

Problem Definition

This is a simple video game simulating the game of Skeet. The object of the game is for the user to shoot pigeons coming from the left side of the screen with a rifle stationed at the bottom right corner of the screen. The game continues until the player gets tired of playing.

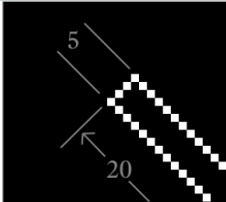
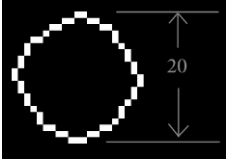
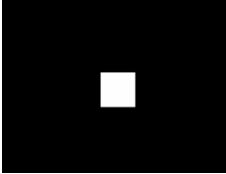
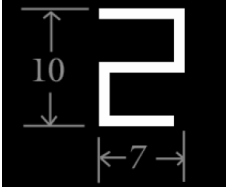
Design Overview

The game consists of a collection of entities (bird, rifle, and bullets). Each entity knows how to draw itself and advance itself. Of course, if the entity is dead, it does neither.

With each iteration of the game (signified by the `callback()` function getting called), all the entities are asked to advance and draw themselves. Input is sent to the rifle entity. Collision detection is performed between all alive bullets and the bird.

Interface design

Output

	Description
	Gun is a rectangle 20 x 5
	Bird, a circle of diameter 20
	Bullet: box 2 pixels square
	Score, 7 x 10 pixels

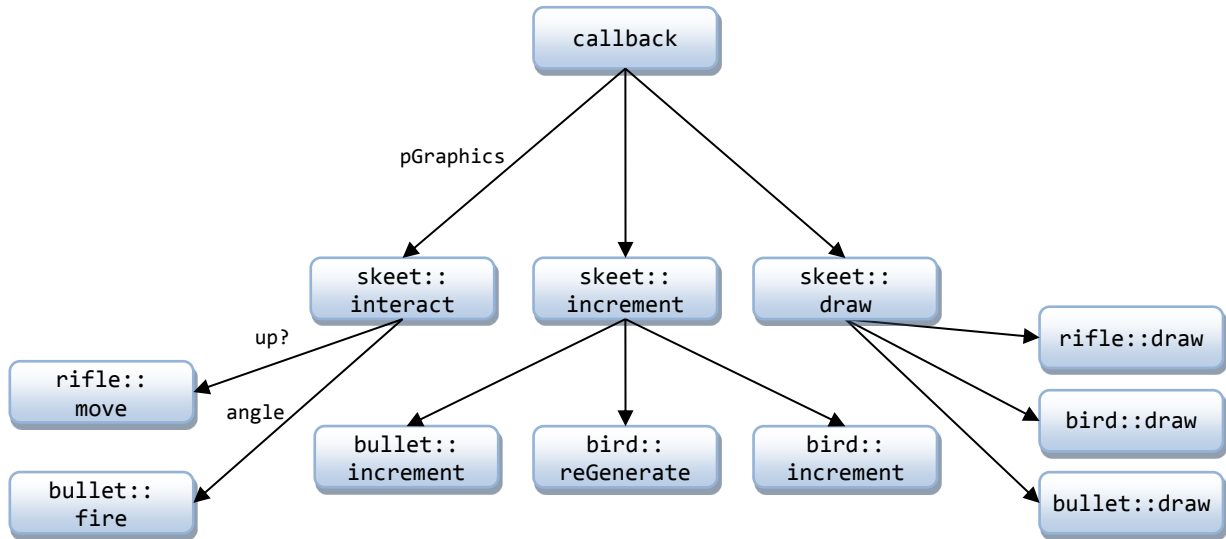
Input

Event	Action
Right arrow (left is inverse)	Angle decreases by 2 degrees first 5 frames, by 3 next 5 frames, etc.:
Space	If there is an available bullet, then fire it at the rifle's current orientation and with its current angle

Errors

No user errors are possible.

Structure chart



Data Structures

Point	Vector	Bird	Bullet	Rifle	Skeet
-x -y	-point	-vector	-vector	-angle	-rifle
-fNoCheck	-dx -dy	-dead	-dead	+Rifle	-bird
-xMin -xMax	+getPoint	+Bird	+Bullet	+move	-bullets
-yMin -yMax	+getDx	+regenera	+fire	+draw	-hit
+getX +getY	+getDy	te	+draw	+getAngle	-miss
+addX +addY	+getMax	+draw	+advance		+Skeet
+setX +setY	+getAngle	+advance	+isDead		+draw
+getXMin +getXMax	+setDx	+isDead	+kill		+interact
+getYMin +getYMax	+setDy	+kill			+advance
	+advance				
	+difference				

Algorithms

```
Bullet::fire(angle)
  x = xMax - 1
  y = yMin + 1
  dx = -cos(angle) * speedBullet
  dy = sin(angle) * speedBullet
  dead = false
```

```
Bird::regenerate()
  x = xMin
  y = rand(yMin ... yMax)
  dx = rand(3 ... 6)
  IF y > 0
    dy = rand(-4 ... 0)
  ELSE
    dy = rand( 0 ... 4)
```

```
Rifle::move(up, down)
  if (up)
    angle += (up + 9) / 5
  if (down)
    angle -= (down + 9) / 5

  angle = MAX( 0, angle)
  angle = MIN(90, angle)
```

```
Skeet::interact(ui)
  rifle.move(ui.isUp, ui.isDown)
  if (ui.isSpace)
    FOR iBullet = 0 .. numBullets
      IF bullets[iBullet].isDead
        bullets[iBullet].fire(rifle.getAngle)
```

```
Skeet::advance()
  FOR all bullets
    IF bullets[i].isAlive
      bullets[i].advance
      IF sizeBird > bullets[i].vector-bird.vector
        bird.kill
        bullet[i].kill
        score++
  IF bird.isDead
    IF 0 == random(0 ... 30)
      bird.regenerate
  ELSE
    bird.advance
```

File formats

There are no files

Error Handling

User Errors

There are no possibilities for user-induced error. Anything but a space and arrows are ignored. Those three are always valid.

File Errors

There are no file errors because the program does not interact with the file.

Internal Errors

Each class will have error checking:

Class	Errors
Point	All points will be limited : $xMin \leq x \leq xMax$, $yMin \leq y \leq yMax$. If the <code>noCheck</code> flag is set, then error handling will be ignored. All manipulation of the <code>x,y</code> member variables will be performed by the <code>set()</code> functions which will perform error checking.
Vector	There are no limitations set on <code>dx</code> and <code>dy</code> values. The <code>x,y</code> values are enforced by <code>Point</code> .
Rifle	Rifle orientation is enforced in <code>move()</code> who is the only method to change <code>angle</code> .
Bird	Bird position is enforced by <code>advance()</code> and <code>reGenerate()</code> who are the only methods to set the position.
Bullet	Bullet position is enforced by <code>advance()</code> and <code>fire()</code> who are the only methods to set the position.