DEPARTMENT OF COMPUTER SCIENCE

ASSESSMENT DESCRIPTION 2014/15 (EXAM TESTS AND COURSEWORK)

MODULE DETAILS:

Module Number:	08240	Semester:	1
Module Title:	2D Graphics and User Interface Design		
Lecturer:	Dr David Parker		

COURSEWORK DETAILS:

Assessment Number:	1	of	1	
Title of Assessment:	The Fault in Our Trees			
Format:	Program	2nd format	3rd format	
Method of Working:	Individual			
Workload Guidance:	Typically, you should expect to spend between	50	and 80	hours on this assessment
Length of Submission:	This assessment should be **no** more than: *(over length submissions **will be** penalised as per University policy)*	N/A - coding exercise **words** *(excluding diagrams, appendices, references, code)*		

PUBLICATION:

Date of issue:	Monday 3rd November 2014

SUBMISSION:

ONE copy of this assessment should be handed in via:	E-Bridge		If Other (state method)	
Time and date for submission:	**Time**	12:00 (midday)	**Date**	Thursday 27th November 2014
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 should 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* (Mit Circs) 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. Your extension form should be submitted to the Departmental Office (RB-308).

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	50	% 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: Show evidence of key concepts and principles of computer graphics.*	Program
4	*Practical Subject Specific Skills: Select from a range of suggested approaches and techniques to implement a real-time graphics application*	Program

Assessment Criteria	Contributes to Learning Outcome	Mark
arc function for AndGate	1,4	3
Bezier curve for OrGate	1,4	3
line segment circle drawing algorithm for BasicEvent and TransferGates	1,4	5
use of your own Bezier object for drawing the curves in OrGate	1,4	10
pan and zoom functionality	1,4	5
zoom at the mouse pointer	1,4	10
complete fault tree drawn	1,4	20
good use of transformation functions	1,4	10
fit fault tree to canvas	1,4	5
smooth interpolated animation when fitting to canvas	1,4	5
good and consistent use of coding conventions	1,4	4
use of vectors	1,4	5
switch between example fault trees	1,4	5
toggle between full window (dynamic resize) and fixed size canvas	1,4	5
'minimap' view of the whole fault tree	1,4	10
usage instructions	1,4	5
Modifiable fault trees	1,4	10

FEEDBACK

Feedback will be given via:	Mark Sheet	Feedback will be given via:	Verbal (via demonstration)
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**. 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.

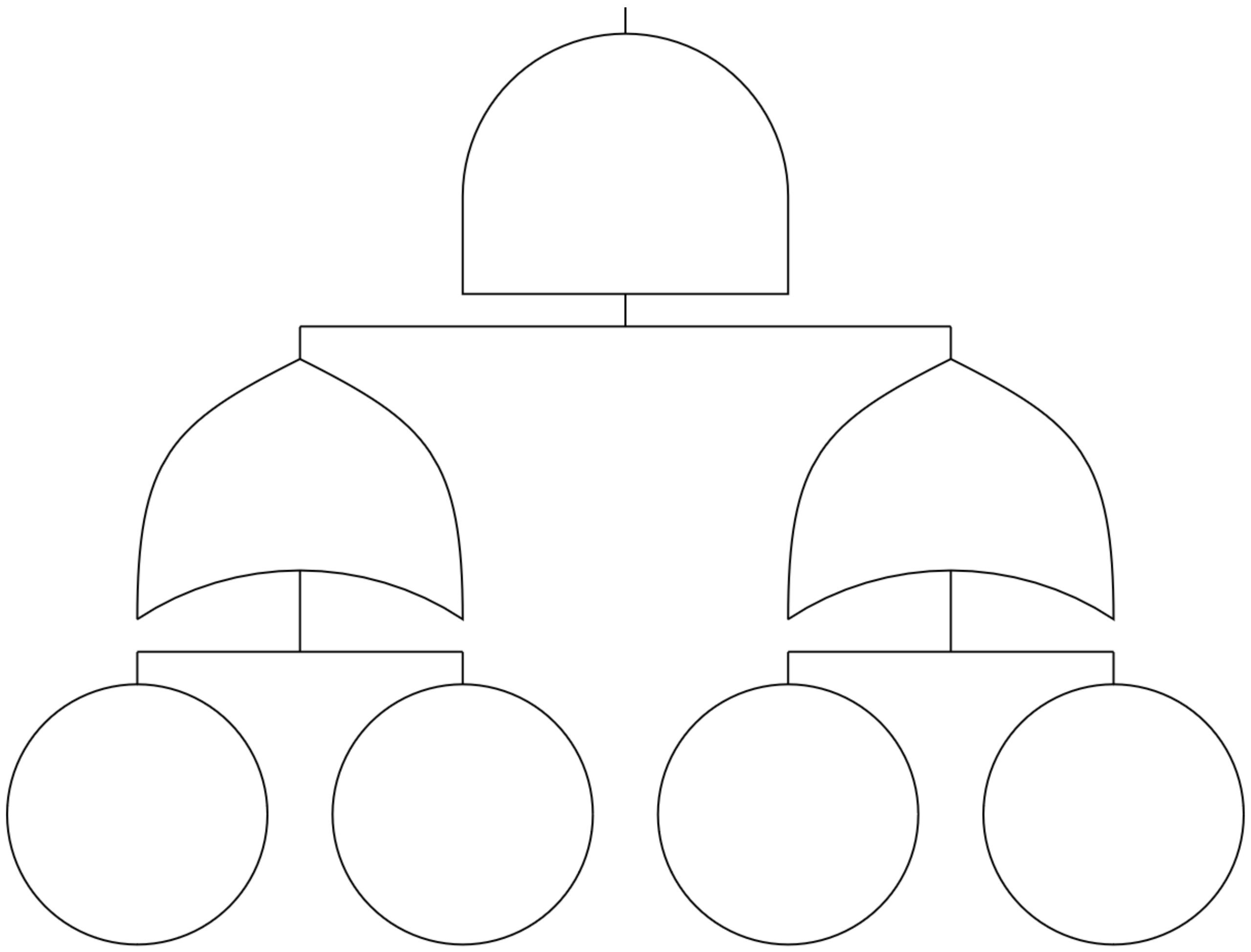

Figure 1 The above is an illustration only

Assessment

This coursework assesses the above learning outcomes.

Context

The context in which these skills will be assessed is through an Fault Tree Viewer application. The project must be completed using an HTML5 canvas element, with JavaScript providing the behaviour. The work must run in Google Chrome.

Base Class Structure

The students will be provided with a base class structure for representing a Fault Tree shown in figure 2. This will include a Fault Tree node, a Fault Tree gate, And gate, Or gate, Basic Event, and a Transfer Gate.

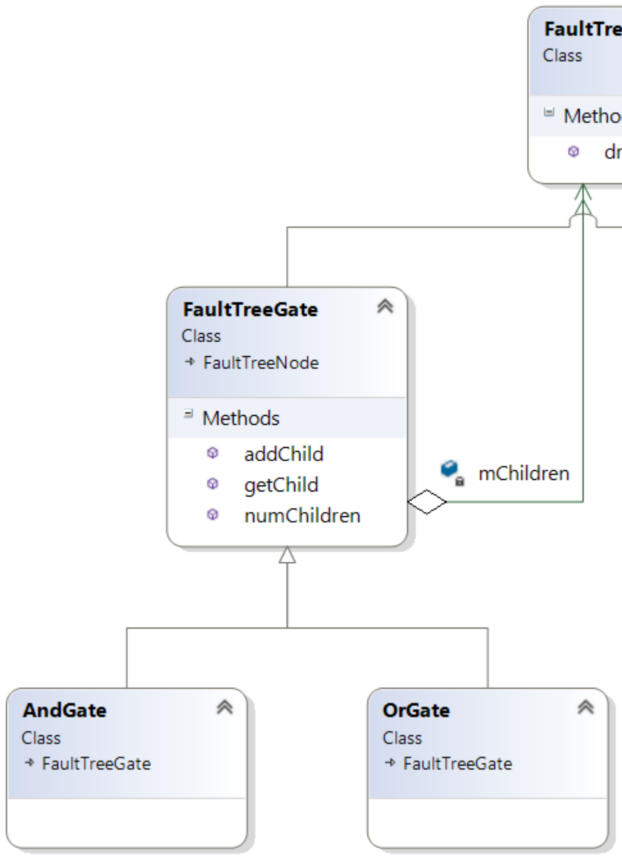

Figure 2 Fault Tree class hierarchy

In addition, they will be provided with an ExampleFaultTrees object that contain functions that will output example fault trees (of different sizes and composition). After instantiating the object, using the functions getNextFaultTree() and getPreviousFaultTree will cycle through the available fault trees returning the fault tree at the current index.

All the provided JavaScript will be missing drawing behaviour, only providing the data structure necessary to build Fault Tree objects.

Detailed Specification

This section details what is required of students to demonstrate the skills and knowledge that they should have acquired over the course of this module.

AndGate

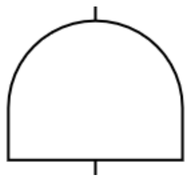

Figure 3 An And gate symbol

Figure 3 shows what an And Gate looks like. The student should add drawing code to the provided AndGate object so that they can draw one. The drawing code should make use of the moveTo, lineTo, and arc functions from the canvas 2d context.

BasicEvent

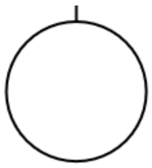

Figure 4 A Basic Event symbol

Figure 4 shows what a Basic Event looks like. The student should add drawing code to the provided BasicEvent object so that they can draw one. The drawing code should make use of the moveTo, and lineTo functions **ONLY** (**NOT** the arc function) from the canvas 2d context.

OrGate

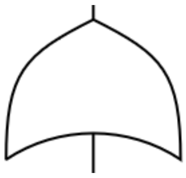

Figure 5 An Or gate symbol

Figure 5 shows what an OrGate looks like. The student should add drawing code to the provided OrGate object so that they can draw one. The drawing code should make use of the moveTo, lineTo, and bezierCurveTo functions from the canvas 2d context.

Extra marks will be available for using a 'home-grown' BezierCurve object to draw the Bezier curves.

TransferGate

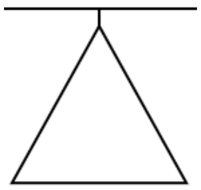

Figure 6 A Transfer gate symbol

Figure 6 shows what a Transfer Gate looks like. The student should add drawing code to the provided TransferGate object so that they can draw one. The drawing code should make use of the same polygon algorithm created for the Basic Event drawing.

Draw a complete fault tree

It should be possible to display a complete fault tree with all the nodes nicely spaced out. This should work with any arbitrarily given fault tree, not just the example ones provided.

Transformations

The positioning, orientation and size of all the objects should be controlled by the built in transformation functions (translate and scale). Careful use should also be made of the save and restore functions to control the matrix stack.

One size does not fit all

It should be possible to toggle between two modes for the canvas size. The first is a fixed size and the second is a dynamic size that is set to fill the brower window. In the second mode, the canvas should dynamically change size as the window size is adjusted.

Zoom zoom

It should be possible to zoom in to and out of the canvas using the scroll wheel of the mouse. Extra marks are available if the zoom occurs at the location of the mouse pointer.

Pan handler
It should be possible to pan (move around) the canvas using mouse click and drag at any zoom level.

Square peg in a round hole (make it fit)
It should be possible to trigger a fit fault tree to canvas action that displays the whole fault tree in the available canvas space. It should be centralised.

Animated transition
Extra marks are available if a smooth interpolated animated transition is included when the fit to canvas action is triggered.

Alternate Fault Trees
It should be possible to select between the provided example fault trees.

Minimap
It should be possible to display a 'minimap' view of the fault tree with a graphical representation of the current viewport so you can see where are on the fault tree when zoomed in.

Vector object
A Vector object should be included and should be used to represent all positions. Furthermore, calculations for where things should be placed/offset/scaled should all be done using functions that are included in the Vector object.

Documentation
You should include instructions on how to use your Fault Tree viewer including any keys or buttons. The instructions should discuss the different elements that you have in your application and how you implemented them. The instructions and descriptions should be embedded in the html page containing your canvas and should include (partial) screenshots to illustrate.

Appropriate coding conventions
Code should be properly indented. You should also make use of objects to encapsulate the functionality and data. You should be using meaningful variable and function names. Comments should be used where appropriate.

Good use of SVN
The SVN logs should provide evidence of good use of source control including meaningful log messages and regular commits as you add new functionality.

Self-assessment
Using the attached Excel spreadsheet, you should assess your own work. Each item in the mark scheme should be given a value between 0 and 10 (with 10 being the best) according to how you think you did. You should only be using a 0 if you did not attempt a particular feature. Each item has a weighting and the spreadsheet will automatically calculate your mark to give you a total out of 100. If your self-assessed mark is within 3 percentage points of my mark then the higher mark stands.

It should be noted that just because a feature exists in your viewer doesn't automatically net you the whole 10. Try to be honest and objective with your work; is it as good as it could possibly be? Could you do better? Could someone else?

Final advice: Don't target 'just a pass'. If you are aiming for 40% there is a non-zero risk that you don't score 10s for all the features you implemented. Suddenly that 40% in your self-assessment (using all 10s) has dropped below the pass threshold and you have a problem.