Final Project: Zombie Assault

Jacob Westerbeek

December 14, 2018

1 Statement of the Problem

The objective is to write a cool and interesting program using all the knowledge about for loops, classes, sorting, parsing, and everything else we have learned throughout the semester. This program must be "cool" and computationally interesting as well as being a unique idea. I chose to create a shooter game called "Zombie Assault" where the player must defend his character from a wave of zombies without dying.

2 Description of Solution

My program has four main classes, these classes are the circle, player, enemy, and bullet classes. The player, enemy, and bullet classes will all inherit the circle class in order to draw each of these objects as a circle, this is because openFrameworks does not have a circle class.

I started with creating the Circle class. The Circle class was very simple as all I needed to setup were the x, y, and r variables that my other three classes could inherit. Of course I could have just as easily setup these variables in each of the three classes, using this method it makes my program streamlined.

I started with creating the player class and allowing the user to move the player. To allow the player the move, I simply increment the players position by a couple of pixels in the direction of the key the user pressed. In addition, I wanted the player to be able to tell where they are aiming their gun. To do this I used a few of openFrameworks built in functions involving angles. I used the *ofPushMatrix()*, *ofPopMatrix()*, *ofTranslate()*, and *ofRotate()* which allows the player to be rotated about its center. To get the player to aim where the user is aiming, I needed to calculate the angle between the player and the mouse position.

$$\theta = \arctan(\frac{mouseY - playerY}{mouseX - playerX})$$

With the player class finished, I started working on the enemy class. In this class I created a *seek()* which allows the enemy to follow the player and a *hits()* which would check if the player and enemy are intersecting. For the seek function I would need to calculate the vector from the enemy to the player, this can be achieved using simple vector math. First find the distance between the the player and enemy.

$$xDistance = playerX - enemyX$$

$$yDistance = playerY - enemyY$$

Next we find the magnitude of the vector so we can can normalize it.

$$magnitude = \sqrt{(xDistance)^2 + (yDistance)^2}$$

Normalizing the vector,

$$xDistance = \frac{xDistance}{magnitude}$$

$$yDistance = \frac{yDistance}{magnitude}$$

With the finalized x and y distance vectors we can increment the enemies position by the x and y distance and because these calculations are in update, they will constantly be updated based on the enemies new position and the players new position.

Next is the hits function which checks if the player and enemy are intersecting so the enemy can do damage to the player. For this function it needs to know the distance between the enemy and the player. This can be solved with the same equation from the seek function to find the distance between the player and the enemy and then finding the magnitude which we will now label as the distance. This time we will check if the distance is less than the sum of the radii, then it will return true in turn doing damage to the player.

In addition to this, I wanted it to appear as if the zombies were facing towards the player rather than circles following the player. For this, I used the same exact method for finding the angle in the player class only this time the angle is between the enemy and the player rather than the player and mouse.

Finally, in the bullet class I again used many of the same methods from the enemy and player class to allow the user to fire a bullet in there desired direction to do damage to the enemies. In order to allow the users bullet to fire in the direction that they clicked, I needed to find the vector from the player to the mouse position. This follows the same set of equations from the enemy class, instead now with the mouse position rather than enemy position.

$$xDistance = mouseX - playerX$$

$$yDistance = mouseY - playerY$$

$$magnitude = \sqrt{(xDistance)^2 + (yDistance)^2}$$

$$xDistance = \frac{xDistance}{magnitude}$$

$$yDistance = \frac{yDistance}{magnitude}$$

Again, we can then increment the bullets position by the x and y distance which was calculated for wherever the player clicked on the screen.

Finally, in order to damage the enemy I can use the same method from the enemy class to check if the enemy and player are intersecting to instead check if the bullet and enemy are intersecting. We simply find the distance between the bullet and the enemy.

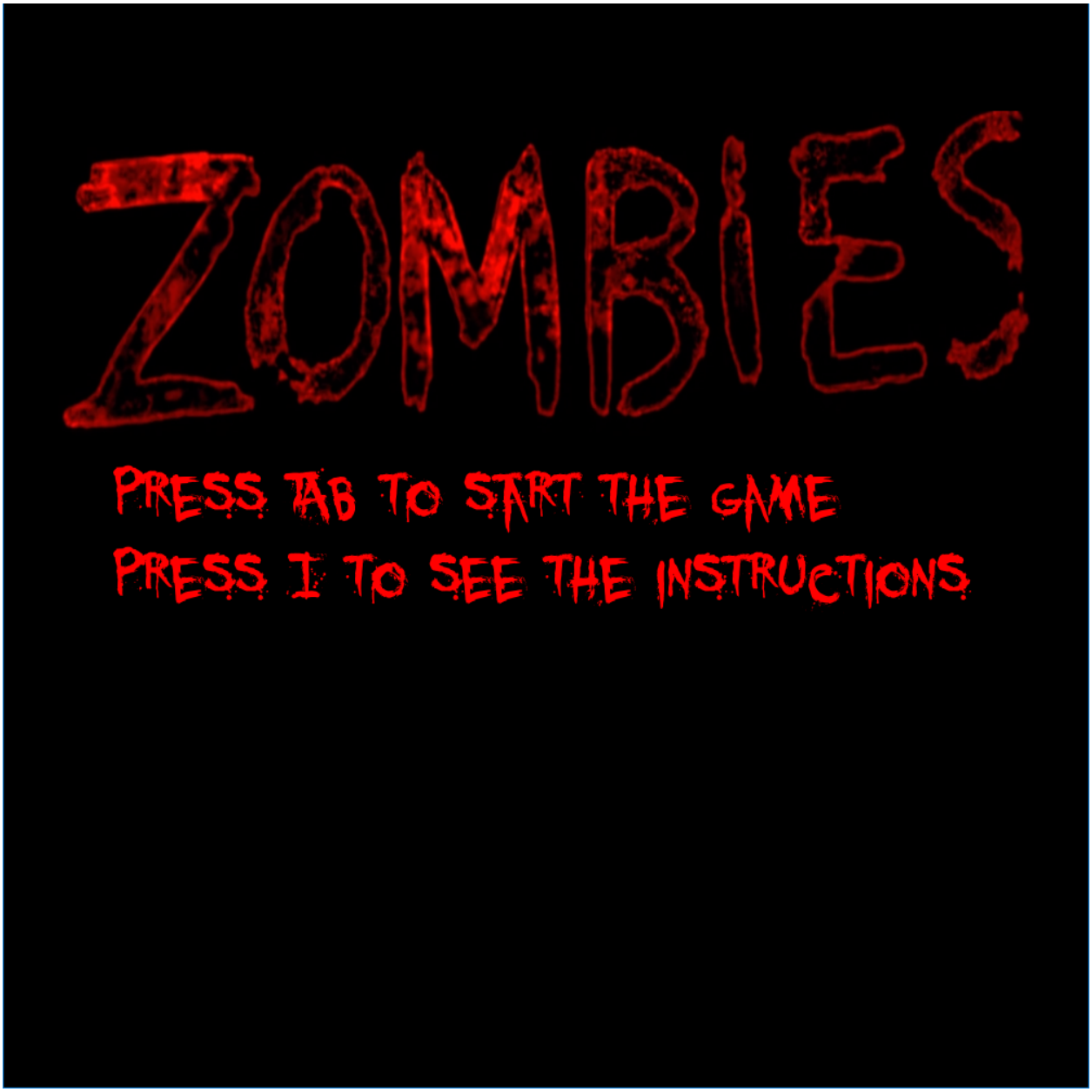
$$distance = \sqrt{(enemyX - bulletX)^2 + (enemyY - bulletY)^2}$$

We again check if the distance is less than the sum of the radii, then we damage the enemy.

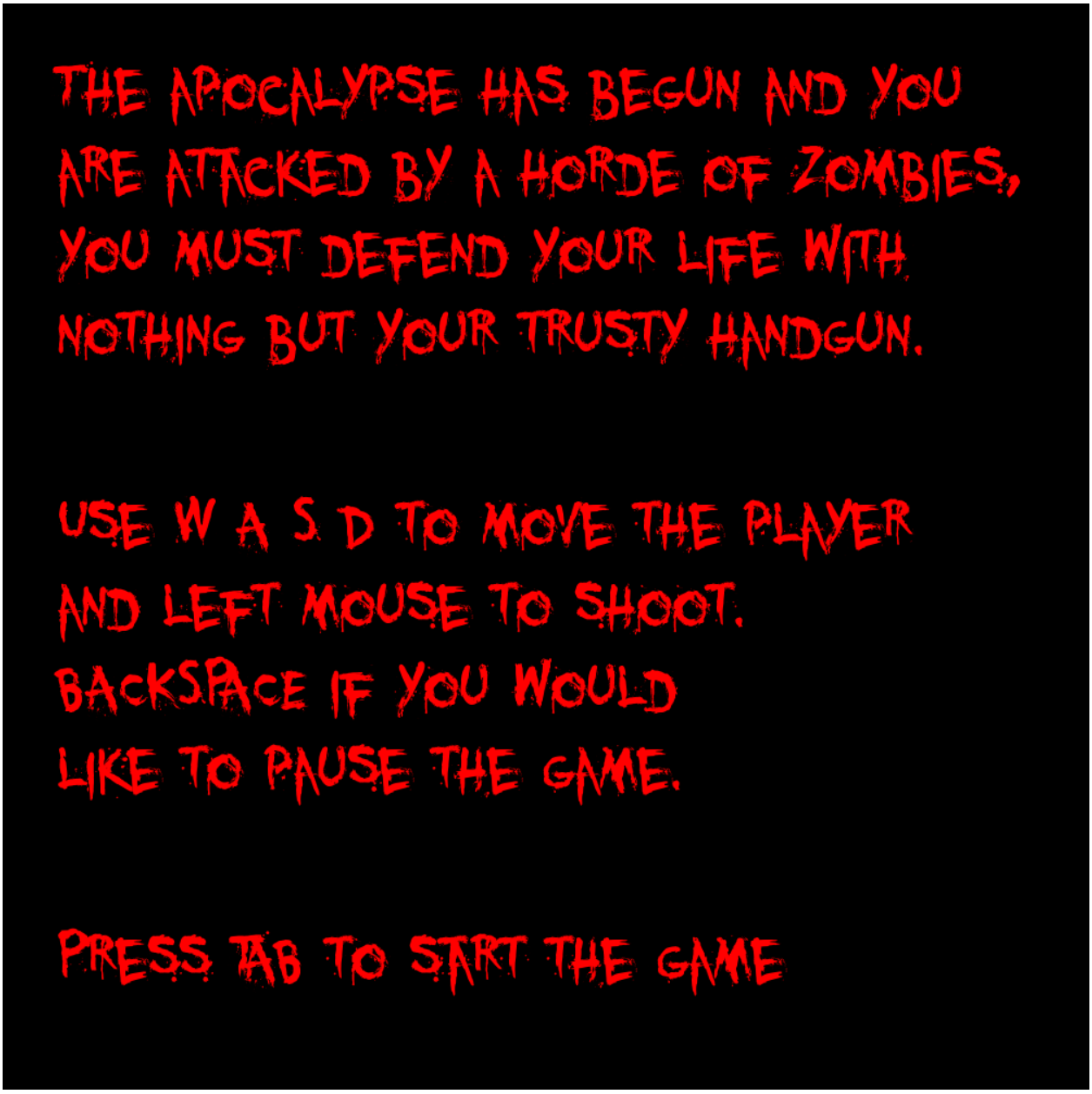
With all of my desired classes complete, I now moved on to call all of my functions in ofApp.cpp in the appropriate locations such as setup, update, draw, keypressed, and mousepressed. As well, to make the game look nicer I created a main menu screen to introduce the player with a graphic and some instructions. Also, whether the user wins or loses they will be shown a screen based on whether they won or lost, with the option to play again.

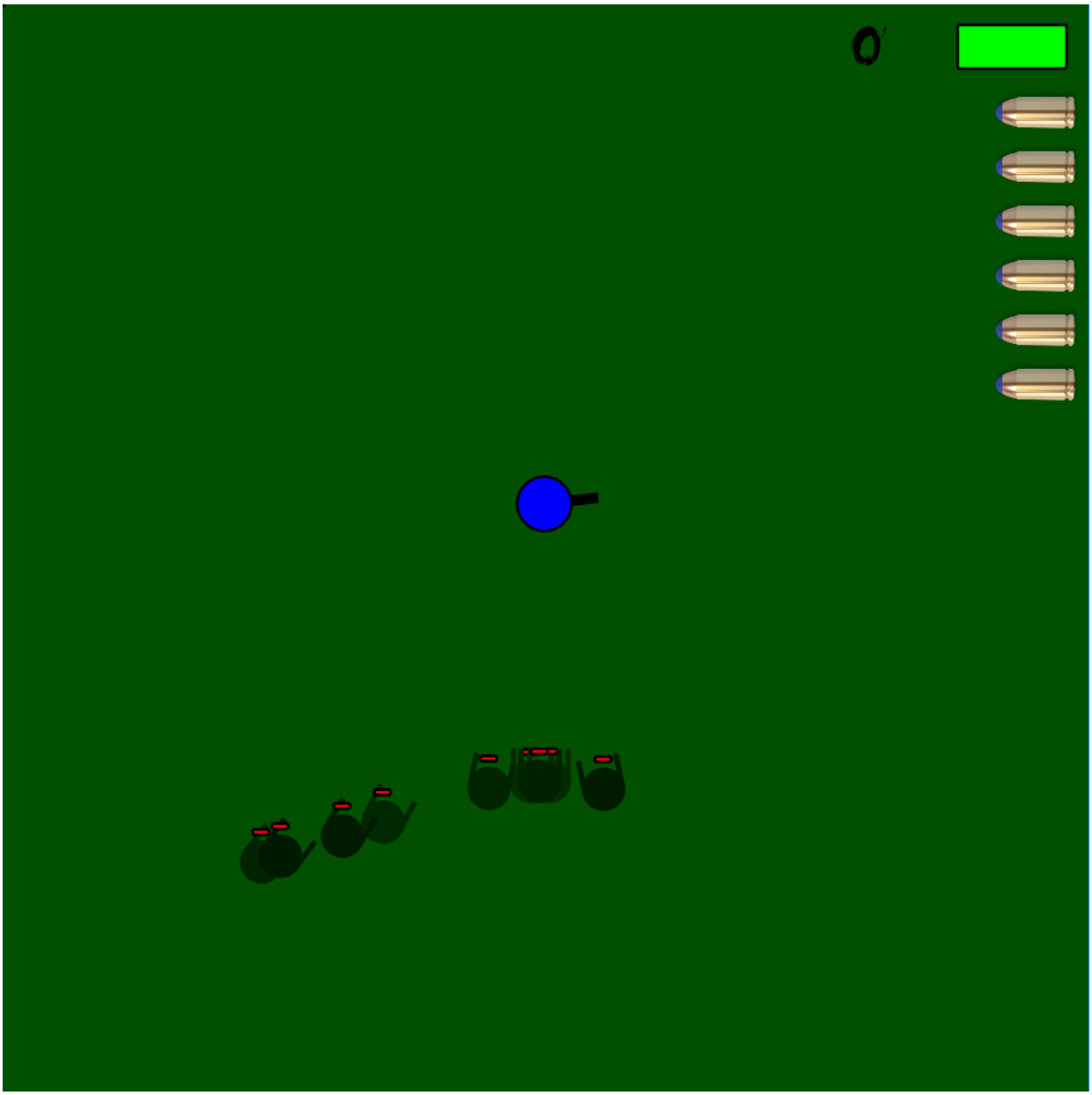
3 Testing and Output

To make sure everything was working correctly, I first only drew the main menu to the screen as well as ensuring the button press to transition the state of the game worked correctly.



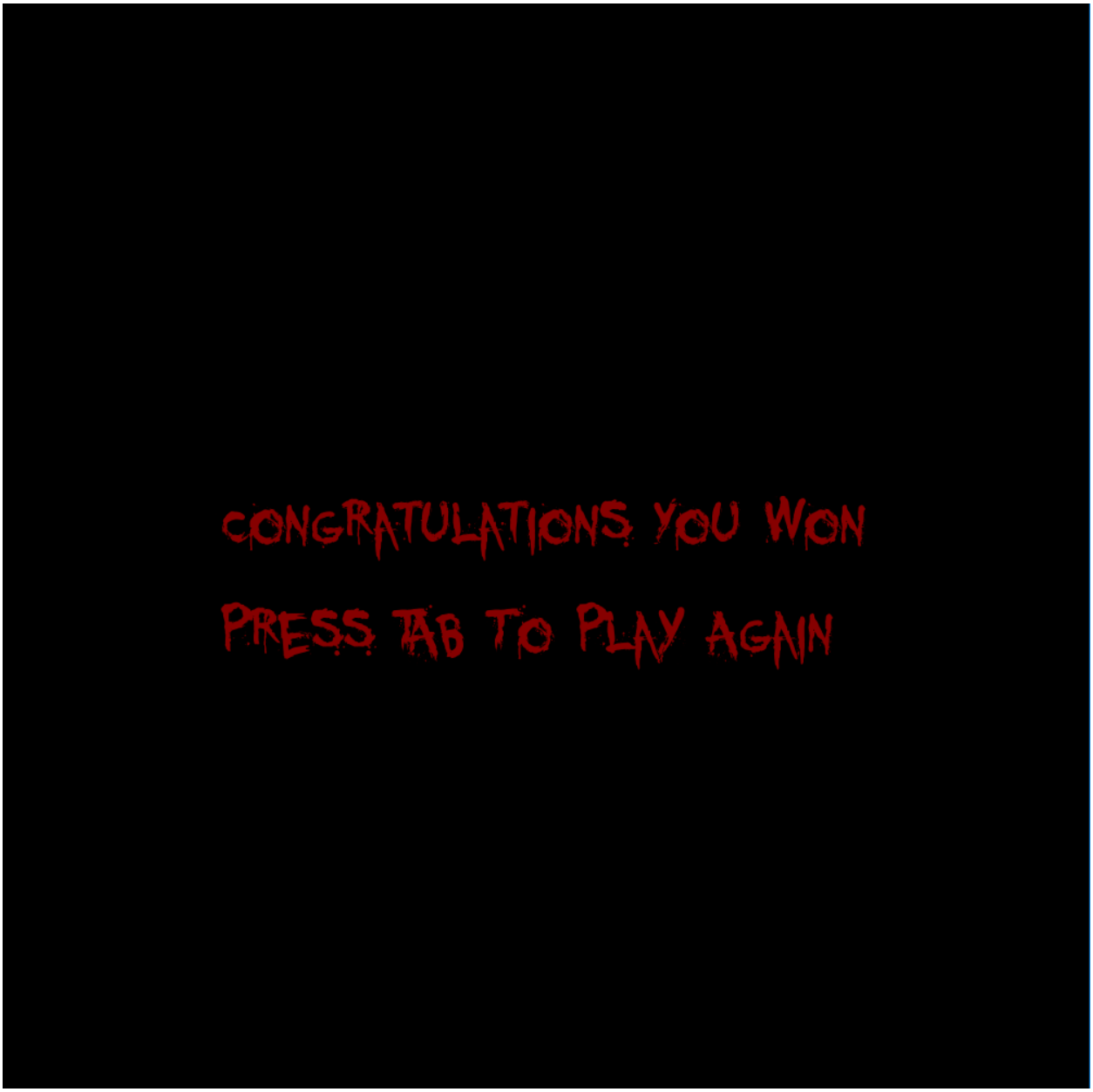
With all of the transitions working correctly we can transition either straight into the game or the user can see the instructions and controls of the game.




0

As well, while they are in game they can pause it using the backspace key.

If the player successfully wins the game by defeating all of the enemies they will be prompted with a win screen.



And if they take too much damage, they will be prompted with a lose screen.

4 Code

```
class Circle
{
public:

   float x;
   float y;
   float r;

   Circle();
   Circle(float x_in, float y_in, float r_in);
};
```

```
#include "circle.h"

Circle::Circle()
{
   x = 0;
   y = 0;
   r = 1;
}

Circle::Circle(float x_in, float y_in, float r_in)
{
   x = x_in;
   y = y_in;
   r = r_in;
}
```

```
class Player : public Circle
{
public:

   ofColor playerColor;

   ofTrueTypeFont scoreFont;

   int health;

   int score;
```

```
    float theta;

    void setup();
    void update();
    void draw();

    void up();
    void down();
    void left();
    void right();
};
```

```
#include "player.h"

void Player::setup()
{
    x = 500;
    y = 500;

    r = 25;

    scoreFont.load("bloody.otf", 60);

    playerColor.set(0, 0, 255);

    health = 100;

    score = 0;
}

void Player::update()
{
    if (health <= 0) health = 0;

    theta = atan2(ofGetMouseY() - y, ofGetMouseX() - x) * 180 / PI;
}

void Player::draw()
{
    //Player

    ofPushMatrix();

    ofTranslate(x, y);

    ofRotate(theta);
```

```
    ofFill();
    ofSetColor(playerColor);
    ofDrawCircle(0, 0, r);

    ofNoFill();
    ofSetLineWidth(3);
    ofSetColor(0, 0, 0);
    ofDrawCircle(0, 0, r);

    ofFill();
    ofSetColor(0, 0, 0);
    ofDrawRectangle(25, -5, 25, 10);

    ofPopMatrix();


    //Player Health Bar
    ofFill();
    if (health <= 100) ofSetColor(0, 255, 0);
    if (health <= 70) ofSetColor(255, 255, 0);
    if (health <= 30) ofSetColor(255, 0, 0);
    ofDrawRectangle(880, 20, health, 40);

    ofNoFill();
    ofSetLineWidth(3);
    ofSetColor(0, 0, 0);
    ofDrawRectangle(880, 20, 100, 40);

    //Player Score
    scoreFont.drawString(to_string(score), 780, 60);
}

void Player::up()
{
    y = y - 10;
}

void Player::down()
{
    y = y + 10;
}

void Player::left()
{
    x = x - 10;
}
```

```
void Player::right()
{
   x = x + 10;
}
```

```
class Enemy : public Circle
{
public:

   int health;

   ofColor enemyColor;

   string isAlive;

   float theta;

   float xDistance;
   float yDistance;
   float magnitude;
   int speed;

   void setup();
   void update();
   void draw();

   void takeDamage();


   bool hits(Circle other);

   void seek(Circle other);
};
```

```
#include "enemy.h"

void Enemy::setup()
{
   isAlive = "alive";

   x = 25 + rand() % 950;
   y = 1050;
   r = 20;
```

```
    health = 15;

    enemyColor.set(0, 25 + rand() % 41, 0);

    speed = 1;
}

void Enemy::update()
{

}

void Enemy::draw()
{
    if (isAlive == "alive")
    {
        //Enemy

        ofPushMatrix();

        ofTranslate(x, y);

        ofRotate(-theta);

        ofFill();
        ofSetColor(enemyColor);
        ofDrawCircle(0, 0, r);

        ofFill();
        ofSetColor(enemyColor);
        ofDrawRectangle(-20, 0, 5, 30);

        ofFill();
        ofSetColor(enemyColor);
        ofDrawRectangle(15, 0, 5, 30);

        ofPopMatrix();


        //Enemy Health
        ofFill();
        if (health <= 100) ofSetColor(0, 255, 0);
        if (health <= 70) ofSetColor(255, 255, 0);
        if (health <= 30) ofSetColor(255, 0, 0);
        ofDrawRectangle(x - 8, y - 30, health, 5);
```

```
      ofNoFill();
      ofSetLineWidth(3);
      ofSetColor(0, 0, 0);
      ofDrawRectangle(x - 8, y - 30, 15, 5);
   }
}

void Enemy::takeDamage()
{
   health = health - 5;
}

bool Enemy::hits(Circle other)
{
   if (isAlive == "alive")
   {
      float sum, dist;

      bool result;

      sum = (x - other.x) * (x - other.x) + (y - other.y) * (y - other.y);

      dist = sqrt(sum);

      if (dist < r + other.r) result = true;
      else result = false;

      return(result);
   }
}

void Enemy::seek(Circle other)
{
   if (isAlive == "alive")
   {
      xDistance = other.x - x;
      yDistance = other.y - y;

      theta = atan2(xDistance, yDistance) * 180 / PI;

      magnitude = sqrt(xDistance * xDistance + yDistance * yDistance);

      xDistance = xDistance / magnitude;
      yDistance = yDistance / magnitude;

      x = x + xDistance * speed;
      y = y + yDistance * speed;
```

```
    }
}
```

```
class Bullet : public Circle
{
public:
    //Varibles
    ofImage bullet;
    ofSoundPlayer reloadSound;
    ofSoundPlayer firedSound;

    int numBullets;

    int vx, vy;

    int isFired;

    float xDistance;
    float yDistance;
    float magnitude;
    int speed;

    //Methods
    void setup();
    void update();
    void draw();

    void fired(Circle other);

    void reload();

    bool hits(Circle other);

    void bulletStop();
};
```

```
#include "bullet.h"

void Bullet::setup()
{
    bullet.load("bullet.png");
    reloadSound.load("reload.mp3");
    firedSound.load("fired.mp3");
```

```
    numBullets = 6;

    r = 5;

    isFired = 0;

    speed = 1;
}

void Bullet::update()
{
    if (isFired = 1)
    {
        x = x + xDistance * speed;
        y = y + yDistance * speed;
    }
}

void Bullet::draw()
{
    //Ammo Counter
    for (int i = 0; i < numBullets; i++)
    {
        ofSetColor(255, 255, 255);
        bullet.draw(850, 50 + 50 * i, 200, 100);
    }

    if (isFired == 1)
    {
        ofSetColor(0, 0, 0);
        ofFill();
        ofDrawCircle(x, y, r);
    }
}

void Bullet::fired(Circle other)
{
    isFired = 1;
    speed = 20;
    numBullets = numBullets - 1;

    if (numBullets >= 0)
    {
        firedSound.play();

        if (isFired == 1)
        {
```

```
        x = other.x;
        y = other.y;

        xDistance = ofGetMouseX() - x;
        yDistance = ofGetMouseY() - y;

        magnitude = sqrt(xDistance * xDistance + yDistance * yDistance);

        xDistance = xDistance / magnitude;
        yDistance = yDistance / magnitude;
      }
   }
}

bool Bullet::hits(Circle other)
{
   if (isFired == 1)
   {
      float sum, dist;

      bool result;

      sum = (x - other.x) * (x - other.x) + (y - other.y) * (y - other.y);

      dist = sqrt(sum);

      if (dist < r + other.r) result = true;
      else result = false;

      return(result);
   }
}

void Bullet::reload()
{
   if (numBullets <= 0)
   {
      reloadSound.play();
      numBullets = 6;
   }
}

void Bullet::bulletStop()
{
   x = 50000;
   y = 50000;
   speed = 0;
```

```
   isFired = 0;
}
```

```
class ofApp : public ofBaseApp{

   public:

      Player player;
      Enemy enemies[100];
      Bullet bullet;



      ofTrueTypeFont myFont;

      ofImage mainMenu;

      ofImage gameOver;

      string gameState;
}
```

```
void ofApp::setup()
{
   ofSetFrameRate(60);
   ofSetCircleResolution(100);

   myFont.load("bloody.otf", 60);
   mainMenu.load("mainmenu.png");
   gameOver.load("gameover.jpg");

   gameState = "mainMenu";

   player.setup();
   bullet.setup();

   for (int i = 0; i < 10; i++)
   {
      enemies[i].setup();
   }

   cout << gameState;
}
```

```
//--------------------------------------------------------------
void ofApp::update()
{
   if (gameState == "inGame")
   {
      player.update();
      bullet.update();

      for (int i = 0; i < 10; i++)
      {
         enemies[i].update();
         if (enemies[i].hits(player) == true) player.health = player.health - 1;
         if (bullet.hits(enemies[i]) == true && enemies[i].isAlive == "alive")
         {
            enemies[i].takeDamage();
            if (enemies[i].health <= 0)
            {
               enemies[i].isAlive = "dead";
               player.score = player.score + 10;
            }

            bullet.bulletStop();
         }

         enemies[i].seek(player);
      }

      if (player.score == 100) gameState = "youWon";

      if (player.health <= 0) gameState = "gameOver";
   }
}

//--------------------------------------------------------------
void ofApp::draw()
{
   if (gameState == "mainMenu")
   {
      ofSetBackgroundColor(0, 0, 0);

      ofSetColor(255, 255, 255);
      mainMenu.draw(50, 100, 900, 300);

      ofSetColor(255, 0, 0);
      myFont.drawString("Press TAB to start the game", 100, 475);

      myFont.drawString("Press I to see the instructions", 100, 550);
```

```
}
if (gameState == "instructions")
{
   ofSetBackgroundColor(0, 0, 0);

   ofSetColor(255, 0, 0);
   myFont.drawString("The apocalypse has begun and you", 50, 100);
   myFont.drawString("are attacked by a horde of zombies,", 50, 175);
   myFont.drawString("you must defend your life with", 50, 250);
   myFont.drawString("nothing but your trusty handgun.", 50, 325);

   myFont.drawString("Use W A S D to move the player", 50, 500);
   myFont.drawString("and left mouse to shoot.", 50, 575);
   myFont.drawString("BACKSPACE if you would", 50, 650);
   myFont.drawString("like to pause the game.", 50, 725);

   myFont.drawString("Press TAB to start the game", 50, 900);
}
if (gameState == "inGame")
{
   ofSetBackgroundColor(0, 80, 0);

   player.draw();
   bullet.draw();

   for (int i = 0; i < 10; i++)
   {
      enemies[i].draw();
   }
}
if (gameState == "paused")
{
   ofSetBackgroundColor(0, 0, 0);

   ofSetColor(255, 0, 0);
   myFont.drawString("PAUSED", 450, 500);

   myFont.drawString("Press TAB to resume", 275, 600);
}
if (gameState == "gameOver")
{
   player.health = 100;

   ofSetBackgroundColor(0, 0, 0);

   ofSetColor(255, 255, 255);
   gameOver.draw(-130, 100);
```

```
      ofSetColor(134, 0, 0);
      myFont.drawString("Press TAB To Try Again", 250, 600);

      player.setup();
      bullet.setup();

      for (int i = 0; i < 10; i++)
      {
         enemies[i].setup();
      }
   }
   if (gameState == "youWon")
   {
      player.health = 100;

      ofSetBackgroundColor(0, 0, 0);

      ofSetColor(134, 0, 0);
      myFont.drawString("Congratulations You Won", 200, 500);

      myFont.drawString("Press TAB To Play Again", 200, 600);

      player.setup();
      bullet.setup();

      for (int i = 0; i < 10; i++)
      {
         enemies[i].setup();
      }
   }
}

//--------------------------------------------------------------
void ofApp::keyPressed(int key)
{
   if (gameState == "inGame")
   {
      if (key == 'w') player.up();
      if (key == 's') player.down();
      if (key == 'a') player.left();
      if (key == 'd') player.right();

      if (key == 'r') bullet.reload();

      if (key == OF_KEY_BACKSPACE) gameState = "paused";
   }
```

```
    if (gameState == "mainMenu")
    {
        if (key == OF_KEY_TAB) gameState = "inGame";

        if (key == 'i') gameState = "instructions";
    }
    if (gameState == "instructions")
    {
        if (key == OF_KEY_TAB) gameState = "inGame";
    }
    if (gameState == "paused")
    {
        if (key == OF_KEY_TAB) gameState = "inGame";
    }
    if (gameState == "gameOver")
    {
        if (key == OF_KEY_TAB) gameState = "mainMenu";
    }
    if (gameState == "youWon")
    {
        if (key == OF_KEY_TAB) gameState = "mainMenu";
    }
}

//--------------------------------------------------------------
void ofApp::mousePressed(int x, int y, int button)
{
    if (gameState == "inGame")
    {
        if (button == 0) bullet.fired(player);
    }
}
```