CO1301: Games Concepts
Lab 15 - Timing

1. Introduction

1. This lab introduces Tl-Engine's Timer() function and the use of variable time.

2. Setting Up

1. Create a new TL-Engine project called "Lab15_Timing_Project".
2. Download the zipped media folder from blackboard and unzip its content into your working directory.
3. Load and create a model using "stars.x". This will be your skybox.
4. Load the mesh file: "particle.x", and create a particle model.
5. Create a 'kManual' camera and position it at (0, 0, -60).
6. You need to call the Timer() method in order to establish the first base time. The command needs to be placed outside the game loop, but make sure to place it immediately before the game loop. Call it using the following code:

   ```
   myEngine->Timer();
   ```

7. You then need to call Timer() once per frame, at the beginning of the game loop, immediately after:

   ```
   myEngine->DrawScene();
   ```

8. Declare a variable of type float called frameTime. Use Timer() to get the time taken for the last scene to be rendered:

   ```
   float frameTime = myEngine->Timer();
   ```

9. Use the draw command to output the frame time and FPS (frames per second can be caluclated by 1/frameTime) to the screen.

3. Adjusting Movement by Time

1. Move particle back and forth across the screen from -10 to 10. Use the frame time to adjust the speed of the movement by multiplying the distance moved by the frame time. The speed of movement should be 10 units. This is a lot higher than you are used to using. However, you are now using values adjusted according to the frame time

   ```
   particle->MoveX(kSpeed * frameTime);
   ```

2. Set up the W, A, S, D keys to move the camera. Use the frame time to adjust camera movement rate in the same way that you used it to adjust the movement of the particle.
3. If you are using mouse movement to rotate the camera then you do not use the frame time to adjust rotation caused by the movement of the mouse. The movement of the mouse is independent of the frame time and so does not need adjustment

   In this way you will build a camera whose movement is independent of the speed of the computer

4. Slowing the Frame Rate

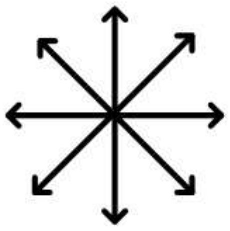
1. In this section, you will reduce the fps significantly to demonstrate that the velocity of movement remains the same even when the fps changes radically.
2. Load the cube mesh.
3. Create a wall of cubes. The wall should be 100 cubes by 100 cubes, i.e. 10,000 cubes in total

   The size of a cube is 10 by 10. Use an array of type IModel, with a size of 10000. Create the wall of cubes outside the game loop. You will need to use a nested loop to make the wall: the outside loop for columns of the wall and the inside loop for the rows of the wall.

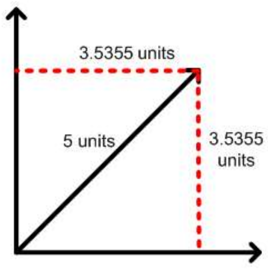
If you cannot see how to do this then you can get the code here. **However, I would strongly suggest trying it out yourself first.**

5. Advanced: Pulse

1. Create a new TL-Enegine project called Lab15_TimedPulse_Project. Download the zipped media folder from blackboard and unzip its content into your working directory.
2. Add a stars skybox, a camera at (0, 0, -60), establish the first base time before the game loop and call Timer() once per frame and assign its value to `float frameTime` as you did in Lab15_Timing_Project.
3. Load the particle.x mesh and create an array of 8 particle models.
4. Each particles should move out from the origin in one of the cardinal directions at a velocity of 5 units per second;



5. The movement of the particles going vertically or horizontally is straightforward since they are moving along a single axis.
6. Those particles which are moving along the diagonal require movement along both x and y axes. The velocity of movement along these axes can be found by using Pythagoras' Theorem to calulcate the length of the two equal sides of a right angle traingle with a hypotenuse of length 5.



   The x and y components will each be 3.5355 units respectively. **Note that the sign of the velocity or the component will change depending on the direction of movement.**

7. Declare a timePassed variable. Start the particles moving out, each in their own direction. Use timePassed to store how much time has passed since the particles first started moving out. After 3 seconds have passed place all of the particles back at the centre point. Reset timePassed to 0, the particles then continue moving out as before, the timePassed variable is increased and so on
8. The effect of doing this is that the particles will pulse out from a centre point once every 3 seconds.