Reference Manual

Generated by Doxygen 1.8.13

Contents

1 File Index

1.1 File List

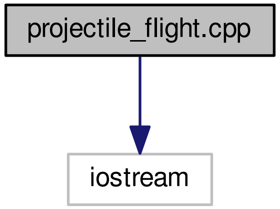
Here is a list of all files with brief descriptions:

2 File Documentation

2.1 projectile_flight.cpp File Reference

```
#include <iostream>
```

Include dependency graph for projectile_flight.cpp:



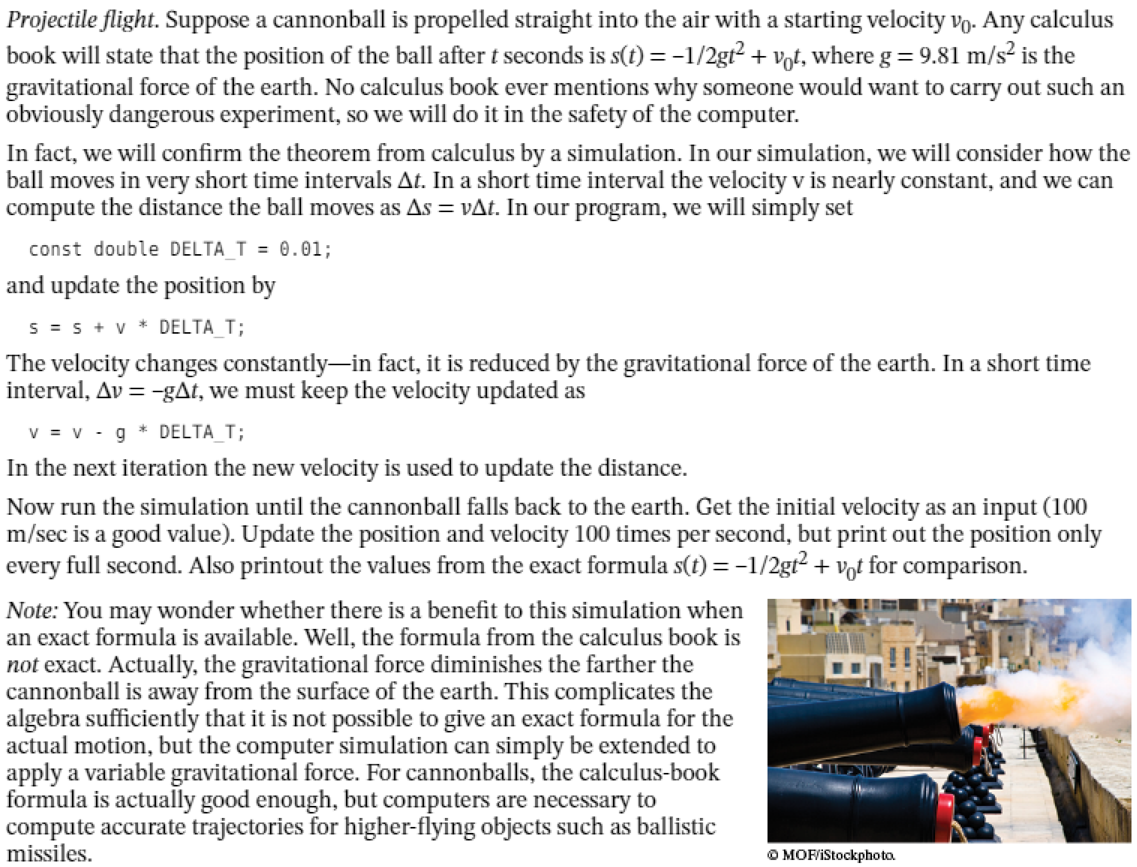
Functions

- int main ()

Variables

- const double DELTA_T = 0.01
- const double GRAVITY = 9.81

2.1.1 Detailed Description



Projectile flight. Suppose a cannonball is propelled straight into the air with a starting velocity v_0. Any calculus book will state that the position of the ball after t seconds is $s(t) = -1/2gt^2 + v_0t$, where $g = 9.81$ m/s^2 is the gravitational force of the earth. No calculus book ever mentions why someone would want to carry out such an obviously dangerous experiment, so we will do it in the safety of the computer.

In fact, we will confirm the theorem from calculus by a simulation. In our simulation, we will consider how the ball moves in very short time intervals Δt. In a short time interval the velocity v is nearly constant, and we can compute the distance the ball moves as $\Delta s = v\Delta t$. In our program, we will simply set

```
const double DELTA_T = 0.01;
```

and update the position by

```
s = s + v * DELTA_T;
```

The velocity changes constantly—in fact, it is reduced by the gravitational force of the earth. In a short time interval, $\Delta v = -g\Delta t$, we must keep the velocity updated as

```
v = v - g * DELTA_T;
```

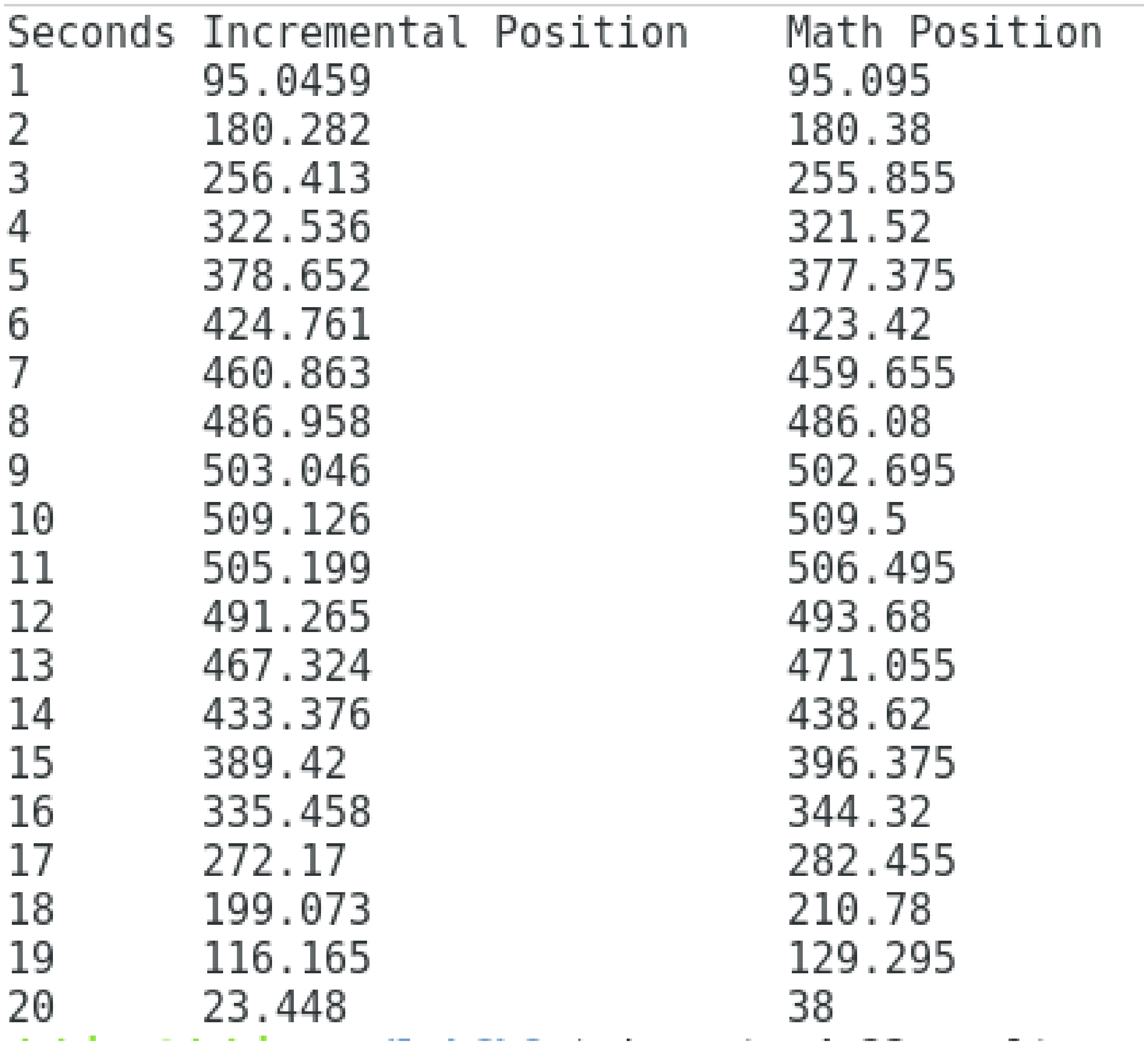
In the next iteration the new velocity is used to update the distance.

Now run the simulation until the cannonball falls back to the earth. Get the initial velocity as an input (100 m/sec is a good value). Update the position and velocity 100 times per second, but print out the position only every full second. Also printout the values from the exact formula $s(t) = -1/2gt^2 + v_0t$ for comparison.

Note: You may wonder whether there is a benefit to this simulation when an exact formula is available. Well, the formula from the calculus book is *not* exact. Actually, the gravitational force diminishes the farther the cannonball is away from the surface of the earth. This complicates the algebra sufficiently that it is not possible to give an exact formula for the actual motion, but the computer simulation can simply be extended to apply a variable gravitational force. For cannonballs, the calculus-book formula is actually good enough, but computers are necessary to compute accurate trajectories for higher-flying objects such as ballistic missiles.

© MOF/iStockphoto.

Figure 1 Projectile Flight



```
Seconds Incremental Position    Math Position
1       95.0459                 95.095
2       180.282                 180.38
3       256.413                 255.855
4       322.536                 321.52
5       378.652                 377.375
6       424.761                 423.42
7       460.863                 459.655
8       486.958                 486.08
9       503.046                 502.695
10      509.126                 509.5
11      505.199                 506.495
12      491.265                 493.68
13      467.324                 471.055
14      433.376                 438.62
15      389.42                  396.375
16      335.458                 344.32
17      272.17                  282.455
18      199.073                 210.78
19      116.165                 129.295
20      23.448                  38
```

Figure 2 Projectile Flight

2.1.2 Function Documentation

2.1.2.1 main()

```
int main ( )
```

Definition at line 11 of file projectile_flight.cpp.

```
12 {
13     double initial;
14     cout << "Input value for initial velocity: " << endl;
15     cin >> initial;
16 
17     double velocity = initial;
18     double iPosition = 0.0;
19     bool air = true;
20     double mathPosition = 0.0;
21     int count = 1;
22     cout << "Seconds\tIncremental Position\tMath Position\n";
23     while (air)
24     {
25         for (double t = count - 1; t < count; t+= DELTA_T)
26         {
27             velocity = velocity - GRAVITY * DELTA_T;
28             iPosition = iPosition + velocity * DELTA_T;
29         }
30         if (iPosition < 0)
31             air = false;
32         else
```

```
        {
            cout << count << "\t"<< iPosition << "\t\t\t";
            mathPosition =  -(GRAVITY * count * count) / 2+ initial * count;
            cout << mathPosition << endl;
        }
        count++;

    }
    return 0;
}
```

2.1.3 Variable Documentation

2.1.3.1 DELTA_T

```
const double DELTA_T = 0.01
```

Definition at line 9 of file projectile_flight.cpp.

2.1.3.2 GRAVITY

```
const double GRAVITY = 9.81
```

Definition at line 10 of file projectile_flight.cpp.

Index