**Structure Chart**

A picture containing text, screenshot, diagram, line

Description automatically generated

**Class Diagrams:**

Bullet Class:

A picture containing text, screenshot, font

Description automatically generated

Fidelity: This class can be considered complete. It covers all required information for the projectile used in this program.

Robustness: This class has yet to be used or tested. So, it has a fragile level of robustness

Convenience: This class is easy to use and implement into a program to represent a bullet or projectile. Thus, it has seamless convenience.

Physics Class:

A picture containing text, screenshot, font, number

Description automatically generated

Fidelity: This class also covers all required states/attributes required for the physics of our program. Thus, it is complete in fidelity.

Robustness: This class has been tested and used already, but not formally and fully tested, so it has a tested level of robustness

Convenience: This class does require an understanding of physics to use in its fulness, but it does make working with numbers and the physics of our program a lot simpler. Thus, it has an easy level of convenience.

Angle Class:

A picture containing text, screenshot, font, number

Description automatically generated

Fidelity: This class is quite simple, but does cover all needed states for our angle, though it does use radians for both radians and degrees in angle measures. As such, this class can be considered complete in fidelity.

Robustness: This class has been tested and used, but has not been formally tested fully. Thus, it has a tested level of robustness

Convenience: This class is simple and easy to use to represent an angle in any program. It has a seamless level of convenience.

Trigonometry Class

A picture containing text, screenshot, font, line

Description automatically generated

Fidelity: This class fulfills all required functions. It has a complete level of fidelity.

Robustness: This class has been tested, but not entirely. Thus it has a tested level of robustness

Convenience: This class again, fulfills all required functions and is clear and easy to use. Thus, it has seamless convenience.

**Pseudocode:**

void setRo(double altitude)

IF altitudeToDensityMap.find(altitude) != altitudeToDensityMap.end()

ro <- altitudeToDensityMap[altitude]

ELSE

ro <- interpolation(altitude, altitudeToDensityMap)

void setGravity(double altitude)

IF altitudeToGravityMap.find(altitude) != altitudeToGravityMap.end()

gravity <- altitudeToGravityMap[altitude]

ELSE

gravity <- interpolation(altitude, altitudeToGravityMap)

void setSpeedOfSound(double altitude)

IF altitudeToSoundMap.find(altitude) != altitudeToSoundMap.end()

speedOfSound <- altitudeToSoundMap[altitude]

ELSE

speedOfSound <- interpolation(altitude, altitudeToSoundMap)

void setMach()

mach <- v / speedOfSound

void setC()

IF machToCMap.find(mach) != machToCMap.end()

c <- machToCMap[mach]

ELSE

c <- interpolation(mach, machToCMap)

void setDrag()

dragForce <- 0.5 \* c \* ro \* v \* v \* area

void setAcceleration()

acceleration <- dragForce / mass

void setDdx()

ddx <- -sin(radians) \* acceleration

void setDdy()

ddy <- -gravity -cos(radians) \* acceleration

double interpolation(double inputMiddle, map<double, double> m)

it <- m.begin()

WHILE it != m.end()

IF it.key > inputMiddle

inputEnd <- it.key

outputEnd <- it.value

it--

inputBegin <- it.key

outputBegin <- it.value

BREAK

it++

IF it == m.end()

it--;

return it->second;

RETURN outputBegin + (inputMiddle - inputBegin) \* (outputEnd - outputBegin) / (inputEnd - inputBegin)

**Test Cases**

Testing Constructor

We need to make sure that the start conditions are calculated correctly based on altitude and angle.

We have a display() method that will print to the screen the values of some of our private variables. The calculations have been configured such that the time interval between frames is set to one second, for simplicity. This helps verify the results when comparing them to the Whiteboard demo provided last week, which is the first test case.

NOTE: each output is calculated using a setter function!!! This means in addition to testing the Constructor, these test cases are testing 11 additional setter functions.

|  |  |  |
| --- | --- | --- |
| Test Name | Input | Output |
| Whiteboard demo | Altitude: 0, angle: 30 | dx: 413.5  dy: 716.203  ro: 1.225  gravity: 9.807  speed of sound: 340  mach: 2.43235  c: 0.25954  drag: 2048.6  acceleration: 43.8673  ddx: -21.9336  ddy: -47.7972 |
| Altitude in data tables | Altitude: 1000, angle 30 | dx: 413.5  dy: 716.203  ro: 1.112  gravity: 9.804  speed of sound: 336  mach: 2.46131  c: 0.257565  drag: 1845.48  acceleration: 39.5178  ddx: -19.7589  ddy: -44.0274 |
| Interpolation needed | Altittude: 1500, angle 30 | dx: 413.5  dy: 716.203  ro: 1.0595  gravity: 9.8025  speed of sound: 334  mach: 2.47605  c: 0.25656  drag: 1751.49  acceleration: 37.5052  ddx: -18.7526  ddy: -42.2829 |
| Input zeros | Altitude: 0, angle: 0 | dx: 0  dy: 827  ro: 1.225  gravity: 9.807  speed of sound: 340  mach: 2.43235  c: 0.25954  drag: 2048.6  acceleration: 43.8673  ddx: -0  ddy: -53.6743 |
| Max Altitude | Altitude: 80,000, angle: 30 | dx: 413.5  dy: 716.203  ro: 1.85e-05  gravity: 9.73  speed of sound: 324  mach: 2.55247  c: 0.25135  drag: 0.0299618  acceleration: 0.00064158  ddx: -0.00032079  ddy: -9.73056 |
| Side Shot | Altitude: 0, angle: 90 | dx: 827  dy: 0  ro: 1.225  gravity: 9.807  speed of sound: 340  mach: 2.43235  c: 0.25954  drag: 2048.6  acceleration: 43.8673  ddx: -43.8673  ddy: -9.807 |