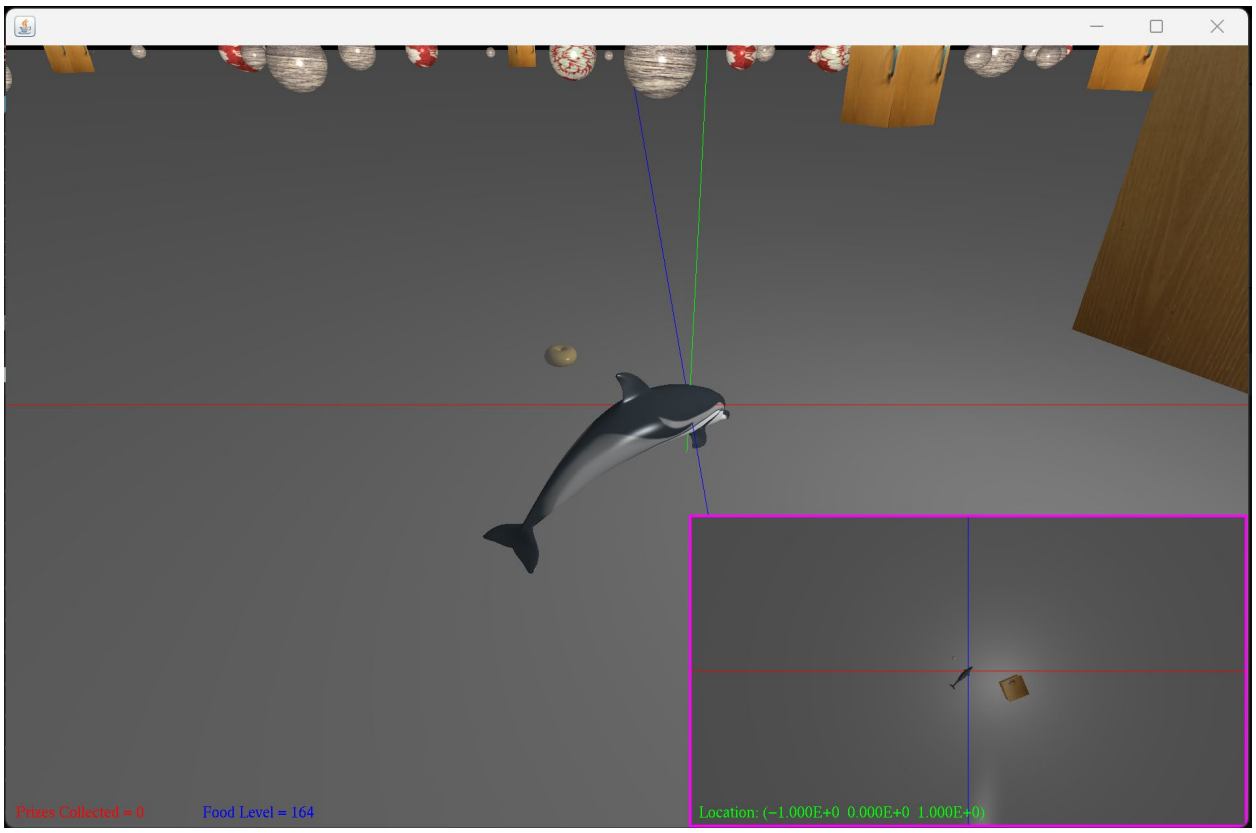
1. Eduardo Guzman, CSC-165 Section 2, A2 – Dolphin Adventure 2

2. 

3. **How to play:**

    a. The game consists of one main character/avatar: the dolphin. You must move the dolphin throughout the Gameworld and collect prizes to win. To "win", the player must collect a certain number of prizes (can be set in the code and will also be described in the "Scoring" section). The prizes are in the shape of spheres, so we can refer to them as prize orbs. To collect a prize orb, the avatar must collide with the prize orb.

    b. There is an additional restriction: the game avatar has a food level and to collect prizes, the food level must be above a certain threshold (also explained in "Scoring"). Since a player can become hungry, there are also Food Stations scattered throughout the world that look like drawers and act as a food source for your avatar. To collect food, the avatar makes contact with the station. The food is not consumed right away; it is stored in a food storage buffer and the player must press a specific key in order to eat food.

    c. The Torus shape will represent food and the dolphin will have a child game object that rotates around it when the food storage buffer contains food. If the buffer is empty, the food torus will disappear from view.

    d. A ground plane is now implemented and the dolphin can only move above the plane.

Scoring:

    e. There are two scores to keep track of:

        i. **Prizes:** You will collect prize orbs and when you collect *30*, you win.

            1. The winning prize number can be changed in the code to enhance/lessen the difficulty.

ii. **Food Level:** You must try to stay above a food level of *50* to collect prizes and win. The food level starts at *200*.

4. **Inputs for Moving the Dolphin:**
    a. **"A", "Left Joystick"**: The "A" key on a keyboard, and right motion with the "Left Joystick**"** on a gamepad, simulates a left turn for the dolphin.
    b. **"D", "Left Joystick":** The "D" key on a keyboard, and right motion with the "Left Joystick**"** on a gamepad, simulates a right turn for the dolphin.
    c. **"W", "Left Joystick":** The "W" key on a keyboard, and upwards motion with the "Left Joystick**"** on a gamepad, simulates a forward movement for the dolphin.
    d. **"S", "Left Joystick":** The "S" key on a keyboard, and downwards motion with the "Left Joystick**"** on a gamepad, simulate a backwards movement for the dolphin.
    e. **"E", "B":** The "E" key on a keyboard and the "A" button on a gamepad act as the eating trigger for the dolphin.
    f. **"1"**, **"Y":** The "1" key on a keyboard and "Y" button on the gamepad toggles the dolphin's view as a wireframe.
5. **Inputs for Moving the Orbit Controller:**
    a. **"Left arrow", "Right Joystick"**: The "Left arrow" key, and left motion with the "Right Joystick**"** on a gamepad, simulates an Azimuth left orbit for the 3D Orbit Camera.
    b. **"Right arrow", "Right Joystick":** The "Right arrow" key on a keyboard, and right motion with the "Right Joystick**"** on a gamepad, simulates an Azimuth right orbit for the 3D Orbit Camera.
    c. **"Up Arrow", "Right Joystick":** The "Up arrow" key on a keyboard, and upwards motion with the "Right Joystick**"** on a gamepad, simulates an orbit elevation for the 3D Orbit Camera.
    d. **"Down arrow", "Right Joystick":** The "Down arrow" key on a keyboard, and downwards motion with the "Right Joystick**"** on a gamepad, simulate a negative orbit elevation for the 3D Orbit Camera.
    e. **"I", "Right Joystick":** The "I" key on a keyboard and a "Right Joystick" button click on a gamepad simulates zooming in with the 3D Orbit Camera.
    f. **"O", "Left Joystick":** The "O" arrow key on a keyboard and s "Left Joystick" button click on a gamepad simulates zooming out with the 3D Orbit Camera.
6. **Inputs for Overhead Camera:**
    a. **"NUMPAD8"**, **"D-PAD UP":** The "NUMPAD8" key on a keyboard and "D-PAD UP" button on the gamepad simulates panning the overhead camera up.
    b. **"NUMPAD6"**, **"D-PAD RIGHT":** The "NUMPAD6" key on a keyboard and "D-PAD RIGHT" button on the gamepad simulates panning the overhead camera right.
    c. **"NUMPAD2"**, **"D-PAD DOWN":** The "NUMPAD2" key on a keyboard and "D-PAD DOWN" button on the gamepad simulates panning the overhead camera down.
    d. **"NUMPAD4"**, **"D-PAD LEFT":** The "NUMPAD4" key on a keyboard and "D-PAD LEFT" button on the gamepad simulates panning the overhead camera left.
    e. **"NUMPAD9"**, **"RT":** The "NUMPAD9" key on a keyboard and "RT" button on the gamepad simulates zooming in with the overhead camera.
    f. **"NUMPAD7"**, **"LT":** The "NUMPAD7" key on a keyboard and "LT" button on the gamepad simulates zooming out with the overhead camera.

7. **Additional Inputs:**
    a. **"Esc":** the "Esc" key on a keyboard will terminate the game.
    b. **"F",** "**BACK":** The "F" key on a keyboard and "BACK" button on the gamepad will toggle the screen from full screen to windowed.
    c. **"Space", "START":** The "Space" key on a keyboard and "START" button on the gamepad will pause/un-pause the game.
    d. **"T"**, **"X":** The "T" key on a keyboard and "X" button on the gamepad toggles the XYZ axis lines instantiated in the world.
8. **Description of Node Controllers**
    a. **tage/nodeControllers/BounceController.java**
        i. This node controller will make food stations bounce when the dolphin first collides with them. The idea behind this bounce is that it simulated the food station depositing food to the players food storage.
9. **Scenegraph Relationships**
    a. The game now includes a food torus, which is a child of the Dolphin. This food torus is the indicator that tells the player the state of the food storage buffer. When a food station is collided with, the food buffer stores the food.
        i. If the food buffer is empty, the food Torus will disappear from the view.
        ii. If the food buffer is not empty, the torus will rotate around the dolphin.
    b. This behavior will continue throughout the game as you interact with food stations
10. **Changes made to TAGE Engine**
    a. **tage/Camera.java**
        i. ***(unchanged from a1)*** Added a global yaw function to turn the camera around the World Y Axis
        ii. ***(unchanged from a1)*** Added a local yaw function to turn the camera around the its local Y Axis
        iii. ***(unchanged from a1)*** Added a local pitch function to simulate the player looking up and down
    b. **tage/GameObject.java**
        i. ***(unchanged from a1)*** Added a global yaw function to give objects turning capabilities
        ii. ***(unchanged from a1)*** Added a local pitch function to give objects upward/downward movement
    c. **tage/HUDmanager.java**
        i. ***(unchanged from a1)*** Created functionality for a third and fourth HUD display
        ii. Created local data to keep track of the Canvas width/height
        iii. Added a function "*isCanvasTabletMode()"* that checks if the Canvas window has reached a certain smaller window size
        iv. Added a function "*setWindowDimensions()"* to set the actual Canvas Dimensions
        v. Added a function "*adjustRelativePosition()"* to adjust the position of the HUD based on the Canvas location
    d. **tage/Viewport.java**

i. Added function "*getLeftOnCanvas()"* which mimics "*getActualLeft()*", except it does not add the Canvas left value

e. **tage/CameraOrbit3D.java**
   i. Created a new class to manipulate the main camera and do orbit manipulations about an avatar
   ii. The orbit camera allows zooming, elevating, and rotating about the avatar
   iii. CamerOrbit3D also contains 3 sub classes to deal with the zoom, elevation, and rotation

11. I was able to meet <u>all</u> the requirements for the assignment.

12. Addition Game Features not required:

a. ***<u>(unchanged from a1)</u>*** I created an additional texture and randomized the selection of them among the items

b. ***<u>(unchanged from a1)</u>*** I created two additional HUD displays to determine the time and also the food level of the player
   i. **(new)** The same amount of HUDS as a1 remain, but this time, the HUDs were updated to reflect the new requirements
   ii. **(new)** The colors have been fixed so the regular displayed ones share a color
   iii. **(new)** A HUD appears when there is a winner and that one has a different color than the other 3

c. ***<u>(unchanged from a1)</u>*** I created collections of prizes and food Stations. They load dynamically based on the number of each that is set in the code.
   i. **(new)** The prizes are also dynamic and the number can be configured through MyGame.java.

d. ***<u>(unchanged from a1)</u>*** I created functions that randomize floats on certain bounds and it allows the game objects to dynamically sit on the page at random locations. It gives the world a very space-like feel.
   i. **(new)** The values used in the functions were updated in a2

e.

13. Assets Used: *<u>(this list is unchanged from a1)</u>*

a. assets/textures/Dolphin_HighPolyUV.png
   i. Already existed in the TAGE engine

b. assets/textures/Floral_Sheet.png
   i. Image was captured by me on my mobile phone (jpg converted to a PNG)

c. assets/textures/Wood_Desk.png
   i. Image was captured by me on my mobile phone (JPG converted to a PNG)

d. assets/textures/Drawer_Door.jpg
   i. Image was captured by me on my mobile phone

e. All other assets that TAGE instantiates itself