# P5.19

Frank Mock

October 10, 2011

## 1 Specification

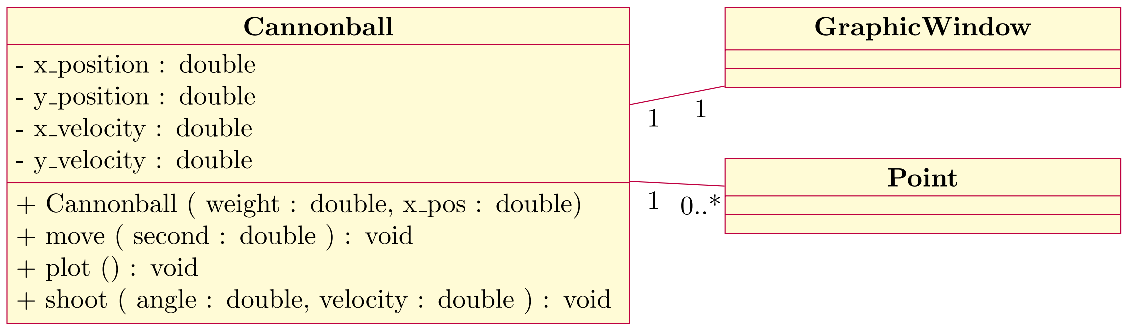
Design a class Cannonball to model a cannonball that is fired into the air. A ball has an x and a y position and an x and y velocity

Supply the following member functions:

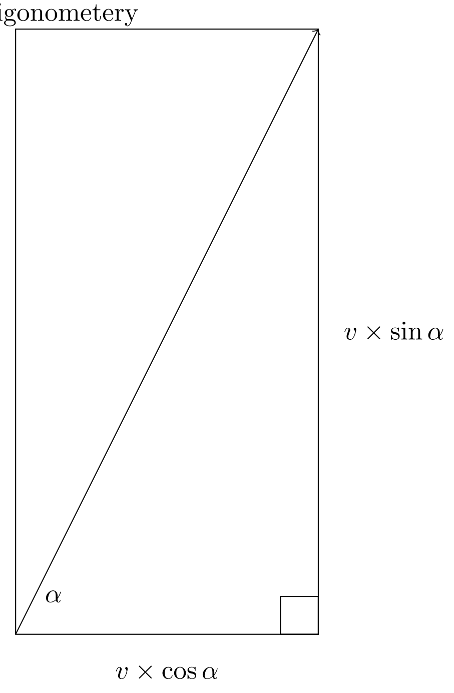
- A constructor with a weight and an x-position (the y-position is initially 0)
- A member function move(double sec) that moves the ball to the next position (First compute the distance traveled in sec seconds, using the current velocities, then update the x- and y-positions; then update the y-velocity by taking into account the gravitational acceleration of - 9.81 m/sec2; the x-velocity is unchanged.)
- A member function plot that plots the current location of the cannonball
- A member function shoot whose parameters are the angle a and initial velocity v (Compute the x-velocity as v cos a and the y-velocity as v sin a; then keep calling move with a time interval of 0.1 seconds until the y-position is 0; call plot after every move.)

Use this class in a program that prompts the user for the starting angle and the initial velocity. Then call shoot.

## 2 Design



The velocity can be split into x and y components using these formulas from trigonometery



## 3 Implementation

g is the gravitational acceleration of $-9.81\frac{m}{sec*2}$. a good approximation of $\pi$ is $4 \times \arctan 1.0$ Which we will need to convert from the angle in degrees to the angle in radians. The c++ sin and cos functions expect the argument to be in radians.

"cannonball.h" 3≡

```
#include"ccc_win.h"
#include<cmath>
const double g = -9.81; //m/sec x sec
const double pi = 4 * atan(1.0);
class Cannonball
{
private:
  double x_position;
  double y_position;
  double x_velocity;
  double y_velocity;
public:
  Cannonball(double w, double x);
  void move(double s);
  void plot() const;
  void shoot(double a, double v);
};
◇
```

Assume the initial y position is always zero (starting from the ground) Weight does not seem to affect the trijectory, but we store it for possible later use. Use the Point class from the authors graphic library to indicate the positions that the cannonball is in as it moves through its trajectory.

"cannonball.cpp" 4a≡

```
#include"cannonball.h"
Cannonball::Cannonball(double w, double x)
{
 x_position = x;
 y_position = 0;
 x_velocity = 0;
 y_velocity = 0;
}
void Cannonball::move(double s)
{
  x_position++;
}
/**
Use the Point class from the authors graphic library
to indicate the positions that the cannonball
is in as it moves through its trajectory.
*/
void Cannonball::plot() const
{

}
void Cannonball::shoot(double a, double v)
{
  do
  {
    move(0.1);
  }while(x_position < 50);
}
◇
```

In the main program, we create a cannonball, get initial angle and velocity from user then call shoot
"p5_19.cpp" 4b≡

```
#include"cannonball.h"
⟨ Include Files 5a ⟩
int ccc_win_main()
{
 Cannonball c(10, 20);
 double angle = cwin.get_double("Enter the launch angle in degrees: ");
 double velocity = cwin.get_double("Enter the initial velocity in m/s:");
 c.shoot(angle,velocity);
 cwin << Message(Point(0,0),"Test");
return 0;
}
◇
```

⟨ *Include Files* 5a ⟩ ≡

```
#include "ccc_win.h"
◇
```

Fragment referenced in 4b.

"p5_19.bat" 5b≡

```
g++ -mwindows -I C:\C++_Programs\CS-116\cannonball\cccfiles -o p5_19 p5_19.cpp cannonball.cpp ^
C:\C++_Programs\CS-116\cannonball\cccfiles\ccc_msw.cpp ^
C:\C++_Programs\CS-116\cannonball\cccfiles\ccc_shap.cpp ^
-lgdi32
◇
```