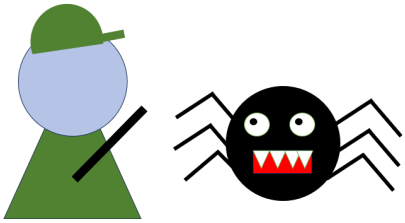
Game Programming Assignment: Hunters vs Monsters

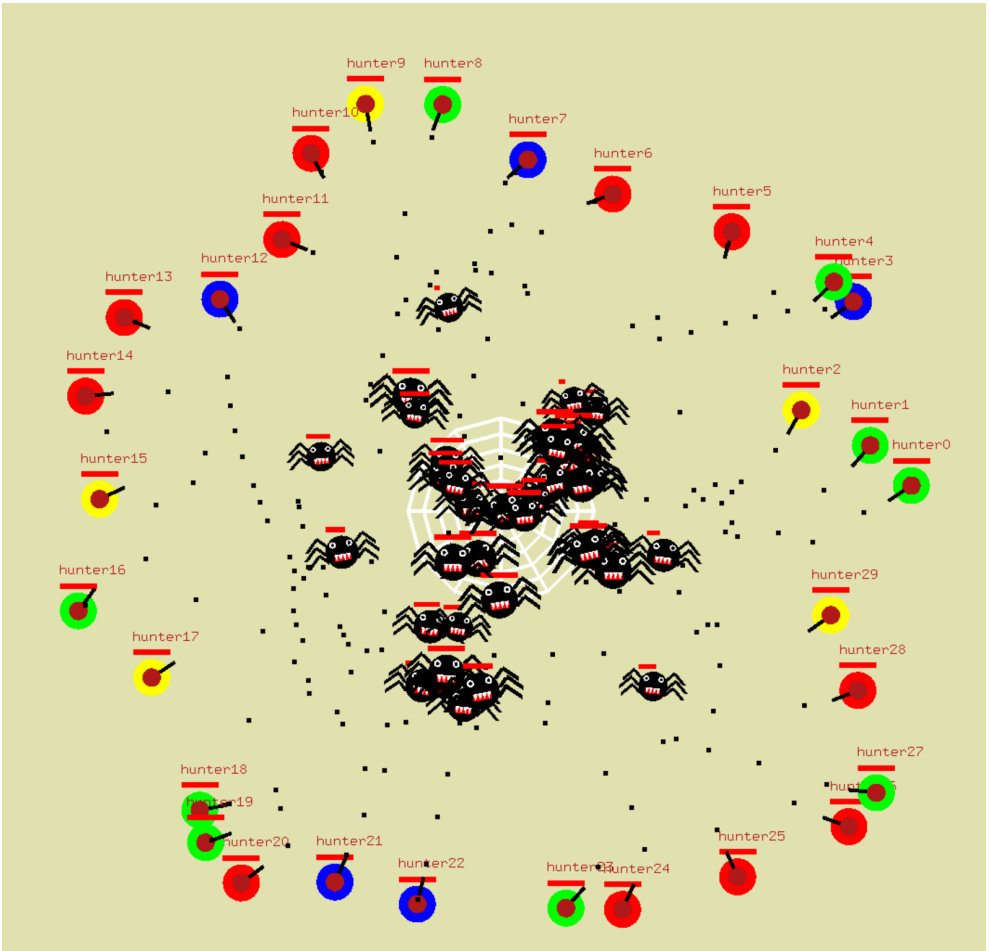


Due Date: *date and time*
Prerequisites: Students should have acquired skills in object-oriented programming using C++ and basic skills in computer graphics programming, and have learned basic data structures and algorithms.

Introduction

Monsters are swarming out of their nest, and now is the moment for hunters to join forces and battle against the monsters. Strategize and fight with the team to defeat the monsters!

- Create your own hunter by finishing the `MyHunter` class.
- Each student implements and submits their own hunter. The instructor will combine all submissions into a single Visual Studio project and demonstrate it in the class.
- The figure below illustrates what to expect when combining all hunters to battle against the monsters.



Requirements

A code template for this assignment has been provided. You only need to work on three functions in the `MyHunter` class: the class's constructor, `update()` function, and `circleIntersect()` function. The `MyHunter` class inherits from the `Hunter` class.

Please note that clones of the hunter you created are added to the scene. This is for testing purposes, illustrating what a crowd of hunters looks like. In the final demonstration, the instructor will replace each of these clones with a different student's hunter implementation. The hunters do not collide with each other or are hurt by bullets fired by the hunters.

When you test your code, you can edit the `setting.txt` file to change the number of the clones by specifying a different value for the variable of "Hunter Number". Please note that increasing the value of "Hunter Number" will result in a greater number of monsters in the scene.

The grading for this assignment is based on the following criteria:

- **Customizing Your Hunter (10 points)**:
    a) Customize your hunter by providing a name and modifying its color and capability attributes.
    b) Utilize the `upgrade(unsigned int armor, unsigned int speed, unsigned int shotgun, unsigned int bullet)` function to specify the hunter's capability attributes.
        i) You have a total of 20 points for upgrading, and each attribute (`armor`, `speed`, `shotgun`, and `bullet`) can't exceed 10 points.
        ii) Consider the following instructions for attribute upgrades:
            (1) Upgrading the `armor` increases the hunter's health.
            (2) Upgrading the `speed` boosts the hunter's moving speed.
            (3) Upgrading the `shotgun` increases the frequency of firing.
            (4) Upgrading the `bullet` increases the speed of the bullet.

- **Implementation of Update Function (60 points):** You are required to complete the function `void update(float deltaTime, const vector<Monster*> monsters, const vector<Hunter*> hunters)`. This function is executed in OpenGL's idle callback function in the main file. You should calculate the hunter's position values (`vec2(x, y)`) and the rotation angle (in degrees) of its shotgun using the provided lists of monsters and hunters (including your own hunter), along with the delta time. While it's not mandatory to use all of these, you should implement appropriate logic to control your hunter's behavior in response to the movements of the monsters

and/or other hunters. For example, consider aiming at the nearest monster and firing. Please review the variables in the `Monster` class to understand its properties.

- **Implementation of Collision Detection (30 points):** You are required to complete the function `bool circleIntersect(vec2 c1, vec2 c2, float r1, float r2)`. `c1` and `c2` are the center positions of the two circles; `r1` and `r2` are their respective radii. This function determines whether or not two circles intersect. In this assignment, the hunter and monsters use bounding circles for collision detection. The `circleIntersect()` function is executed in OpenGL's idle function in the main file to determine collisions between the hunter and monsters.

- **Bonus Opportunities:** Earn bonus points if your hunter accomplishes the following challenges at the end of the game:
    a) 2 bonus points if your hunter is still alive.
    b) 0.5 bonus point for each monster your hunter kills.

Gaming Features Provided

In the code template, the following features have been implemented. Understanding them may help the design of the logic for updating your hunter:

- **Game Duration**: The game in the final demonstration will last for one minute.
- **Automated Shooting**: Hunters automatically shoot bullets using the `rotation` variable in `Hunter` class. Please see `Hunter.h`. The `rotation` variable represents the direction in which the hunter is aiming.
- **Monster Behavior**: The behavior of monsters has been programmed to chase the nearest hunter.

There are a few helper functions you may consider to use:

- `glm::distance(p1, p2)` calculates the distance between two positions, `p1` and `p2`.
- `glm::normalize(p2 - p1)` normalizes a vector to a unit vector. Assuming `p1` is the hunter's position and `p2` is a monster's position. `p2 - p1` results in a directional vector that can be used for aiming at the monster.
- To find the rotation angle in degree for such aiming, you can use `glm::degrees(atan(p2.y - p1.y, p2.x - p1.x))`.

Submission Rules

1. Zip and submit the source code (all .h files and .cpp files) to Mycourses. DON'T submit the whole Visual Studio project.
2. Name the archived file (.zip, .7z, …) with your name, e.g., A05-John-Smith.