CO1301 Games Concepts – Portfolio Assessment
Game 1 – Balls on a Board

You must implement a simple 3D game scene containing moving objects controlled by the keyboard according to the specification overleaf. You are expected to complete this in your own time outside of scheduled labs. But you should ask your lab tutor for advice if you are stuck.

Do not diverge from this specification. If you do, then you will lose marks. Ask for clarification if you are unsure!

Your final submission should include the .cpp file of your project, the media folder containing all your models, and a word document containing your state transition diagram if you are aiming for marks higher than 60%.

In addition to the above, your formative feedback point submission should include a video (not more than 2 minutes) showing your game in execution.

The associated set of model files for this game can be found on Blackboard inside "Game1_Models.zip". The models include a ball (ball.x), a floor (floor.x) and a board (board.x).

Game Specification

You should aim to implement the features described below in order. To be eligible for a mark within any classification, you must have attempted all the features for all the previous classifications. Treat the grade boundaries as milestones in a real game development project.

Basic Scene Layout

To achieve any classification, you must first implement the basic scene layout and then meet the requirements of the classification. The basic scene layout is specified below:

- Create a new TL-Engine Project called "BallsBoard"
- Create a folder called "media" inside your visual studio project folder and copy the supplied models into it.
- In your code use relative addressing to access the media files – i.e. myEngine->AddMediaFolder("./media");
- Using "floor.x" for the mesh, create a floor model at position (0, -0.2, 0).
- Then use the "board.x" mesh to create a board model and position it at the origin so that it appears placed on the floor. The board is checkered with squares of 50 x 50 units.
- Create a manual camera at position (0, 240, -240), and rotate it about the x-axis by 45 units, such that it is suspended in the air and looking downwards towards the ground.
- Using "ball.x" mesh, create sphere1 at position (125, 10, -125), and sphere2 at (-125, 10, 75). The base of each sphere should be touching the floor.

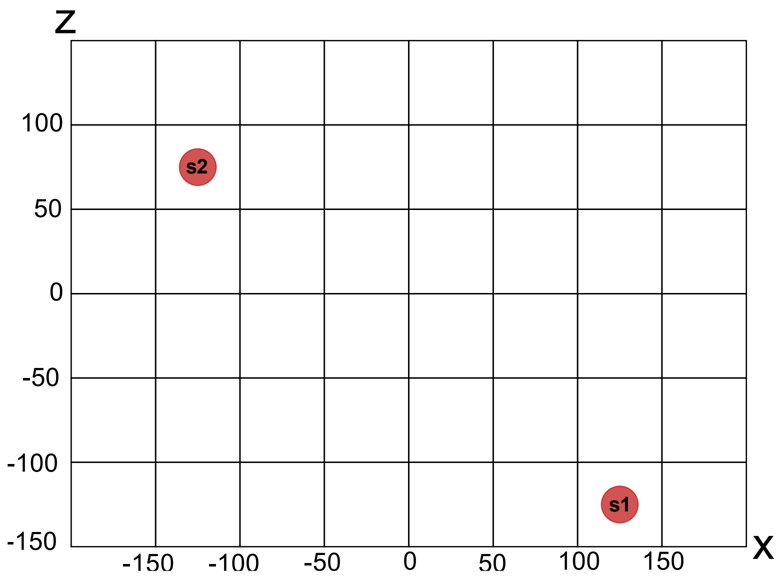

Figure 1 Initial positions of sphere1 and sphere2 (shown in 2D)

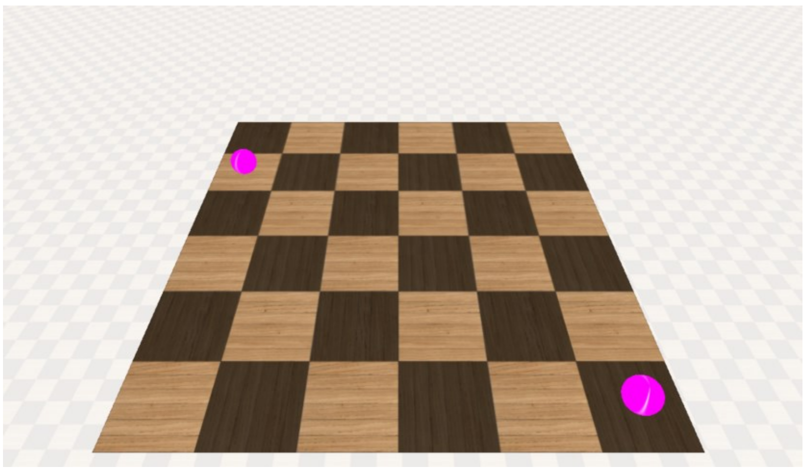

Figure 2 Screenshot of the basic scene layout

Pass Mark/ Third Classification (40% +) Milestone 1

- The code that you submit must compile without errors.
- Your scene is set up correctly as described above.
- The spheres should both slide diagonally forwards and backwards thus:
    - sphere1 should move between its starting point of (125, 10, -125) and position (-75, 10, 75).
    - sphere2 should move between its initial position (-125, 10, 75) and position (75, 10, -125)
- Declare a state variable for each sphere to determine its current direction of travel.
- The speed at which the spheres move must be defined in your code by a constant float named **kGameSpeed**.
- As the game starts, the sphere1 should start moving away from the camera, and sphere2 should start moving towards the camera.
- The player should be able to hit the Escape key to quit the game.

Lower Second Classification (50% +) Milestone 2

- Add a single float variable called **sphereSpeed** to adjust the speed of both spheres.
    - Hint: You should use this variable to multiply the constant **kGameSpeed** to determine how far to move a sphere each frame.
- The variable should have an initial value of 1.
- The player should be able to increase or decrease the speed of both spheres by 1/10th of the value of **kGameSpeed** each time they scroll the mouse wheel up or down respectively.
- Set limits for the maximum and minimum speed possible: minimum speed is half the value of **kGameSpeed**, and maximum speed is five times the value of kGameSpeed.
- The spheres should take turns in moving so that sphere1 completes a single diagonal trip, then stops and wait for sphere2 to complete a single diagonal trip, and so on…
- Use a state variable to determine which sphere should be moving.
- The player should be able to hit the P key as a toggle to pause and resume the movement of both spheres.

Upper Second Classification (60% +) Milestone 3 – breakeven point achieved

- Using the resources supplied, change the skin of sphere2 to blue.
- Expand the possible values of your spheres’ state variables so that each sphere can now move along all four diagonals.
- Rather than moving forwards and backwards along the same diagonal, the spheres should move clockwise along a square-shaped path (sides of 200 units each) as shown in the diagrams below.

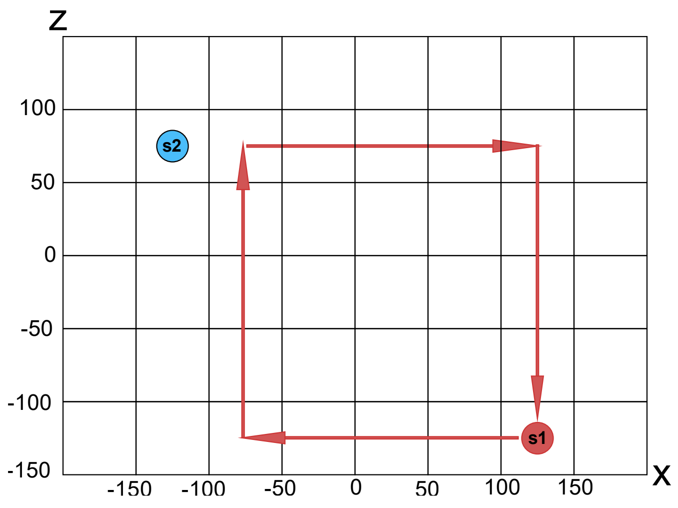

Figure 3 Sphere1's movement path

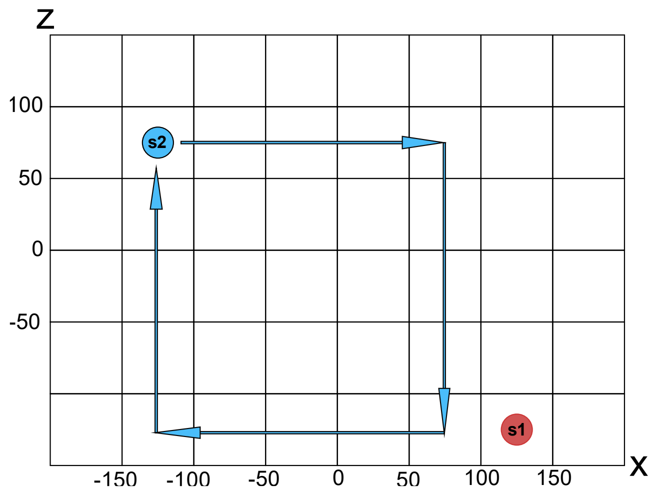

Figure 4 Sphere2's movement path

- Only one sphere should be moving at a time, meaning the spheres take turns to move around their square-shaped paths.
- Each moving sphere should appear to roll along in the correct direction, by rotating about the appropriate axis.
- Create a state transition diagram to show the behaviour of your spheres.

First Classification (70% +) Milestone 4 – bonus payments received

- Add another two spheres to the scene:
    - sphere3 at position (-125,10, -125)
    - sphere4 at position (125, 10, 75)
- Use the indigo and green skins for the new spheres.
- sphere4 should move along the same path that sphere1 moves, while sphere3 moves along the same path that sphere2 moves.
- Therefore when the game starts, spheres 1 and 4 move along their square-shaped path to complete a single trip while spheres 2 and 3 wait, once spheres 1 and 4 complete their trip then spheres 2 and 3 move along their square-shaped path to complete a single trip while spheres 1 and 4 wait, and so on until the game is quitted.
- When the player hits the up-arrow key, all the spheres should be raised 30 units slowly in the y-axis. This should only work if the spheres are touching the floor.
- While the spheres are raised 30 units above their initial position, the player should be able to hit the down-arrow key so that they are slowly lowered till they touch the floor.
- Hitting the up or down arrow while the spheres are in the process of being raised or lowered should not perform any action.
- It should not be possible to raise the spheres higher than 30 units above their initial positions or lower them below the point where their bottoms touch the floor.
- Make sure that as each sphere rolls along the ground, it rotates in the correct direction.
- Spheres should not roll if they are raised, they should only roll when they are on the floor.
- Update the state transition diagram to show the updated behaviour of your spheres.

High First Classification Milestone 4 – extra bonus payments received

There are some final marks available but only if you have completed the first classification:

- When the player hits the E key, cycle the colours of the spheres around in a clockwise direction (i.e. each sphere takes the skin of the sphere that was next to it when the game started).