University of Central Lancashire UCLan Cyprus **School of Sciences**	**UCLan Coursework Assessment Brief**		2022-2023
	Module Title: Games Concepts Module Code: CO1301		Level 4
	Assignment 1: Spheres on a Board	This assessment is worth 45% of the overall module mark	

THE BRIEF/INSTRUCTIONS
You must follow the specification to implement a simple 3D game inspired by Katamari Damacy and the popular "io" games. You are expected to complete this in your own time outside of scheduled labs. But you should ask the module leader or your lab tutor for advice if you are stuck. You **MUST USE TL-ENGINE** to implement this game. The specification for implementing this game can be found at the end of this document, and a template for producing your report can be found on Blackboard. However, **PLEASE READ THIS BRIEF IN FULL** for full guidance on completing and submitting this assessment. Do not diverge from the specification and instructions provided in this brief. If you do, then you will lose marks. Ask for clarification if you are unsure!

LEARNING OUTCOMES ASSESSED
i. Describe key games concepts, e.g. genres, terminology.
ii. Use a game engine to create simple computer game prototypes.
iii. Apply mathematical techniques for analysis and reasoning about problems

CODING STYLE
i. Your code MUST be properly indented and laid out so that it is readable.
ii. Your code should be commented appropriately, over-long lines of code or comments should be split up
iii. Variable names must be meaningful.
iv. You should have no "magic numbers" but instead make proper use of variables and constants.
v. You should pay attention to the correct and appropriate use of data types.

REQUIREMENTS FOR HIGH MARKS
i. You should make use of enumerated types to control the states within the game where appropriate.
ii. Utilise functions for collision detection and vector calculations, implement functions outside the main program as appropriate.
iii. Utilise arrays and loops where appropriate.
iv. Pay attention to the playability of the game as well as achieving the technical requirements.

PLAGIARISM
i. This is an individual project and no group work is permitted.
ii. You will be held responsible if someone copies your work - unless you can demonstrate that you have taken reasonable precautions against copying.
iii. Remember the machines in the Games Lab have shared drives. You must either password protect your work or else remove it from the hard drive each time you finish working on it. You may also consider using a change management system like GitHub.

SUBMISSION
Your final submission should be a zipped file named Assessment1_[FirstnameInitial][Lastname] that contains the following:
i. The .cpp file of your project with your name added as a comment at the top of the file.
ii. The media folder containing all your models.
iii. A report created using the provided report template.

Additional information about submitting this assessment can be found in the "Submission Details" section of this brief.

PREPARATION FOR THE ASSESSMENT
- Lectures will be used to introduce theoretical concepts needed to complete this assessment.
- Labs will be used to practice prototyping games similar to this assessment.
- Extra worksheets and solutions will be provided for students to practice prototyping games outside of labs.

RELEASE DATE AND HAND-IN DEADLINE

Assessment Release date: 21/10/2022 Assessment Deadline Date and time**: 13/01/2023 22:00**

Please note that this is the final time you can submit **– not the time to submit!**
Your feedback/feedforward and mark for this assessment will be provided by 03/02/2023

SUBMISSION DETAILS

Submission is electronic, submission links are available on the Assessments section of the module's Blackboard page. You should submit a single zipped folder named "Assessment1_[Firstname Initial][Lastname]" (for example if I were to submit, my zipped folder will be named Assessment1_LNisiotis). The zipped folder should include the following:

i. The .cpp file of your project with your name added as a comment at the top of the file.
ii. The media folder containing all your models.
iii. A report created using the provided report template.

Note: Please do not include your whole project folder. If your game fails to compile when tested by the marker, your mark will be zero.

HELP AND SUPPORT

- Academic support will be provided during the labs and the tutor's office hours. Please get in touch with your tutor in class or send an email with your query.
- For support with using library resources, please contact CyprusLibrary@uclan.ac.uk. You will find links to lots of useful resources in the My Library tab on Blackboard.
- If you have not yet made the university aware of any disability, specific learning difficulty, long-term health or mental health condition, please contact the Student Support CyprusStudentSupport@uclancyprus.ac.cy.
- To access mental health and wellbeing support, please contact Student Support at CyprusStudentSupport@uclancyprus.ac.cy.
- If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online prior to the deadline. For this, contact the School Office Admin: SchoolsAdmin@uclancyprus.ac.cy

Disclaimer: The information provided in this assessment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and a new version of this assessment brief will be circulated.	Version: 1.1

SPHERES.IO GAME SPECIFICATION

You must follow this specification to implement a simple 3D game inspired by the popular “io” games. Ask the module leader or your lab tutor for advice if you are stuck or unsure. You should implement the features described below in order. To be eligible for a mark within any classification, you must have completed all the features for all the previous classifications.

MEDIA RESOURCES

This assessment has an associated set of media files which can be found inside "Assessment1Resources.zip" on Blackboard. The model meshes included in the folder are:

spheremesh.x	A sphere	Radius = 10 units
water.x	Water	
island.x	The ground	200 x 10 x 200 units
sky.x	A skybox	
minicube.x	A cube	5 x 5 x 5 units

Please remember that all of the models and textures I supply are under license and must not be passed on to any else or used for any purpose other than your University work.

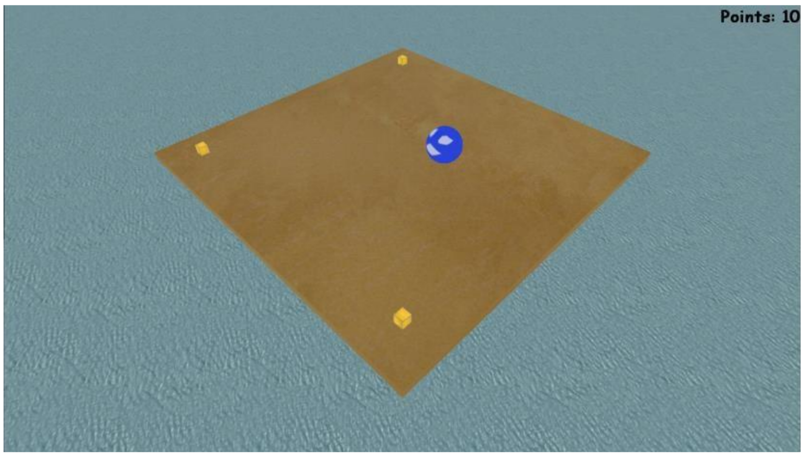
BASIC LAYOUT

Create a new TL-Engine project named Assessment1 and using the resources provided for this assignment, set up your scene as specified below before moving on to meeting the specified milestones.

a. Create a model using water.x and position it at (0, -5, 0).
b. Create an island using island.x and place it at (0, -5, 0), it should appear to be an island, its bottom half-submerged into the water, and the ground of its top half at 0 along the y-axis. This will enable you to position objects on the island more naturally. This will be the stage for your game.
c. Create a sphere model using spheremesh.x and position it at 0 along the x and z-axis, and high enough along the y axis so that it appears to sit on the floor (hint: the radius of the sphere is 10 units, and its origin is at its centre). This will serve as the player-controlled game object.
d. Create four cube models (cube1 at x = -80 and z = 80; cube2 at x = 80 and z = 80; cube3 at x = 80 and z = -80; and cube4 at x = -80 and z = -80) using minicube.x, and make sure they are positioned appropriately along the y axis so that they appear to sit on the floor. All of the boxes must be axis-aligned (i.e. their edges are parallel with the axis).
e. Create a skybox using sky.x and position it at (0, -960, 0).
f. Add a top-down camera by creating a kManual camera at (0, 200, 0) and rotate it by 90 degrees about its local x-axis.

1. **MILESTONE 1: THIRD CLASS (40%+)**
   a. Set up the scene exactly as described in the previous section.
   b. Implement keyboard control for the sphere using WASD keys:
      i. 'W' moves the sphere forward
      ii. 'S' moves the sphere backwards
      iii. 'D' rotates the sphere clockwise
      iv. 'A' rotates the sphere anti-clockwise
      v. The game should enter GameOver state if the player moves into the water, and then stop.
   c. Use kSphereSpeed and kRotationSpeed as constants for storing the speed at which you move and rotate the sphere. Choose a reasonable value that works well in the lab and on your machine.
   d. Implement movement for your top-down camera by adding keyboard keys:
      i. 'Up' moves the camera forward along the world z-axis
      ii. 'Down' moves the camera backwards along the world z-axis
      iii. 'Right' moves the camera right along the world x-axis
      iv. 'Left' moves the camera left along the world x-axis
   e. The P key causes the game to enter a Paused state. In the Paused state:
      i. Nothing should move
      ii. Hitting the P key causes the game to transition to the Playing state
   f. In all game states, the player should be able to press the Escape key to quit the game.
   g. The sphere should pick up cubes that are very close to it:
      i. Calculate the distance between the sphere and all cubes.
      ii. If the distance is not greater than the sum of the sphere's radius and half the width of the cube, the cube should be picked up by the sphere.
      iii. For example, if the sphere's radius is 10 units, then any cube within (10 + 5/2) 12.5 units of sphere should be picked up.
      iv. Picked up cubes should not be visible (think of a way to make the cube disappear).
   h. The code you submit MUST compile without errors.

2. **MILESTONE 2: LOWER SECOND CLASS (50%+)**
   a. Setup an isometric camera that can be activated by pressing the “2” key.
      i. The camera should be positioned at (150, 150, -150), rotated by -45 degrees about its local y-axis **and then** by 45 degrees about its local x-axis (the order is important, why?).
      ii. If the isometric camera is active, pressing the “1” key should reactivate the top-down camera.
   b. Reward the player with points whenever the sphere picks up a cube
      i. Create an integer variable named “playerPoints” for tracking points accumulated
      ii. When the sphere "picks” up a cube, add 10 to the player's points.
      iii. Display the players points at the top and aligned to the right of the game window in the format “Points: X” e.g. after picking up all four cubes, “Points: 40” should be displayed
   c. When 4 of the cubes are picked up, the sphere should grow in size.
      i. Implement this by scaling the sphere to 1.2 its initial size, this will make the radius of the sphere 12 units.
      ii. After scaling up, the y position of the sphere must be changed so that part of it does not appear below the floor (hint: consider the sphere’s new radius).



3. **MILESTONE 3: UPPER SECOND CLASS (60%+)**

a. Rather than loading 4 cube models manually, use an array of IModels to create and store 12 cube models.

   i. Using a loop and a random number generating function, create and place all 12 cubes at random positions on the floor.

   ii. Consider using C++'s rand() function.

   iii. Does the scene look the same every time the game is executed? That is because rand() is a pseudo-random generator.

   Note: checking for collisions between the sphere and 12 cubes will lead to a reduced FPS rate, therefore it is recommended that you utilise variable timing when moving your sphere from this point on.

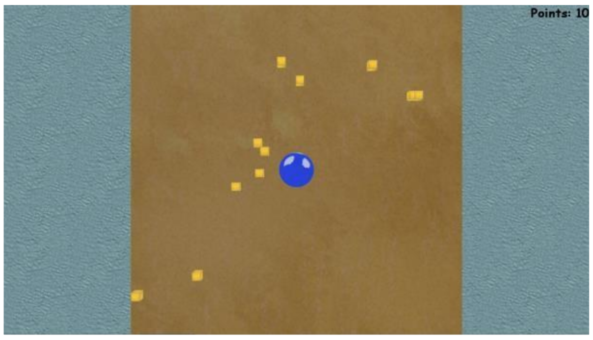
b. Update your cube pickup code to use sphere-to-box collision detection to check for contact between the sphere and the cubes. Remember all the cubes are axis-aligned. Also remember that for every 4 cubes picked up, the sphere should be scaled to 1.2 times its size.

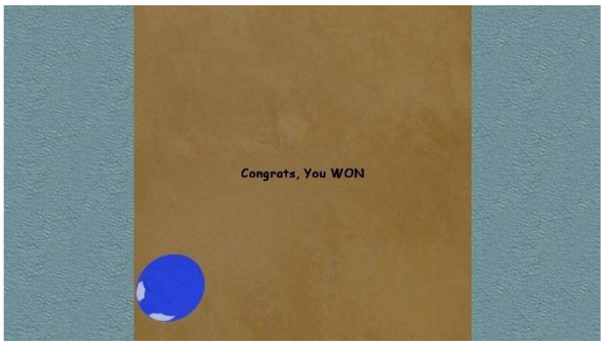
c. When a player accumulates 120 points, the game should enter a GameWon state. In the GameWon state:

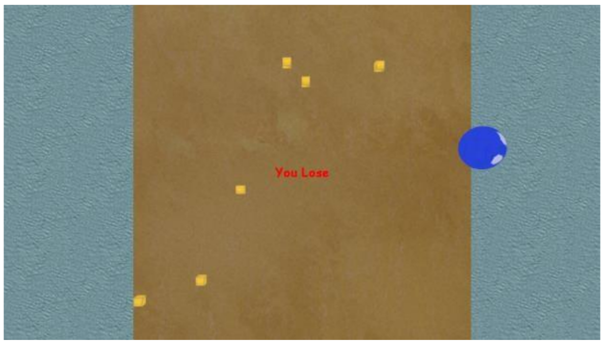
   i. The sphere should not be able to move.

   ii. A congratulatory message should be displayed at the centre of the game window.

d. The game should now have 4 states: Playing, Paused, GameWon and GameOver. You may also want to display an appropriate message in the GameOver state, while keeping the player fixed, instead of just stoping the game.







4. MILESTONE 4: FIRST CLASS (70%+)

a. When randomly spawning cubes, check to ensure that each cube is not within 10 units of other cubes or in collision with the sphere. As long as (hint: while loop) the cube being spawned is within 10 units of another cube or in collision with the sphere, a new random position should be generated for the cube.

b. Instead of hiding picked up cubes, respawn them at a new random position and ensure that the checks in ‘a’ above are performed. Consider utilising a single function for spawning and respawning cubes.

c. Implement a “Hyper Mode” in which the sphere attracts cubes within a 50 unit radius:

   iii. Add a special cube using “minicube.x” and “hypercube.jpg” as its skin at a random position as in ‘a’ above.

   iv. On picking up the special cube, the sphere should transition into a “Hyper” state and its skin should change to “hypersphere.jpg”. The Hyper state should last for 5 seconds, after which the sphere should return to its “Regular” state.

   v. In the Hyper state, the sphere should attract cubes that are within 50 units from its centre “magnetically”, attracted cubes should “fly” into the sphere before they are respawned.

5. **MILESTONE 5: HIGH FIRST(up to 100%)**

a. Introduce an NPC to the game:
   i. Create an enemy sphere model and set its skin to enemysphere.jpg.
   ii. Choose a random position far away from the player sphere to position the enemy sphere.
   iii. Implement automatic movement for the enemy sphere, and set it up so that it can pick up cubes in the same manner as the player sphere. At a minimum, implement this so that the sphere appears to make random movement decisions e.g. whether to move up or down, left or right. Full marks will be awarded for implementing "intelligent" behaviour.
   iv. Declare an integer variable called enemyPoints to keep track of the enemy sphere's points.
   v. If the enemy sphere collides with the player sphere, they should bounce off each other if the difference between their points is not greater than 40, otherwise, the sphere with the higher points should pick up the other sphere, earn 40 points and increase in size by a factor of 1.2.

b. If the enemy sphere accumulates 120 points before the player sphere, or If the player sphere is picked up by the enemy sphere, then the game should enter a GameOver state.
   i. In the GameOver state, a game over message should be displayed on the centre of the screen, however, the enemy sphere should continue to move around as usual on the floor.

c. List the points of the player and NPC, highest point first, and aligned to the right of the screen.

Marks Breakdown		
Feature	**Marks**	**Total**
1a. Basic Scene	20	49
1b. Sphere Movement	6	
1c. Camera Movement	6	
1d. Pause	5	
1e. Quit	2	
1f. Box Pickup	10	
2a. Update Camera	3	12
2b. Points	3	
2c. Scale Sphere	2	
Report - State Transition Diagram	4	
3a. Cube Model Array	3	10
3b. Sphere to Box	4	
3c. GameWon State	3	
4a & 4b. Random (Re)Spawn	8	15
4c. Hyper Mode	7	
5a & 5b.NPC	9	14
5c. Points Ranking	5	