Modern Computer Games
COMP 521, Fall 2023

Assignment 2

Due date: Wednesday, October 18, 2023, by 23:59pm

Note: Late assignments will only be accepted with prior **written** permission of the instructor. You must **explain** all answers and **show all work** to get full marks! Please make sure your code is in a professional style: **well-commented,** properly structured, and appropriate symbol names. Marks will be very generously deducted if not!

Description

Note that this assignment requires you build a scenario with 2D game mechanics. **This must be done in Unity**. To make it easier to draw lines, points and text in Unity, an optional 2D line-drawing asset package is provided in MyCourses.
Note that in this assignment **you must do your own collision detection/resolution and manage your own physics**. All object movement and collisions should be done entirely by code you write! You may, however, continue to use any existing, built-in primitives for doing actual intersection calculations between primitive geometric objects/shapes (points, lines, rectangles, triangles, circles). **Do not use the Unity physics (colliders) to detect or respond to collisions.**

1. First, you need to produce a game terrain. This will be a **2D profile** of two small ponds, each surrounded by some amount of horizontal ground, and separated from each other by a triangular obstacle extending approximately 1/3 of the scene height. See below for the general layout. You do not need elaborate textures, but your ponds should be recognizably different from the surrounding ground. **5**

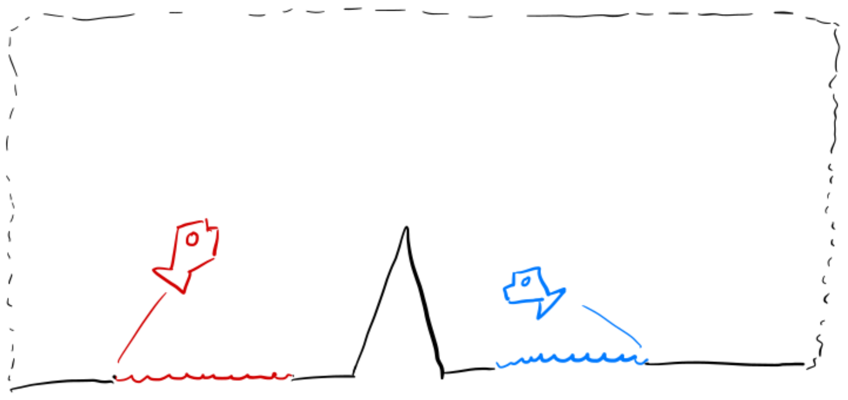

Figure 1: Sketch of terrain and level design.

Each pond contains an infinite number of fish. The main game action is to launch a fish from one pond, aiming to propel it over the obstacle and land (water?) in the opposite pond.

A fish is represented by a simple polygon forming its general shape, inside of which is a relatively large circular eye. You can come up with your own fish design, but its outer shape must be a non-convex polygon of at least 8 vertices, it must have a fairly large eye, and should have a recognizable mouth and tail-fin. An example is shown below.

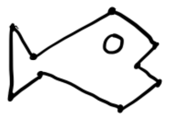

Figure 2: A 9-vertex fish.

2. A launched fish spawns from the edge of its pond closest to the screen edge. Once launched the fish motion should follow a parabolic arc, with the angle and initial velocity selected randomly at each launch, but generally aimed toward the opposite pond. Implement code to move the fish appropriately, but note that the projectile motion is only applied to the fish eye (motion of the fish body will be addressed in questions 4, 5 and 6). **10**

   Ensure that your choice of random values for both parameters results in the fish being frequently able to successfully clear the obstacle and land in the opposite pond, but also that they can easily fail by missing the pond, encountering the obstacle, and even exceeding the scene bounds. Scale the speed so the movement arc is easily observed for most parametrizations.

   Launching of a fish on the left side of the screen should be initiated by pressing "1", and a launch of a fish on the right side by pressing "2". Multiple fish may be in flight at the same time—your implementation must be sufficiently performant that at least 3 fish can be in play from each side at the same time.

3. Fish interact with other fish only through collisions between their respective eyes (i.e., we don't care if their body polygons overlap). Fish eyes also collide with the terrain (non-water ground and the central obstacle). Design and implement your own collision detection and response—collisions should result in a visible response more than just preventing overlap. **10**

   You will also need to detect collisions with the ponds—a fish disappears once its eye encounters water (either pond). Fish should also disappear if they exceed the left/right scene bounds, or after around 5–10s, whether in motion or not (the latter is just to avoid an accumulation of fish).

4. The position of the fish body depends on the position of its eye. Use Verlets to model the body of the fish, moving each vertex on the outline of the fish body independently. **10**

   In order to keep the fish together, design appropriate constraints between the vertices. Your constraints must ensure that the eye remains inside the body. Do not move the eye as part of your Verlet constraint solving, only body vertices (the eye position is entirely based on its projectile motion and collision responses as described in the previous questions, and is thus more of a reference point for the body vertices). Allow some distortion and flexibility of the fish body and its relation to the internal position of the eye, but it should restore its fish shape in the absence of other forces.

   Provide a document describing what constraints you impose and how you enforce them.

5. While fish bodies do not interact with other fish, they do interact with the (non-water) terrain. Extend your Verlet model to try and ensure body vertices do not penetrate the ground or obstacle. Note that the eye is not controlled by Verlet motion, and thus a solution may not exist that removes the penetration while also restoring the fish shape: prioritize removing penetration, and be sure to bound the iterations when constraint solving. **5**

6. (Bonus question, but max 100%) Also ensure the edges forming your fish body do not intersect the ground or obstacle. If you address this provide a separate document describing your strategy for doing so. **(5)**

What to hand in

You must submit your Unity source code, along with your document describing your Verlet constraints (and possibly the bonus question document).
For the Unity questions, hand in a zip file of your project code containing all files needed in order to reconstruct and run your simulation. Please avoid excessive use of high-resolution assets, as it easily results in very large archives. Allow ample time to upload your assignment solution, and please follow the general Unity instructions given in MyCourses.

For the document(s) submit a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files.

Assignments must be submitted on the due date **before midnight**. Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**.
This assignment is worth 15% of your final grade. **40**