1. Overview

The purpose of this assignment is to introduce you to OpenGL animation for hierarchical modeling and collision detection. You will program a 3-D vivarium: a simulated 3-D world of polyhedral creatures moving around.

Basic Requirements

1. **Predator/Prey Models:** To build your vivarium, you will first need to construct two different creatures using polyhedral (solid) parts. Implement your code as we did at "TODO 1" in the *ModelLinkage* file.
   - For the basic parts of your creatures, feel free to use routines provided with the previous assignment. You are also free to create your own basic parts, but they must be polyhedral (solid).
   - The creatures you design should have moving linkages of the basic parts: legs, arms, wings, antennae, fins, tentacles, etc.
   - Model requirements:
     - Predator: At least one (1) creature. Should have at least two moving parts in addition to the main body
     - Prey: At least two (2) creatures. The two prey can be instances of the same design. Should have at least one moving part.
     - The predator and prey should have distinguishable different colors. You are welcome to reuse your PA2 creature in this assignment.
     - All models should have a distinguishable head/front from the tail/rear side.
   - Vivarium requirements:
     - A test scene with only one (1) predator and one (1) prey, bind to key “T/t”
     - Default scene with at least one (1) predator and two (2) prey, reset after pressing key “R/r”

2. **Model animation:** Use transforms to control the motion of each creature and its parts. Implement your code at "TODO 2," following the example of the *animationUpdate* method in *ModelLinkage.py*.
   - Set reasonable joints limit for your creature
   - The creature's limbs should move back and forth in a periodic manner as it explores the vivarium. For example, a bird would continuously flap its wings.

3. **Collision Detection & Reaction:** Creatures in the vivarium should react to the positions of other creatures and move accordingly. Implement your code at "TODO 3" in the *stepForward* method of the *ModelLinkage* file.

   - Your creature should always stay within the fixed-size 3D "tank". You should do collision detection between the creatures and tank walls. When creatures hit the tank walls, they should turn and change direction to stay within the tank.
     - Using bounding box or bounding sphere for collision detection between creature and wall
     - After collision, the outgoing direction should be a proper reflection

   - Your creatures should have a prey/predator relationship. For example, you could have a bug being chased by a spider, or a fish eluding a shark. This means your creature should react to the presence of other creatures in the tank.
     - Use potential functions to change each creature's direction based on other creatures’ locations, their inter-creature distances, and their current configuration.
     - You should detect collisions between creatures.
       1. Predator-prey collision: The prey should disappear (get eaten) from the tank.

        2. Collision between the same species: They should bounce apart from each other. You can use a reflection vector about a plane to decide the after-collision direction.
    - You are welcome to use bounding spheres for collision detection.
    - We recommend that you modularize shared functions (making two creatures bounce off each other, detecting if a creature is within the tank walls, etc.) within the *EnvironmentObject* class. This will allow all creatures to inherit the same basic methods, which will form the backbone of each *stepForward* routine.

  - Your creatures should be able to move in 3 dimensions, not only on a plane.

4. **Creature orientation:** Creatures should face in the direction they are moving. For instance, a fish should be facing the direction in which it swims. Remember that we require your creatures to be movable in 3 dimensions, so they should be able to face any direction in 3D space. Implement your code at "TODO 4" in the *rotateDirection* method of *EnvironmentObject.py*.

5. **(CS680 Required, Extra credit for CS480 students) Feed your creatures with food:** Add chunks of food to the vivarium which can be eaten by your creatures. Implement your code at "BONUS 5(TODO 5)" in the *Vivarium* file.
    - When 'f' is pressed, have a food particle be generated at random within the vivarium.
    - Be sure to draw the food on the screen with an additional model. It should drop slowly to the bottom of the vivarium and remain there within the tank until eaten.
    - The food should disappear once it has been eaten. Food is eaten by the first creature that touches it.

Extra Credit (Maximum 10 points)

6. **Flocking behavior:** Add at least 5 creatures to the vivarium and make it possible for creatures to engage in group behaviors, for instance flocking together. This can be achieved by implementing the Boids animation algorithms of Craig Reynolds. Implement your code at "BONUS 6" in the *ModelLinakge* file.
7. **Creature stays upright**: Use an extra corrective factor to ensure the creature always remains upright. The model also needs to have a distinctive upper and lower body, so the behavior is obvious enough for grading.

Programming Style

For any modified or newly added source file, you should include a brief description of how this file was changed. Add this information to the file heading along with your name and ID. Your code should be readable with sufficient comments. You should use consistent variable naming and keep reasonable indentation.

README

With each assignment, please upload a README text or markdown file with the following information:

- your name
- any collaborators that you spoke to
- the class
- the assignment number
- a brief summary of the code and implementations you have written for the assignment

For the last point, you should outline the method you used to solve the problem, describe the variables and data-structures used in the program, any error conditions which might be encountered in the solution, and how these error conditions are handled by the solution.

If these details are described in the comments above the relevant functions or files, you may be brief here. You may call this file "READMEStudent" or something analogous to differentiate it from the README in the skeleton code.

Please include a mention of any resources that you consulted while completing your assignment.

2 Resources

2.1 Starter code

A Python program skeleton, which includes basic classes, methods, and main pipeline, is provided for you. You are expected to complete the parts marked with TODO. There are comments in the skeleton code that will help guide you in writing your own subroutines.

PA3_Fall2024.zip

2.2 Environment setup

Installing the appropriate programming environment should be covered in a lab session. For more step-by-step instructions, please check the "Environment Setup".

2.3 User Interface

R/r: Reset everything in the Vivarium (including put eaten creatures back)

T/t: Test scene with only one predator and one pray

F/f: Add food

(Optional) You can set your key bindings to add creatures to the Vivarium.

2.4 Video Demo

A video demo has been prepared to help you better understand your tasks and speed up your debugging process:

PA3 Demo.m4v

3. Submission: Due Tuesday 11/5, 11:59 PM EST(midnight)

3.1 Files to Submit

Your program's source files are to be submitted electronically on Gradescope. The code you submit should conform to the program assignment guidelines.

3.2 Demo

Part of your grade for this programming assignment will be based on your giving a short demo (10 minutes) during the CS480/680 scheduled labs following the assignment due date. You will be expected to talk about how your program works, and will be asked related comprehension questions.

4. Grading

Students can receive at maximum 10 points of extra credit.

Requirements	**CS480**	**CS680**
Produce predator & prey with moving parts	25	25
Creatures face in direction they are moving	25	25
Collision detection and creatures react to each other	20	20
Creatures stay inside the tank as described above	20	20
Food can be added, creatures find and eat the food as described above	5 (extra)	10
Group behavior modeling	10 (extra)	10 (extra)
Creature stays upright	2.5 (extra)	2.5 (extra)
Programming Style	10	10

5. Code Distribution Policy

You acknowledge this code is only for the course learning purpose. You should never distribute any part of this assignment, especially the completed version, to any publicly accessible website or open repository without our permission. Keeping the code in your local computer or private repository is allowed. You should never share, sell, gift, or copy the code in any format with any other person without our permission.