Engr 123 **Assigned: February 6, 2017**
Programming Assignment 4 **Due: February 20, 2017**

Reminder: This is a programming project, and work on this assignment should be done individually. Assistance from other students is limited to questions about specific issues as noted in the syllabus.

Bouncing Ball Gravity
For this problem we will simulate a bouncing ball on the console screen using console graphics. (See *Console Graphics Notes* on the website.) The result of running your program should be similar to that shown in Figure 1.

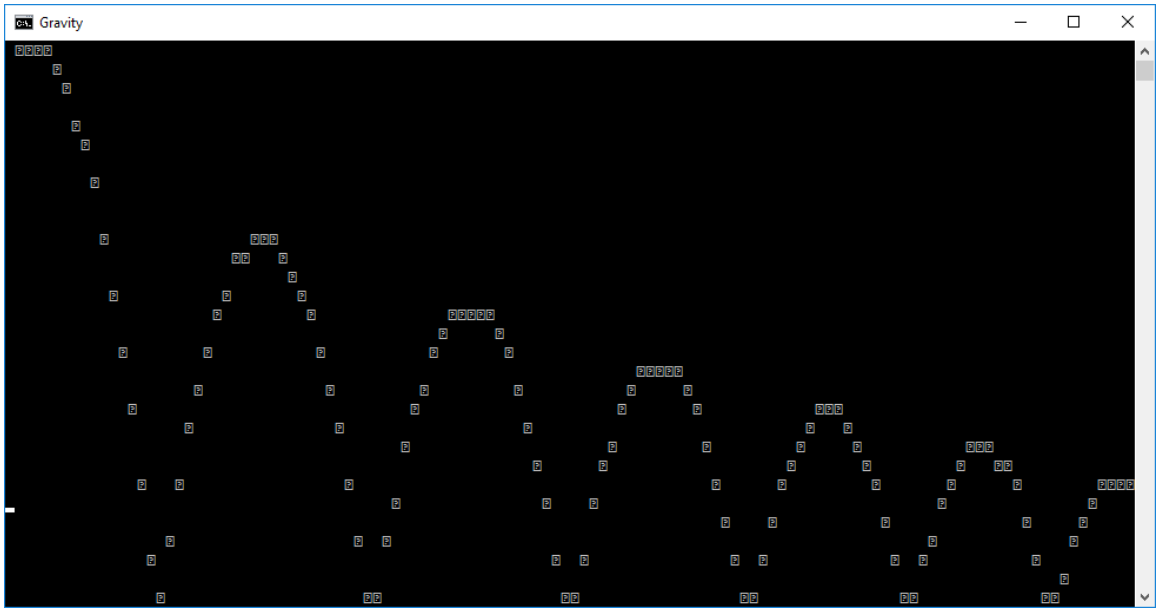

Figure 1
Bouncing ball simulation. The first bounce is a half-parabola and the following bounces are full parabolas.

Note that the top left corner of the screen is column 0, row 0 and the bottom right corner is `Console.WindowHeight -1`, `Console.WindowWidth -1`. So that *x* the column number gets larger going left to right but *y*, the row number, gets larger going top to bottom.

For each iteration the *x* position increases by 1 unit – the horizontal velocity and the *y* position increases according to the laws of gravity. $y = v_0 t + (1/2) g t^2$. For this problem you can increase *t* by 100 milliseconds for each iteration. The variable v_0 is the initial velocity in the *y* direction.

The calculation of the first half parabola goes like this:

```
double x=0, y=0, t=0, v0 = 0;
double g = 32.2;
while(y < window height)
  {Display the ball at position x, y
   x += 1;
```

```
  t = t + 0.1;
  y = v0*t + g*t*t/2;
  wait for 100 milliseconds
 }
```

When this loop completes *y* will be greater than or equal to the window height. You will need to calculate the final velocity of the ball when it reaches the bottom of the window. Since the initial velocity was zero the final velocity is given by:

$$v_f = \sqrt{2g(windowheight - 1)}$$

When the ball hits the bottom of the window it bounces upward but in the process it loses some energy. The new initial velocity will be

$$v_i = -rv_f$$

Where *r* is a coefficient of restitution. For this problem we will take the coefficient of restitution to be 0.9 (or you can input it from the user). The minus sign is due to the change in direction. You can use a loop similar to that shown above to calculate the next full parabola.

Keeping track of time

To make this all happen in approximately real time you can add the following into your loop:

```
Thread.Sleep(100);
```

To use this function you will need to add the threading library:

```
using System.Threading;
```

The delay time produces is the argument of the function in milliseconds.

Your program MUST be modular - that is, it should consist of a main program that is largely a sequence of function calls.

After you get your program running correctly, right click on the *project folder* and choose Send To → Compressed zip file. Rename the compressed zip file as Asn04XXX.zip where XXX are you three initials. Upload the renamed file to \\cecsfp01\users\everyone\engr123.