ECE 4122/6122 Lab #3

(100 pts)

Section	Due Date
ECE 4122 - A	Oct 24th, 2021 by 11:59 PM
ECE 6122 - A	Oct 24th, 2021 by 11:59 PM
ECE 6122 – Q, QSZ, PDO	Oct 24th, 2021 by 11:59 PM

ECE 4122 & **ECE 6122** students do all problems.

Note:

You can do all your code development locally on your personal computer or you can use pace-ice.

Submitting the Assignment:

Create a zip file that contains all your code and the necessary files and folders to compile and build your code. You can submit either a narrated video of you playing the game showing all the game functionality or a link to the video. If you do not submit a video then you need to make sure that your code builds and runs on pace-ice. You should include a CMakeLists.txt file that the TAs can use to build your code.

I have included a zip file with all the images and dlls for use on a windows machine.

Grading Rubric

If a student's program runs correctly and produces the desired output, the student has the potential to get a 100 on his or her homework; however, TA's will **randomly** look through this set of "perfect-output" programs to look for other elements of meeting the lab requirements. The table below shows typical deductions that could occur.

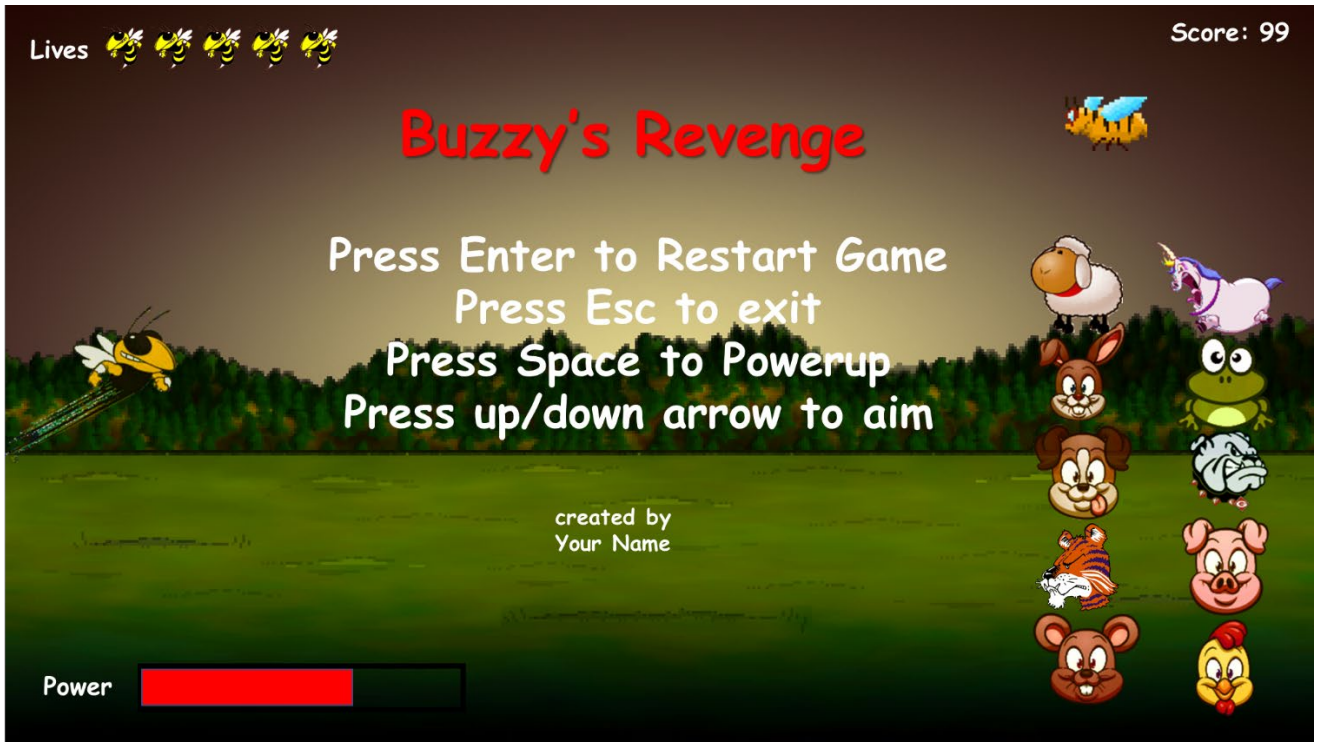
AUTOMATIC GRADING POINT DEDUCTIONS PER PROBLEM:

Element	Percentage Deduction	Details
Does Not Compile	30%	Code does not compile on PACE-ICE!
Does Not Match Output	10%-90%	See points below.
Clear Self-Documenting Coding Styles	10%-25%	This can include incorrect indentation, using unclear variable names, unclear/missing comments, or compiling with warnings. (See Appendix A)

LATE POLICY

Element	Percentage Deduction	Details
Late Deduction Function	score – H*0.5	H = number of hours (ceiling function) passed deadline .

Problem 1: Buzzy's Revenge (100 pts)



For this assignment you will be creating a derivation of "Angry Birds" called "Buzzy's Revenge" using the SFML API. In this game, the user launches buzzy across the field attempting to hit the evil mascots from other universities, while avoiding the cute woodland creatures. The game plays out as follows:

1. (10 pts) Your application needs to create the opening screen shown above and have the game start when the user hits the "Enter" key.
2. (5 pts) The "space" key is used to increase buzzy's initial velocity. As the user holds down the key the power indicator advances. You need to determine the minimum and maximum initial velocity that buzzy will have when launched when the space key is released. You need to setup the game's dimensions so that buzzy takes 1-4 secs to travel across the field.
3. (10 pts) Buzzy's initial launch angle is increased and decreased using the "up" and "down" arrow keys. The buzzy image needs to be rotated to indicate the changing launch angle.
4. (10 pts) Buzzy flight across the field follows a simple projectile motion path.

Projectile Motion Equations

Vertical Direction

$y_f = y_i + v_{iy}t + (1/2)a_y t^2$

$y_f = y_i + (t/2)(v_{iy} + v_{fy})$

$v_{fy} = v_{iy} + a_y t$

$v_{fy}^2 = v_{iy}^2 + 2a_y(y_f - y_i)$

$(a_y = -9.8 m/s^2)$

Horizontal Direction

$x_f = x_i + v_{ix}t$

$v_{fx} = v_{ix}$

Auxiliary Equations

$v_{ix} = v_i \cos\theta$
$v_{iy} = v_i \sin\theta$

5. (5 pts) The rotation angle of buzzy should change as he fly across the field.
6. (5 pts) When buzzy hits one of the evil mascots (shown below) you get 25 pts and that column disappears


7. (5 pts) When both mascots are hit the level is recreated.
8. (10 pts) Creating the columns.
    a. Only one evil mascot should be in each column.
    b. The vertical placement of the evil mascots should randomly change each time the level is recreated.
    c. The placement of all the other wood land creatures should also change each time the level is recreated.
9. (5 pts) The bee () should fly from right to left along a random/changing path. If buzzy is able to intercept the bee then the user gets 75 extra points. The bee disappears until the level is recreated.
10. (5 pts) If the user is able to hit the mad unicorn () then the user gets an extra life up to a maximum of 5 lives. When hit the mad unicorn disappears and all the images above drop down by one.
11. (5 pts) Buzzy uses up a life when he is launched and does not hit either
    a. An evil mascot
    b. The bee
    c. The mad unicorn
12. (5 pts) If buzzy hits a woodland creature he immediate reappears back at his launch point.
13. (5 pts) If buzzy does not hit anything then he immediate reappears back at his launch point when the center of the buzzy image leaves the screen.
14. (5 pts) The game ends when buzzy uses up all his lives.
15. (5 pts) Each time a life is lost one of the life images ()should disappear.
16. (5 pts) The score needs to be updated when points are won.

Extra Credit:

- Up to 6 points extra credit for adding animation to buzzy hitting objects and sound effects (TA's discretion)
    - (2pts) If buzzy hits a woodland creature then he falls back spinning to the bottom of the screen before reappearing.
    - (2pts) Instead of just having a column disappear, have the images fall spinning to the bottom of the screen before disappearing.
    - (2pts) Add sound effects.

Appendix A: Coding Standards

Indentation:

When using *if/for/while* statements, make sure you indent 4 spaces for the content inside those. Also make sure that you use spaces to make the code more readable.
For example:

```
for (int i; i < 10; i++)
{
    j = j + i;
}
```

If you have nested statements, you should use multiple indentions. Each { should be on its own line (like the *for* loop) If you have *else* or *else if* statements after your *if* statement, they should be on their own line.

```
for (int i; i < 10; i++)
{
   if (i < 5)
   {
       counter++;
       k -= i;
   }
   else
   {
       k +=1;
   }
   j += i;
}
```

Camel Case:

This naming convention has the first letter of the variable be lower case, and the first letter in each new word be capitalized (e.g. firstSecondThird). This applies for functions and member functions as well! The main exception to this is class names, where the first letter should also be capitalized.

Variable and Function Names:

Your variable and function names should be clear about what that variable or function is. Do not use one letter variables, but use abbreviations when it is appropriate (for example: “imag" instead of “imaginary”). The more descriptive your variable and function names are, the more readable your code will be. This is the idea behind self-documenting code.

File Headers:

Every file should have the following header at the top

```
/*
Author: <your name>
Class: ECE4122 or ECE6122
Last Date Modified: <date>

Description:

What is the purpose of this file?

*/
```

Code Comments:

1. Every function must have a comment section describing the purpose of the function, the input and output parameters, the return value (if any).
2. Every class must have a comment section to describe the purpose of the class.
3. Comments need to be placed inside of functions/loops to assist in the understanding of the flow of the code.