

The Ultimate Shooting Game

Name | Object Oriented Analysis and Design | 22/11/17

TABLE OF CONTENTS

* System Requirements…………………………………….…2
* How to start the game………………......………… .2
* Controls………………………………………………………..3
* Starting the game…………………………………………..4
* About game objects…………………………………………5

## About the working of the game……………………10

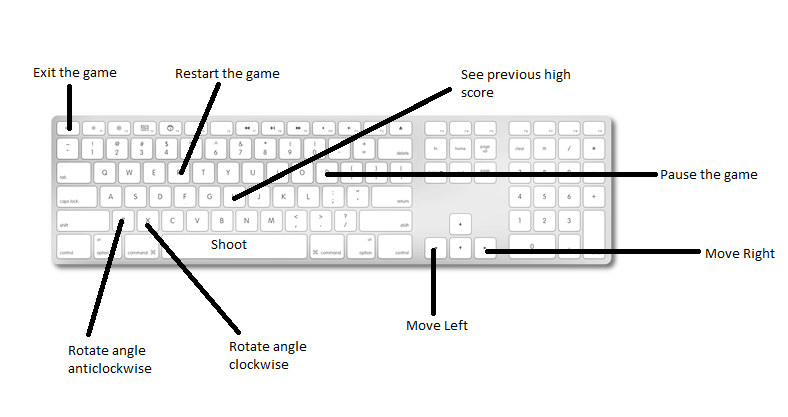
System Requirement

* To Compile the project you need incule graphics.h and and conio.h(if compiling with g++ compiler).

How to start the game

* **Windows Users :** Simply double click the .exe file.
* **Linux Users :** You need to install windows campatiblity in linux i.e **wine** and then Start the .exe file by writing “wine Ultimate\_shooting\_game.exe” in the directory of the game (in terminal).

Controls



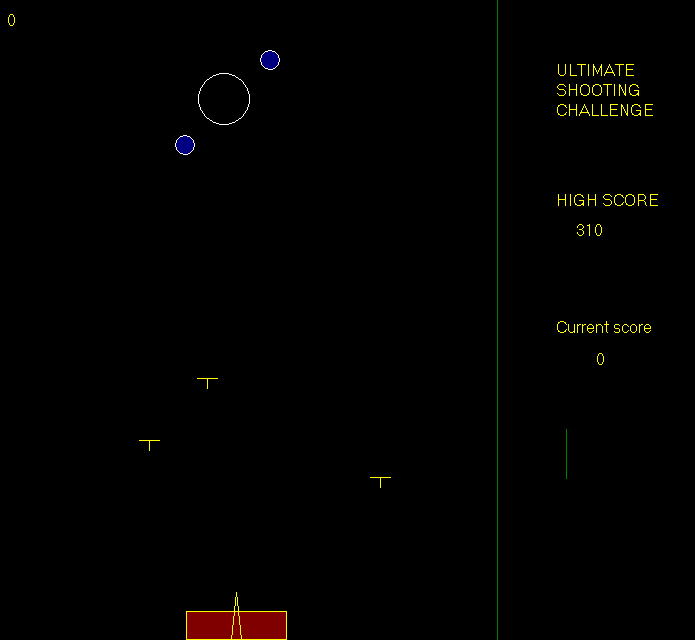
|  |  |
| --- | --- |
| <- (left arrow key) | Move left |
| -> (right arrow key) | Move right |
| * Z | Rotate angle anti-clock wise |
| X | Rotate angle clock wise |
| Space | Shoots bullets |
| Esc | Exit the game |
| P | Pause the game |
| R | Restart’s the game |

STARING THE GAME

* First screen will be this you have to press space to go to next screen



* Next screen will contain the instructions of the game.
* After that you have to enter your name so that if you break any previous high score we will save your high score in our database.
* After this screen your game will be started.



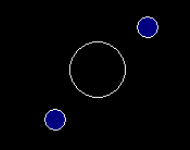
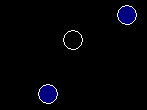
* Health Bar & the Shooter tank

1. This Tank is actually you, you have to move this and protect this.
2. The triangle represents your gun you can rotate it as you like you have to shoot at the circles genrating randomly on the screen using this gun.
3. The red bar represets your health
4. Initially you have 7 health means you can take upto 7 damage from bullets.
5. After getting certain amount of score level will be changed and you health bar will get full and then you can get upto 14 damage from the bullets.
6. As the damage increases your red bar will Starts decreasing.

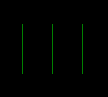
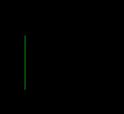
# Randomly generating circles

* This object will be generated on your screen randomly anywhere you have to shoot the inner bigger circle before it disappears
* Radius of the inner circle will be decreasing at some rate after which it will disappear and your life will decrease.

* Those 2 blue circles will be rotating in opposite direction to each other around the inner circle and these two will be an obstacle for you to shoot the inner circle.

# Player’s Life

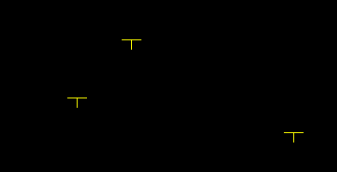
* Initially player will have 3 lifes as you as game goes if you are unable to shoot the target circle your life will decrease.
* If you miss more than 3 circles you will not have any life lift so you will lose.

# level change

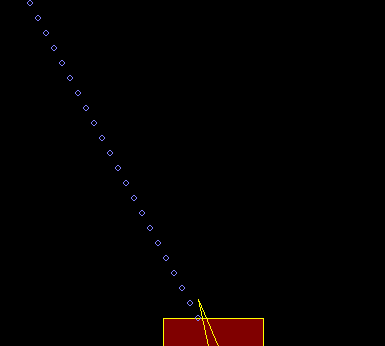
* As you break certain amount of score the level will get change speed of bullets falling from about will increase and color of the bullets will change from yellow to red and your health will increase from 7 to 14.

# falling bullets

* These Bullets are the real enemy of you in this in this game.
* If you got hit by them them your health will decrease.
* As the level change speed of their falling will increase
* At a time, maximum 4 bullets can fall.

# The gun



* The weapon you have in this game is this gun. It shoots circle in range from 0 to 180 degree.
* You have to shoot the randomly genrating target(circles) with this gun.

# pause:

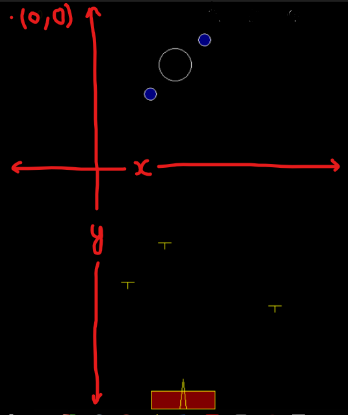
To pause the game at any movement press ‘p’ and to resume press ‘p’ again.

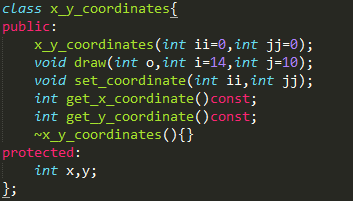
## Game over

* To restart the game press ‘r’ and to see high score press h and to exit press esc.

How the game works?...

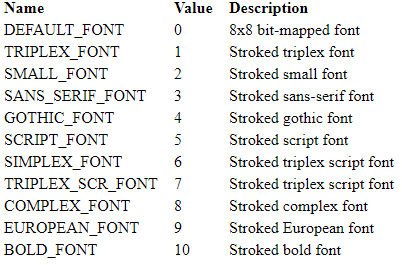
* At first there exists a base class that is x\_y\_coordinates, which contains and maintains all the x and y coordinates of the game.
* Origin of the screen is at top left point of the screen.





# Basic functions which we have used from graphcs.h

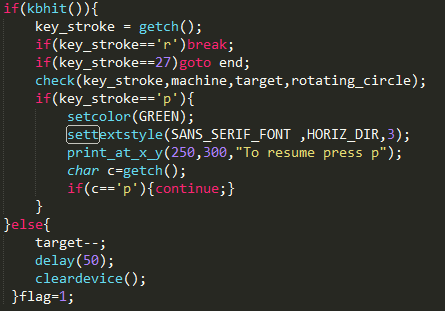
1. Outtextxy(x\_coordinate,y\_coordinate,char\* textstring): It silmpy prints the given string to to given coordinates.
2. Settextstyle(int font,int direction,int charsize): It decides to font of the string which we like to print.



1. Setcolor(color name) : It decide the color of the whatever printing after this function line.
2. Circle(x\_center\_coordinate,y\_center\_coordinate,radius): make circle at at provided center.
3. Line(x1,y1,x2,y2): make line from (x1,y1) to (x2,y2).
4. Delay(time(ms)): stop the code execution for that much milli seconds.
5. Getch(): saves char in its stream.
6. Cleardevice(): clear the whole screen .
7. Initwindow(): creates a window on which we work in graphics.

# why we have used conio.h?

we include conio.h just to use only one of its basic function and that is kbhit() and its work is to return true if any key is pressed.

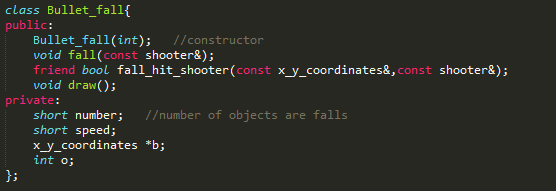


# how the shooter tank is moving horizontally and shooting bullets at different angels

# 

* Here shooter tank is derived class of x\_y\_coordinate as all the x,y coordinates are maintained by x\_y\_coordinate class.
* set\_s\_keys and set\_keys set the gun and tank movement keys.
* get.. returns left ,right keys of the tank.
* get\_s.. returns left and right of gun.
* get\_h\_key return x coordinate of top of the gun.
* Here prefix overloading is for angle movement.
* Here postfix overloading is for shooter tank moving horizontally.

# how bullets are falling?



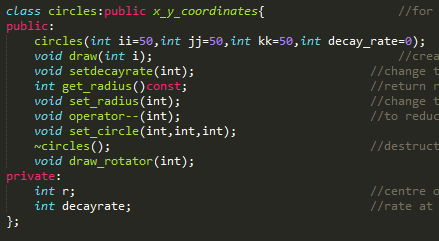
* here fall() is responsible for bullets to fall and for checking simultaneously that the tank is hitted by the bullet or not by calling the friend function fall\_hit\_shooter.
* draw() make bullets.
* We can also control the speed of the bullets falling.

# how we are checking does the bullets hit the tank or not?

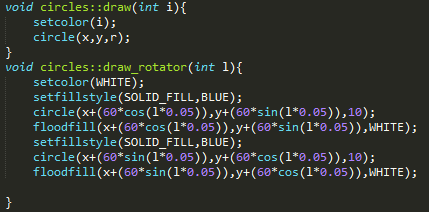
We simply check whether x coordinate of bullet is in range of length of x coordinate of the tank and greater than y coordinate of the top of tank.



# how circles are generating at random place and how gun is shooting and those rotating circles?

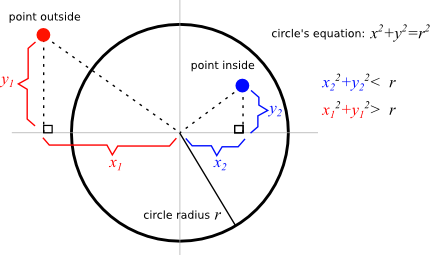


* As discussed above all the x,y coordinates are maintained by x\_y\_coordinates class that’s why circles is also derived class of this base class.
* Here we can set decay rate of the circle using setdecayrate() i.e how fast circle radius will decrease.
* get\_radius returns current radius of the circle.
* set\_radius sets radius to a circle.
* Postfix – operater is overloaded to reduce the radius of the circle.
* Set\_circle sets coordinates to circle i.e random coordinates
* ~circles() is distructor of the circle it makes radius of the circle to 0
* draw\_rotator() draws the rotating circles, takes int l in argument to continuously rotate and increase the angle of rotation.
* draw draws the circles and gun also use this same function to shoot the circles.



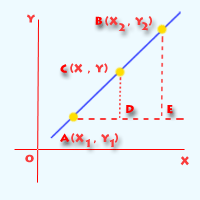
# how we are checking gun has shooted the target circle or not?

* Shooter is actually shooting circle whose center coordinates are increasing in straight line.
* So if distance from centre of these shooted circles to the target circle is less than that radius of the target circle than the shooted circle lies inside that of target circle so in other way target has been shooted so in that case distructor of the target circle has been called and score will increase we will see later how score is changing.



* Same concept has been used when the gun shoots the rotating circle but in this case destructor of the shooting circle has been called because these rotating circles made to be solid shooting circle can’t pass these rotating circles.

# how we are shooting at different angles?



In our case y2=y1 and we are only changing x2 with help of postfix ++ and – overloading in shooter class we have discussed earliear.

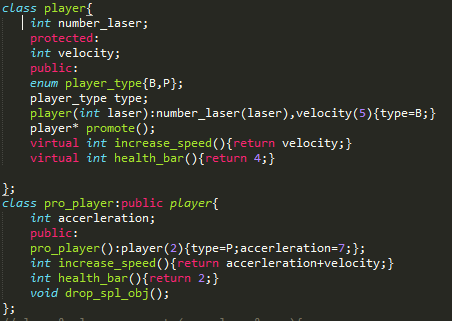
So by using 2 point form we get a direction for shooting

2 point form:

(y-y1)=((y2-y1)/(x2-x1))(x-x1)

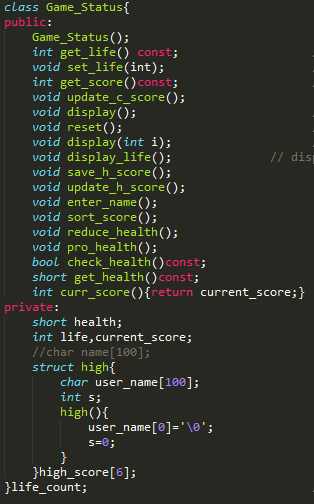
# how level of the game is changing?

* Run time polymorphism has been used to do this.
* Here we have made 2 classes to this.



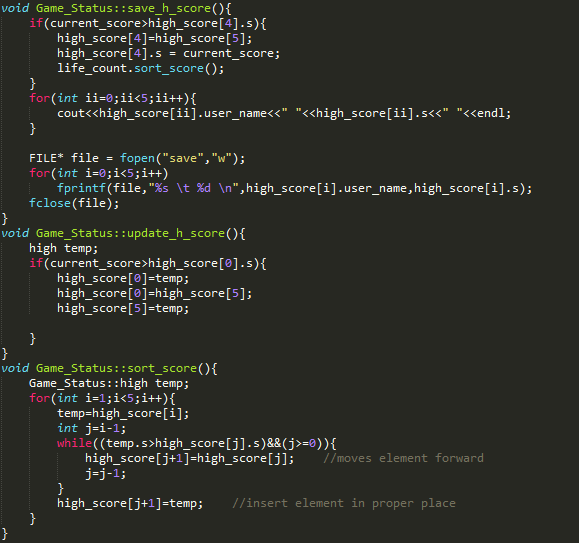
* At the beginning of the game we have pointer p to the player class or we call that player initially a beginner.
* So initially all the function like speed, health bar are used of that of player class as pointer p is pointing to player classs.
* As player breaks certain amount of score pointer p will start pointing to an object of pro\_player class or player has become a pro-player.
* So now pointer p will start using functions of pro\_player class so speed of the bullets wil increase and health bar will also change according to what we want to set.

## how score, health and life has been maintained?



* get\_life(), set\_life() maintains life of the player.
* Reduce\_health(), pro\_health(),check\_health(),get\_health() maintains life bar of the player.
* Update\_c\_score() updates current score continuously.
* Update\_h\_score() updates high score if current score>top score till now.
* Sort\_score sort the scores in the file.
* Save\_h\_score() save the score into the file
* Enter() name used for entering the name of the player at the beginning of the game.

# how score is saved into the file?



# why we have used c++?

Because It is object oriented I could have also used C but C++ being object oriented helped us to deal with complex situation much better than that of C which is not object oriented.

Future plans

# implement this using parallel programming because right know whole game is running in a single loop which makes the game looks flashing.

# special object falling, by shooting that object 5 times health and life of the player increases.

# better levels.

# better score saving method.

bibliography

# en.cppreference.com

# bjarne stroustrup book