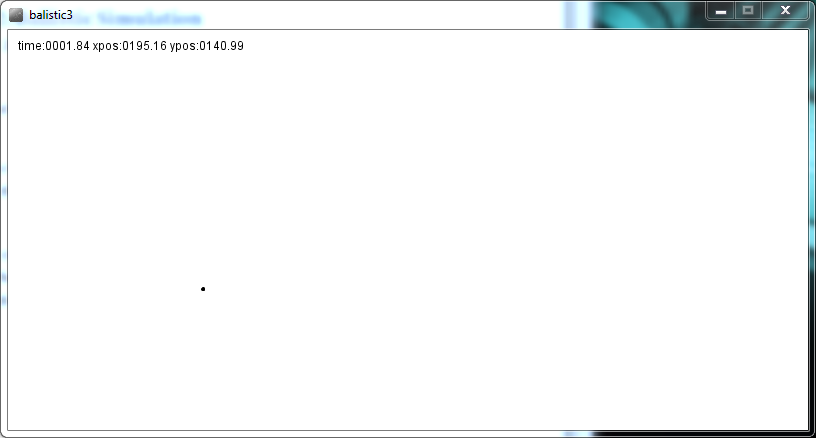
# Processing Assignment 2: Ballistic Simulation

Assignment Due 10/28

## Description

The goal of this project is to construct a Processing program that simulates a ballistic projectile as demonstrated in class during the Wends/Friday labs.



Students are encouraged to work within their teams. The students designated as Development Leads (DL) in the team rosters are to help the other students on their teams understand how the game’s programming works and come up for ideas to improve the game together.

## Calculating the Ballistic Trajectory

The formulas used to calculate the x,y coordinates of the projectile were found:  
<http://www.cefns.nau.edu/~edo/Classes/CS122_WWW/Programs/proj2.html>

From this source three formulas were used:

Total simulation time: maxT = (2×v×sinθ) / g

X Position at time t: xt = v×t×cosθ

Y Position at time t: yt = v×t×sinθ - 1/2g×t2

Where:

V = the projectile velocity e.g. 150.

G = Gravity constant e.g. 32.

Θ = the initial angle of the projectile in radians e.g. 0- 1.57079633. Note I specified Θ in degrees and converted to radians using Processing’s radians() function. I would suggest using 45 degrees (0.785398163) for initial development.

## Expressions in Processing

The following Processing code expressions represent the three formula given above:

maxT is the duration of the projectile’s flight i.e. it is the duration of the simulation.

maxT = 2\*v\*sin(angle)/g;

xpos and ypos are the x and y location of the projectile on the screen at time ‘currT’. Remember that the variables ‘angle’ and v (velocity) are fixed when the simulation starts. It is the simulation time (currT) that is advanced for each step in the simulation i.e. each time the draw() function is called.

xpos = v\*currT\*cos(angle);

ypos = (v\*currT\*sin(angle)) - .5\*(g\*(sq(currT)));

## Psudocode

Like most Processing programs, this program is spread between two functions ‘setup()’ and ‘draw()’. The following is a pseudo code presentation of what your programs need to do. What is given will not execute until you provide the missing functionality.

The setup() function will be called when the program is started i.e. the run button is pressed. It sets the simulation time to zero and initializes the total running time according to the first formula.

setup() {

currT = 0;

maxT = 2\*v\*sin(angle)/g;

}

The draw() function is called repeatedly until the simulation is ended. Each time the function is called, the game display is redrawn with the projectile at a new position as determined by currT.

draw() {

// Advance currT

**currT** = currT + deltaT;

// Test if the simulation should be stopped

if(**currT** > maxT) {

noloop(); // Processing function that halts Processing

}

// Calculate the new x & y position based on currT

xpos = v \* **currT** \* cos(angle);

ypos = (v \* **currT** \* sin(angle)) - 0.5 \* (g \* (sq(**currT**)));

// Draw the current information in the window

drawPositoin(x,y,currT);

// Draw the ellipse in the window

drawShot(x,y);

}

The drawPosition() function draws the text string in the game window. A key feature of this function is that the formatting of the three floating point variables. The Processing function ‘nf()’ stands for Number Format and its arguments are the number to format for printing, the number of decimal places before the decimal point (whole) and the number of places behind the decimal point (fractional). The Processing function ‘int()’ removes the fractional portion of a floating point number and returns the whole number to be printed.

drawPosition() {

String text = "time:" + **nf(currT, 4, 2)**

+ " xpos:" + **int(xpos)** + " ypos:" + **int(ypos);**

text(text, 10, 20);

}

The drawShot() function draws the shot (projectile) in the game window at x,y by simply drawing a filled circle. A key feature of this function is that we draw the elipse at the position ( x, height-y). Subtracting the y position from the window’s height causes the ellipse (projectile) to start its flight from the bottom of the window and progress towards the top. Without this subtraction, the ellipse will start is flight at the top of the window and move down.

drawShot() {

ellipse(x, **height-y**);

}

## Deliverables

There will be only one program submission per team. Teams will submit their sketchbook directory containing their program on a USB thumb drive. Just copy the sketchbook directory containing your program onto the USB thumb drive. THE THUMB DRIVE MUST CONTAIN THE SKETCHBOOK FOLDER TO BE GRADED.

Also include a README.txt file on the USB drive containing your section and team numbers.   
**No File..No Grade.**

Graders will evaluate the submissions by executing the submitted program file and ensuring that it executes correctly.