MARKING CRITERIA for **INB381 Assignment 2 - Semester 2, 2016**

This assignment is worth 30% of your marks for the unit. It will be marked out of 100 as follows.

Overall marking criteria

1. Well-written report (including answers to tutorial questions, screenshots for weekly practical solutions, and user instructions and screenshots for the main application program).
   Note1: You need to answer the specified tutorial questions and provide evidence that your practical code working.
   Note2: The report must be in the format of PDF or WORD. Other formats such as *.odt will not be accepted.

2. Working code with specified functionalities and good programming practice.
   Note1: Key bindings must be consistent with the specification e.g. s for forward.
   Note2: The main application program has to work and should be well documented (comments); easy to understand (good names for variables and functions); and easy to read (proper use of indention and white space).

High marks will be given for a basic graphics application that implements the specified features and which is easy to follow. You will not get any extra credit for extravagant solutions.

Grade	(1/2/3)	4 (Satisfactory)	5	6	7 (Excellent)
Written report (overall criteria)	**No or poorly written report**	**Written Report containing fair amount of information required**	**Well-written report clearly describing the tasks/functionalities**	**Well-written report clearly describing the functionality with sufficient details.**	**Well-written report clearly describing the functionality, details and how problems were solved**
1.1a Tutorials Week8-13 (2 marks per week, totalling **12 marks**)	Completed 1 or less practical and tutorials	Completed most practical and tutorial questions. Some small errors may exist.	Completed most practical and tutorial questions correctly.	Completed all practical and tutorials correctly with sufficient details.	Completed all practical and tutorial questions to excellent quality.
1.1b Statement of completeness **(1 mark)** *Note: argument for an additional feature is marked separately see 5.1*	No Statement of completeness	Simple Statement of completeness		Statement of completeness with some details	Statement of completeness clearly describing the tasks/functionality with details.
1.1c Statement of contribution **(1 mark)**	No Statement of contribution	Simple Statement of contribution			Honest and clear written Statement of contribution
Programming (overall criteria)	**Code not working or with very limited functions**	**Working code with most functionalities. Some coding issues may exist.**	**Working code with all required functionalities. Minor issues may exist.**	**Working code with all required functionalities with no known issues.**	**Working code with all required functionalities and good coding practice.**
1.2 Create graphical objects and save to file(s) **(9 marks)**	No Blender exported objects or object consists of less than 3 components	Blender exported object consists of a body and two other components	Blender exported object contains body, wings, and head.	Blender exported object appears to be a flying thingy with wings and limbs.	Blender exported object appears to be a flying thingy with bendable wings.
1.3 read the objects from file, storing information about vertices, vertex indices	No object loader or does not load object.	Working object loader	Working object loader that reads file and stores information as specified.		Working object loader with good coding practice

(9 marks)					
1.4 Continually flaps the flying thingy's wings using tree **(9 marks)**	No tree data structure used or code not working	tree data structure used for representing the flying thingy components			Elegant tree data structure used for representing the flying thingy components
1.5 Wing flapping achieved by traversing the tree, updating information and then rendering by again traversing the tree to draw the component objects. **(9 marks)**	No tree traversing code or not working	Working tree traversing code	Working tree traversing and updating code	Working tree traversing, updating and tree drawing code	Fully functional and elegant code to continuously flap the flying thingy's wings using tree
Task 2	(1/2/3)	4	5	6	7
2.1	Not Completed Task 1				Completed Task 1
2.2 steer your flying thingy using the keyboard. key 'a' for left, 'd' for right key 's' for forwards **(9 marks)**	Code not working	All three keys function correctly most times	All three keys function correctly all the time	All three keys function correctly with good coding practice	All three keys function correctly with best coding practice
2.3 Your flying thingy should move at a constant speed in the direction of its motion (no flying backward) and never escape the world view. **(6 marks)**	Code not working or flying thingy not moving	Flying thingy moves forward/backward	Flying thingy moves forward only at a constant speed but may escape from the view.	Flying thingy moves forward only at a constant speed and never escape from the view.	Fully functional as specified and best coding practice
Task3	(1/2/3)	4	5	6	7
3.1	Not Completed Task 1 & 2				Completed Task 1 & 2
3.2 Add material to the objects, a single point light and Phong model **(3 marks)**	No shading implemented or shading not working.	Lighting model implemented somehow and working.	Material and light source all specified properly. Lighting model works.	Phong model implemented correctly.	Phong model implemented correctly with good coding practice.
3.3 Add functionality so that when the 'z' key is hit, keyboard control stops working and the flying thingy falls straight down. **(3 marks)**	'z' key not working.	'z' key works and flying thingy falls somehow.		'z' key works as specified.	'z' key works as specified and good coding practice
3.4 Add a ground to your scene **(3 marks)**	No visible ground in the scene.	Visible ground in the scene.	Visible ground is colored.	Realistic ground in the scene e.g. grass, road etc.	Realistic ground in the scene and good coding practice
3.5 When crash-landed, resume keyboard control. **(1 mark)**	Not working				Working as specified.
Task 4	(1/2/3)	4	5	6	7

4.1	Not completed Tasks 1,2 & 3				Completed Tasks 1,2 & 3 with high quality
4.2 Add functionality so that when the 'x' key is hit, the keyboard control stops working and the flying thingy spirals towards the ground, always oriented in the direction of its motion. **(3 marks)**	'x' key not working	'x' key works and flying thingy gets to the ground somehow	'x' key works and flying thingy spirals to the ground	'x' key works as specified.	'x' key works as specified and good coding practice
4.3 The spiral path needs to have a wide radius at the top which reduces as the flying thingy continues to fly down until it reaches the centre of the bottom of the viewport. **(3 marks)**	Not working			Working as specified	Working as specified with good coding practice.
4.4 The flying thingy should accelerate all the way down to your ground, then decelerate quickly over a short straight-line distance parallel to the ground plane before it comes to a complete stop. **(3 marks)**	Not working	Flying thingy accelerates to the ground.	Flying thingy accelerates to the ground then stops somehow.	Working as specified	Working as specified with good coding practice.
4.5 When landed, keyboard control should be resumed. **(1 mark)**	Not working				Working as specified
Task 5	(1/2/3)	4	5	6	7
5.1 Describe an additional feature and argue for it's inclusion in your Statement of Completion. **(5 marks)**	No Inclusion statement or specified feature	Inclusion statement specifies a feature already covered in the prior tasks.	Inclusion statement specifies a feature already covered in the prior tasks and the argument for use is debatable.	Inclusion statement specifies a new feature backed with a good argument.	Inclusion statement specifies a new feature backed with a well supported argument.
5.2. Implement your feature. **(10 marks)**	Feature not implemented	Feature implemented and functional with some issues.	Feature implemented and fully functional with minor issue.	Feature implemented and fully functional with no known issue.	Feature implemented and fully functional with good coding practice.