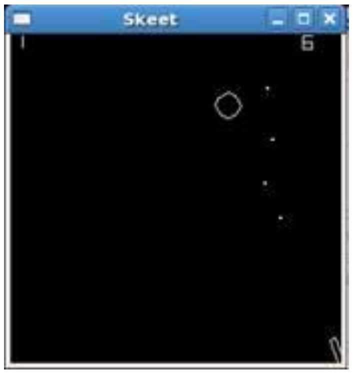
Project 3 - Advanced Skeet
CS 165

Skeet or Trap Shooting is a game played by hunters where a clay pigeon is ejected from a throwing device and a marksman attempts to shoot them with a shotgun. Sadly, many clay pigeons (also called "targets" or "birds") are lost during the typical game. Though this sport is an Olympic event and played the world over, it is far too dangerous to play in the computer lab. Therefore, we will build a safer version where nothing but pixels are shot.

Overview



Your assignment is to create a game that simulates skeet shooting. On the left side of the screen, clay pigeons are randomly shot across the screen. On the bottom right corner of the screen, the "marksman" (the term is used very loosely here) aims his rifle. Each match ends when either a bullet from the rifle destroys the pigeon or when the pigeon exits the screen (called "birds away").

The game will be played using OpenGL. A simplified interface to this library is provided:

```
/home/cs165lc/prj3/uiInteract.h     ← Header file describing the OpenGL interface
/home/cs165lc/prj3/uiInteract.cpp   ← Source for uiInteract.h. Should not change
/home/cs165lc/prj3/uiDraw.h         ← Header file with drawing function prototypes
/home/cs165lc/prj3/uiDraw.cpp       ← All the drawing functions
/home/cs165lc/prj3/game.cpp         ← Skeet game
/home/cs165lc/prj3/point.h          ← Ultra simple class describing a single point
/home/cs165lc/prj3/point.cpp        ← A few methods for the Point class
/home/cs165lc/prj3/velocity.h       ← velocity header file
/home/cs165lc/prj3/velocity.cpp     ← velocity class to manage changes for a point
/home/cs165lc/prj3/rifle.h          ← Header file for the Rifle class
/home/cs165lc/prj3/rifle.cpp        ← Rifle class to display the rifle
/home/cs165lc/prj3/makefile         ← Instructions to build skeet
```

`You can use the Linux command: "cp /home/cs165lc/prj3/* ." to copy all of the files.`

Advanced Features

In previous semesters, students complete a more simplistic version of Skeet, where only one kind of Bird crossed the screen. However, armed with the skills of inheritance and polymorphism, you are equipped to create a more interesting game that contains 3 different kinds of birds that fly by. After a bird has exited the screen (either because it was hit or because it reached the far side), another one of a random type should be created. The three types of birds are as follows:

1. A “regular” bird – This bird is shown by a circle, takes a single hit to be destroyed, and is worth 10 points.
2. A “strong” bird – This bird is shown by a circle with a number inside it. It takes 3 hits to be destroyed, and is worth 50 points.
3. A “safe” bird – This bird is shown by a star, and should not be hit, but a single shot will destroy it and result in the loss of 50 points.

User Interface

All the elements of the graphical user interface will need to be described. Please see the following code for examples of how your program will run:
`/home/cs165lc/skeet.out`

Architecture design

The entire program will need to be implemented using the principles of encapsulation. Please pay special attention to the design of these components so they can be as general purpose as possible. We will be using them for Project 4 later in the semester.

Some notes about the physics of the game:

- **Frame Rate**: The frame-rate is 30 frames/second
- **Birds**: For the playable version, the bird direction, velocity, and timing to release are random. When the game begins and after the previous bird is finished, a random delay of up to 1 second is introduced before the next random bird is released. The initial position of the bird will be anywhere on the left side of the screen. If the bird is on the top half, then it will have a generally downwardly facing direction. If it is released on the bottom half, then it will travel up and to the right. The horizontal velocity is between 3 and 6 pixels/frame. The vertical velocity is between -4 and 4 pixels/frame. Each bird has a diameter of 20 pixels.
- **Bullets**: Up to 5 bullets can be on the screen at a time. If there are less than 5 bullets on the screen, then the space bar will introduce another. Each bullet travels 10 pixels/frame. The direction is determined by the orientation of the rifle.
- **Rifle**: The rifle is in the lower right corner and can move from the vertical to the horizontal direction. The right arrow will move the rifle to the right 2 degrees/frame. The rate of motion increases to 3 degrees/frame with every 5 frames the arrow key is depressed. The left arrow behaves the same.

You should not have a separate variable for each kind of Bird, but rather, should have a single Bird * that can be set to a new instance of any type of Bird. Don’t forget to delete each Bird when you are finished with it.

To determine if a bullet hits the bird, it is necessary to determine if the two velocity vectors intersect. Hint: Use the subtraction operator from the velocity class from Project 2.

Error handling

As usual, your classes and the game itself will need to be robust to any type of user or file input. Extensive error handling should also be built into each class to ensure that clients of the class will use them correctly. Similarly, there should be no way the user can cause the program to malfunction due to incorrectly formed input.

Assignment: Design Document

Your assignment is to create a design document describing how you will write the code and the code for the working program. A few notes:

- The classes will have to exhibit the highest level of encapsulation design. They should be designed for reuse later in the semester.
- The program implementing the game will need to exhibit both procedural and data abstraction.

Please create a design document for your project following the guidelines outlined in the "Design Document" specification. Note that not all of the components mentioned in the specification will apply here.

Your assignment is to create a design document of sufficient detail to begin writing code. Face-to-face students must submit a hard-copy of the design document in class; no electronic files are accepted. The grading criteria are:

	Exceptional 100%	Good 90%	Acceptable 70%	Developing 50%	Missing 0%
Problem definition & design overview 10 %	Design overview paints a complete picture of how the code will be written	Design overview answers some "how" questions but there remains unanswered questions	Design overview does not answer any "how" questions	Components of the problem definition and design overview are present	Missing problem definition or design overview
Interface design 20%	There is no ambiguity in the interface design	The input <u>and</u> output are both described in detail	All the interface elements (output) <u>or</u> operations (input) are described in detail	Some mention of the interface design exists	Missing interface design
Structure chart 10%	All the functions exhibit high degrees of modularization	All the functions and interfaces are specified	The most important functions and interfaces are listed, but some are missing.	The program design is not described in sufficient detail to determine whether the solution is workable	No structure chart
Data structures 40%	Design exhibits a high degree of encapsulation	All the details of all the classes are described; nothing is missing	Both methods and member variables are described in the UML	UML is included describing some of the data-structures	Missing class definition
Algorithm design 10%	Algorithms are elegant, well thought out, and complete	Two functions provided that: · Are non-trivial · Bug free · Detailed enough to write code	One function provided that: · Is non-trivial · Bug free · Detailed enough to write code	Pseudocode exists that… · Enough detailed · Not C++ · Uses pseudocode keywords	Missing algorithm design
Error handling 10%	It is clear that error handling was thoroughly considered	Two complete internal errors are discussed, each of which can be readily turned into an assert	At least one error is completely described: how it will be detected and how it will be handled	A discussion exists regarding sources of error or error handling	Missing error handling section

You are required to attach the above rubric to your design document. Please self-grade.

Hint:

- You may want to look at the Pong Design Document for inspiration.
- Only describe code you will write; do not copy-paste anything from the Pong Design Document
- Skeet is considerably more complex than Pong. You will need additional data-structures.