Force and Motion

By John Ying

**Introduction**

In the world of video games, many times the player will be controlling a flying object. Through this force and motion program, the user will see the end calculation of a rocket lifting off from its destination in force and motion 2.cpp as well as how long it took the rocket to get back down to land. In the force and motion.cpp program, the user inputs a coefficient of friction and an initial speed. With those slots they get a result of how far that object traveled.

**Methods**

Using this type of formula, the change in movement in increments of tChange (change in time) will calculate the movement of the object (aka iPosition) until the object stops. The calculation of movement stopping is the while (iSpeed > 0) showing that the velocity is less than zero where the object hasn’t moved anymore.

acceleration = (-1.0 \* cFriction) \* gravity;

while ( iSpeed > 0 )

{

iPosition += iSpeed \* tChange;

iSpeed += acceleration \* tChange;

}

By Using the below “if” function in the calculation of a rocket, I was able to say that once the time hit one second, the thrust of the rocket would stop. To which inside the thrust vector increment would be set to zero.

if ( fTime > thrustTime )

In the below step, the following calculation projected the amount of change in the X-Direction based upon time, and velocity based off of the position it was currently at. This is also known as movement of an object in a direction from small increments of time. The same was repeated for the Y and Z coordinates.

PosX = PosX + xVelocity \* tStep;

Here is where the velocity is updated based upon the current acceleration and the amount of time. The same was repeated for the Y and Z coordinates.

xVelocity = accelerationX \* tStep;

This calculated the wind resistance based upon the coefficient of wind in the opposing direction of the velocity (pretty much the push in the opposition direction the rocket was heading). The same was repeated for the Y and Z coordinates.

wXResistance = (-1 \* cResistance) \* xVelocity;

The net force calculation in the example include all 3 because of the influence of weight (mass \* gravity) in the Z coordinate. The calculation of gravity was crucial into the calculation or else the rocket would more than likely be in space with wind resistance (which would be really quite weird) and continue to go on for a VERY long time.

xNet = xThrust + wXResistance;

yNet = yThrust + wYResistance;

zNet = zThrust + (mass \* gravity) + wZResistance;

Finally, the acceleration calculation at the end of the loop. Obviously calculated by taking the acceleration = NetForce / Mass. The calculations were needing to be done on also the Y and Z coordinates due to the factors of the calculation.

accelerationX = xNet/mass;

Lastly, to finish off the end of the update, the time was increased so that the next time the loop ran through, it wouldn’t repeat the same calculation upon itself over and over.

fTime += tStep;

**Results**

Below are some test instances of some random user input data. The user input data is the initial speed and the coefficient of friction. The program calculated data is the final position.

|  |  |  |
| --- | --- | --- |
| Initial Speed | Coefficient of Friction | Final Position |
| 26.7 | 0.25 | 146.823 |
| 135 | 0.82 | 1140.71 |
| 2.35 | 0.154 | 1.94896 |

Below is the predetermined set of results provided for the rocket. The final position and elapsed time is what the program calculated.

|  |  |
| --- | --- |
| Start Position | (0, 0, 0.2m) |
| Mass | 0.0742 kg |
| Coefficient of Wind Resist | 0.02 |
| Thrust Force (aka Magnitude) | 10.0 N |
| Thrust Duration from Start | 1.00 seconds |
| Heading | 23 degrees |
| Pitch | 62 degrees |
| Time-Step | 0.10 seconds |
| Final Time of Landing | 13.8 seconds |
| Final Position of Landing | (6.2384, 2.64804, -0.034726) |

**Conclusion**

Overall, the programming was pretty difficult and long. Majority-wise, the length was due to the fact that each X, Y, and Z coordinate needed to be calculated on its own. There was one hitch in the program that was somewhat strange. During the calculation of the Thrust direction increment, the C++ math was different than my calculator. In a simple calculation of Pitch (in radian form), the product of sin was off by a very large amount (when talking about decimals). By doing the sin(Pitch), it produced a result of 0.018885157 but on my calculator, it produced the result 0.882948.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/

/ Force and Motion.cpp simulates the motion of a box with a specific

/ amount of force in a direction. The box will have a user input

/ initial speed and a coefficient of friction that will apply to it.

/ It will output the final distance traveled with hidden calculated

/ time intervals of 0.10 seconds.

/

/ John G Ying, March 2012

/ Johnson County Community College

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <string>

#include <cmath>

using namespace::std;

void main()

{

double iSpeed;

double cFriction;

double tChange = .10;

double iPosition = 0;

double acceleration;

double gravity = 9.8;

double fPosition;

cout << "Please enter the initial speed:\n";

cin >> iSpeed;

cout << "Please enter the coefficient of friction:\n";

cin >> cFriction;

acceleration = (-1.0 \* cFriction) \* gravity;

while ( iSpeed > 0 )

{

iPosition += iSpeed \* tChange;

iSpeed += acceleration \* tChange;

}

cout << "The final distance traveled is: " << iPosition << " units." << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/

/ Force and Motion 2.cpp simulates the motion of a rocket launched

/ at a specific position, mass, wind resistance coefficient, thrust,

/ time length on thrust, heading, pitch, and time interval. The

/ program produces the final destination of the rocket after launch

/ and final landing (or crash) into the ground as well as how long

/ it took the rocket to get there.

/

/ John G Ying, March 2012

/ Johnson County Community College

/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <string>

#include <cmath>

using namespace::std;

void main()

{

double fTime = 0; //the current time of the program

double PosX = 0; //current x position of the rocket

double PosY = 0; //current y position of the rocket

double PosZ = 0.2; //current z position of the rocket

double tStep = 0.1; //the time step intervals between calculations

//the variables for magnitude, heading, and pitch variable increments in vector format

double xThrust;

double yThrust;

double zThrust;

double mass = 0.0742; //mass of the rocket in kilograms (kg)

double cResistance = 0.02; //coefficient of wind resistance (not actual wind resistance)

double force = 10.0; //upward force of the rocket in newtons

double thrustTime = 1.00; //

double heading = 23;

double pitch = 62;

double fHeading = 0;

double fPitch = 0;

//cout statements giving the user the starting statistics

cout << "The rocket begins at position (" << PosX << ", " << PosY << ", " << PosZ << ")" << endl;

cout << "The rocket has a mass of " << mass << "kg\n";

cout << "The coefficient of wind resistance is " << cResistance << endl;

cout << "The rocket provides a thrust of " << force << " N for " << thrustTime << " seconds.\n";

cout << "The heading is set at " << heading << " and the pitch at " << pitch << endl;

//degrees to radians

fHeading = heading / 57.29577951;

fPitch = pitch / 57.29577951;

//thrust force in vector format calculations

xThrust = force\*cos(fPitch)\*cos(fHeading);

yThrust = force\*cos(fPitch)\*sin(fHeading);

zThrust = force\*sin(fPitch);

//zSin should equal 0.018885157 but instead equals 0.882948

/\*

double zSin = 0;

zSin = sin(fPitch);

cout << "zSin " << zSin << endl;

\*/

double gravity = -9.8; //force of gravity in meters per second square

//net force coordinate direction

double xNet;

double yNet;

double zNet;

//wind resistance coordinate direction

double wXResistance = 0;

double wYResistance = 0;

double wZResistance = 0;

//acceleration coordinate direction

double accelerationX = 0;

double accelerationY = 0;

double accelerationZ = 0;

//velocity coordinate direction

double xVelocity = 0;

double yVelocity = 0;

double zVelocity = 0;

while ( PosZ >= 0 ) //while the rocket is above ground (0 coordinate)

{

if ( fTime > thrustTime ) //while the rocket is thrusting

{

//set thrust force to zero

xThrust = 0;

yThrust = 0;

zThrust = 0;

}

//update the current position

PosX = PosX + xVelocity \* tStep;

PosY = PosY + yVelocity \* tStep;

PosZ = PosZ + zVelocity \* tStep;

//update the current velocity

xVelocity = accelerationX \* tStep;

yVelocity = accelerationY \* tStep;

zVelocity = accelerationZ \* tStep;

//update the current resistance

wXResistance = (-1 \* cResistance) \* xVelocity;

wYResistance = (-1 \* cResistance) \* yVelocity;

wZResistance = (-1 \* cResistance) \* zVelocity;

//update the net force

xNet = xThrust + wXResistance;

yNet = yThrust + wYResistance;

zNet = zThrust + (mass \* gravity) + wZResistance;

//update the acceleration

accelerationX = xNet/mass;

accelerationY = yNet/mass;

accelerationZ = zNet/mass;

fTime += tStep; //update the current time with the increment

}

cout << "The final time is: " << fTime << " seconds." << endl;

cout << "The final position is: (" << PosX << ", " << PosY << ", " << PosZ << ")\n";

}

**Post Lab Questions**

1. The check for zero velocity in procedure in the behavior is a box is important because if the program does not correctly recognize that the object has obtained zero velocity, the object could technically KEEP going in the direction simply because if the program didn’t recognize that it stopped, then the only other thing it could do would be to keep going.
2. While both surface friction and wind resistance are both forces in the –v direction, they are two different forces due to several factors. In the chance that an object was not colliding with a surface (for example: flying), it wouldn’t have any surface friction at all but pure wind resistance. Also, an object can have BOTH forces at the same time. For example, if a car was braking after an initial velocity of 200 mph, then the braking (surface friction) AND wind resistance (increases and decreases with velocity) would “stack” with each other in contributing to the reduction of speed.
3. In applying wind speed to the motion of a rocket, the solution would ideally be quite simple. Within the same calculation of thrust, apply another type of “thrust” as the force of the wind. So by calculating the direction of the rockets thrust, the program would then add (+) the direction of the force of the wind.