**Assessment Guidelines**

**Overview**

This document outlines the guidelines for instructors and teaching assistants to use as they assess the assignment. It is intended to give structure and guidance to the grading process to ensure that the most important components of the assignment are highlighted, and to promote consistency across sections.

In all cases, the instructor has the latitude to deviate from this rubric on a case-by-case basis to provide the holistic assessment that, in their judgment, best matches the assignment.

**Fundamental Concepts**

This assignment is designed to highlight the following fundamental concepts:

* Inheritance
* Polymorphism
* Correct use of composition (HAS-A) relationships

**Scoring**

The following categories are used to score this assignment:

/10 - Functionality: Bullets

/10 - Functionality: Basic target creation / movement

/10 - Functionality: Collisions and scoring

/10 - Functionality: Multiple target types

/13 - Design: Inheritance

/10 - Design: Overriding functions

/10 - Design Cohesion / modularity

/10 - Code: General code quality

/10 - Code: Style

/7 - Above and beyond requirements

----------

/100 - Total

**OVERALL SCORING**

For each of the categories the following general scoring guidelines apply:

* 100% - Correctly implemented.
* 85% - Minor error exists.
* 70% - A major error or multiple minor errors exist.
* 50% - A good attempt was made.
* 0% - No attempt made.

If necessary, scores in between the above classifications can be assigned.

**FUNCTIONALITY**

* Minor issues - Not adhering to the spec for the rules of the game, or speed. (If things have been changed for an "above and beyond" feature, and it's obvious that the concepts are understood, then changing the rules of the game is not a problem.)
* Major issues - Required functionality missing (e.g., bullets firing at incorrect angles, targets not being hit correctly).
* Attempt made - A game element is there, but doesn't really do anything. Or, this could also include cases where code is present for the class, but it doesn't render on the screen at all.

**INHERITANCE**

Things to look for:

* They should have base flying object.
* Bullet and target should both inherit from flying object.
* Specific kinds of targets should inherit from target. (It is fine to either have target be an abstract base class and then have 3 derived classes from it, or to have the base target class be used for the normal target and only 2 derived classes.)
* The base class should handle: Point, velocity, advance(), and is\_off\_screen()

Scoring guidelines:

* Minor issues - Inheritance is used for all obvious components, but additional opportunities exist to remove redundant code
* Major issues - Inheritance is used for some things (e.g., point and velocity) but not for other obvious things (e.g., advance())
* Attempt made - The program has inheritance, but doesn't take advantage of it

**OVERRIDING FUNCTIONS (POLYMORPHISM)**

Things to look for:

* a draw function should exist on each derived class
* hit function should probably be in the base "target" class and then overridden in the strong class. (Normal and safe can potentially take advantage of the same logic, but strong logic is likely different)
* The virtual functions should be used in Game (for example, target.draw and target.hit) -- Make sure they aren't putting logic in the game class that says "if type == strong\_target, then do a different kind of hit")

Scoring guidelines:

* Minor issues - Separate overridden functions are not cleanly separated.
* Major issues - Hit or draw function is all in one place / overly complex / lots of ifs.
* Attempt made - Methods were overridden, but then in Game class, if statements were used to determine the type of target to call separate methods, rather than using the virtual functions.

**COHESION / MODULARITY**

This is scored more or less holistically, with questions like the following in mind:

* Do classes contain what they should and nothing else?
* Is there logic in a base class that does not apply to all derived classes?
* Is game doing too much with the inner workings of a class that should be handled via methods?

**CODE QUALITY**

For this item, we are looking for general readability. Consider items like the following:

* Is logic overly complex (e.g., to determine a calculation)
* Are constants used instead of hard coded values

**STYLE**

We are looking for the following:

* Class and method headers (should be using **"""** rather than **#**)
* Appropriate inline comments
* Avoiding too short or non-descriptive variable names (using a short name is fine for a short lived variable such as "**for b : self.bullets**", but not for a member variable.
* Appropriate use of whitespace (vertical and within an expression)

**ABOVE AND BEYOND**

We are looking for two "minor" additions or one "major" one. Use your best judgment here.