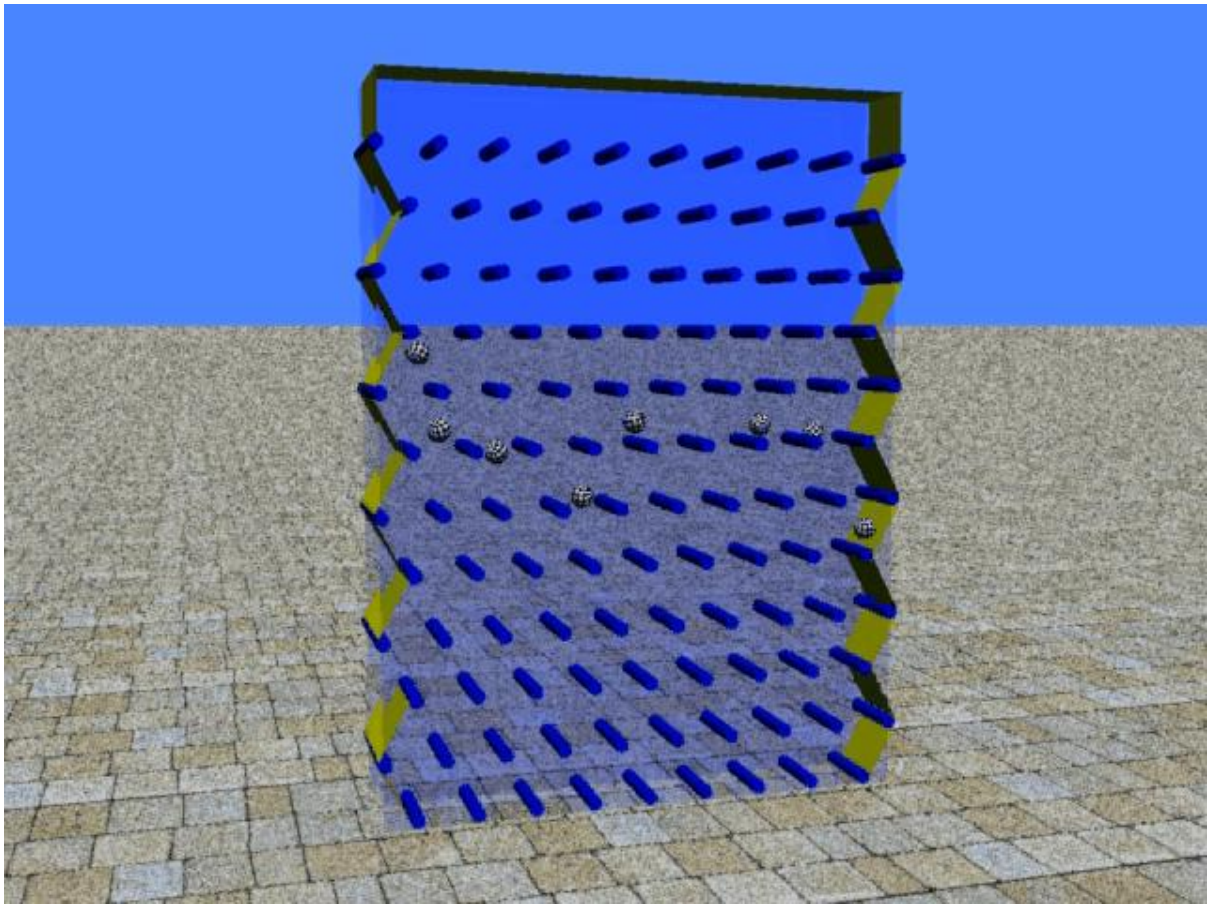

figure 1 – Galton Box example

At the top of the box is a launch area for ball to fall. Each ball may drop from any point from this top area and therefore may take a different route through the box. At the bottom of the box are a number of exit slots which align with the gaps between the bottom row of pins.

In the initial configuration of the box, a zig-zag of polygons make up the sides of the box, connecting pins along each edge. A variation to this configuration is to replace each edge polygon with a deformable net which will react to impacts from the balls as they drop through the box.

Three types of ball will be required, all with a diameter 4 cm, but made from different materials – metal, wood and rubber. Each material should have a realistic coefficient of elasticity and frictional coefficient. The mass of a rubber ball should be **m** grams, a wooden ball 3**m**/2 grams and the metal ball 3**m** grams. The mass of the rubber, wooden, and metal balls should be set from a configuration file.

Balls, when dropped, will collide with:

1. Pins: Collision detection between a pin and a ball will be needed, together with a collision response.
2. Balls: One ball may collide with another. This must be detected and dependent upon their materials, a suitable collision response calculated.
3. Front and back face: A collision detection/response is required between a ball and plane face.
4. Edge polygon: Similar to 3 but the polygons are angled. Friction should be considered.

5. Deformable nets: Collision detection between a ball and the deformable nets is required. Under impact the mesh should deform and the ball react appropriately
6. Exit slots: The exit slot taken by the ball should be recorded and for each slot the accumulation of balls passing through it should be recorded.

3.0 Implementation

The simulation should be implemented in C++ and OpenGL/gxBase.

The Galton Box should be implemented on two PCs, the top half of the box should be on PC 1 (the emitter and the top 6 rows of pins) and the bottom half on PC 2 (bottom 6 rows of pins and the exit slots). Balls should fall from one machine to the other but equally, balls should be able to bounce back onto PC 1.

The graphical representation of the box should be a full 3D perspective view which can be rotated by use of the mouse. The +/- keys should alter the rate that balls are dispatched from the top of the box, ranging from 1 to at least 100 balls falling through the system at any instant of time. Balls should be textured/coloured differently to differentiate between metal, wood and rubber. Key 1 should select all metal balls, key 2 all wooded balls, key 3 all rubber balls and key 4 should select a random selection of the three type of ball.

As balls fall through a slot they disappear, but below each slot you should indicate the number of balls that have fallen through the slot since the start of the simulation – this may either be graphical or numerical. You should also display the difference between the number of balls emitted from the top and the total number of balls that have fallen through slots at the bottom. In this way it will be clear how many balls have escaped the system due to problems with collision detection or have become stuck as they pass down the box. Key '**m**' should toggle between the flat solid edge polygons and the deformable nets.

Key '**s**' should toggle the simulation between running and paused. Key '**x**' should halt the simulation and exit. When paused, the user should still be able to rotate the scene and zoom in and out using the '**<**' and '**>**' keys.

The '**space**' bar should be used to toggle between real-time simulation and 1/10 time simulation. 1/10 time simulation, where time passes 10 time slower than normal, is included to allow for better observation of collision detection and responses within your system.

4.0 Physics

The initial implementation can assume that the balls act like particles, allowing you to ignore spin in your calculations. However, further marks will be awarded if you include spin and inertia in your final implementation. Elasticity should be included in all reactions and further marks will be awarded if friction is also included. It is suggested that a spring/dashpot approach is used for the nets at the side of the box.

5.0 Threading

The simulation should be implemented across the 4 CPU cores on the games lab PCs.

- Core 1: Graphics, UI, networking
- Core 2: Physics (including collision detection, response, and net)
- Core 3: Physics (including collision detection, response, and net)
- Core 4: Physics (including collision detection, response, and net)

Cores 2 to 4 should be load balanced to ensure equal distribution of load. To demonstrate the effectiveness of the load balancing, a graphical output of the CPU's idle time is required. The output should show the percentage of Idle time (Y axis) plotted against time (X axis), across each CPU, updated once per frame. Use the system clock to determine the ideal time for each core.

Only the Win32 threading library is permitted

6.0 Networking

The simulation is to be split across two PCs using a peer to peer network configuration. Each PC runs identical simulation software, including graphics, UI, networking and physics. Balls are released on the 1st PC, and travel through the Galton Box. Upon reaching the halfway point (as described above) the ball details are transferred to the 2nd PC where the simulation continues. As noted above, it is possible for balls to bounce back from the 2nd PC to the 1st PC.

Fault tolerance is to be provided on the network connection. If the connection is broken, the 1st PC will record the balls as leaving, but they will not be transferred to the 2nd PC. Once the connection is re-established the simulation will continue and balls will once again be transferred. Balls that failed to transfer are lost and recorded as such.

Only the winsock networking library is permitted

7.0 Report Details

Threading / Distribution

- A diagram of the system architecture, clearly showing the major processes within the implementation and how they are mapped to threads, cores and multiple PCs.
- A short explanation of the strategy employed to manage the interactions between the multi-threaded physics code and the ball database. The explanation should include a critique of the effectiveness of your approach and the lessons learnt from this implementation. [Maximum 2 pages, including diagrams]

Simulation

- A short explanation of the collision detection and response algorithm that you have employed for ball collision with a cylindrical pin. Your description should include any mathematical equations that you have employed and you should also include a critique on the effectiveness of your approach. [Maximum 2 pages, including diagrams]

Marks will be lost if you exceed page limits (see handbook for Over length penalties)