# EXPLAIN CORE CONCEPT OF CUSTOM PROGRAMS – AAA game

Link to the game < https://drive.google.com/file/d/1YQWGl4Ps5uUmaxgHUq8nwBtUmev3wucL/view?usp=sharing >

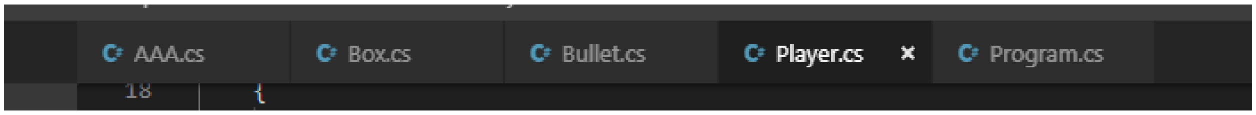
Or <https://www.youtube.com/watch?v=BjpwOxpWzgY&feature=youtu.be > if the previous link is not working

Link to download the code and resources

< https://drive.google.com/file/d/1MBse7ax7uYDxKSviyRvpB5I_6IpJWtC0/view?usp=sharing >

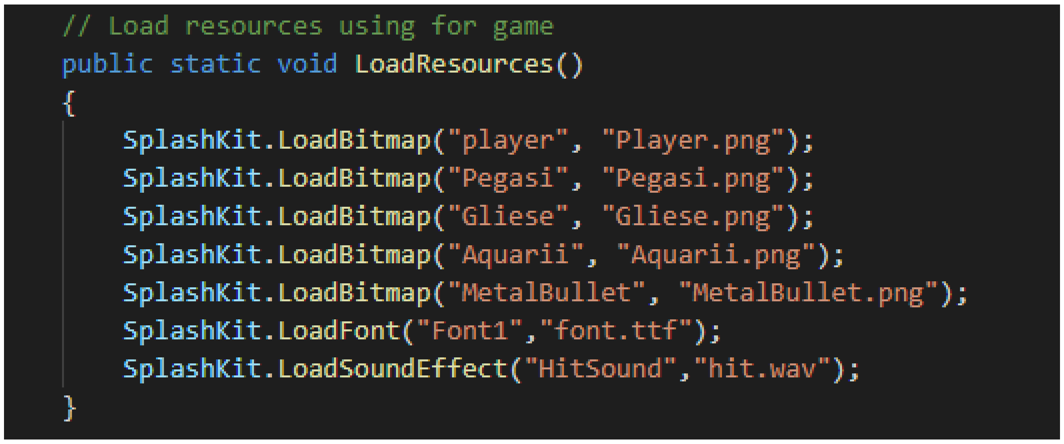
This video explains the core concept and execution of AAA game

We have 5 main cs files for creating the game: program, AAA, Bullet, Player, Box



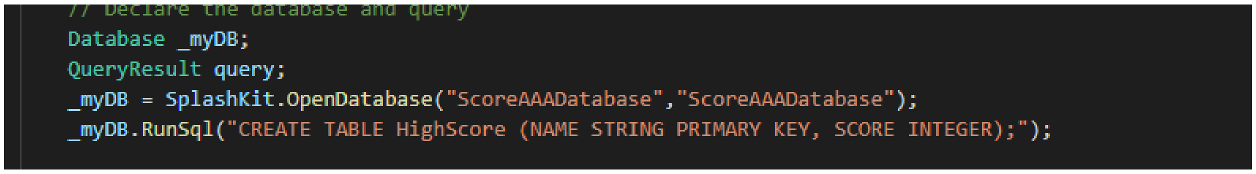
Let's go through the code together

First we need to Load All the resources we will use for the game includes the images, sound, font



```
// Load resources using for game
public static void LoadResources()
{
    SplashKit.LoadBitmap("player", "Player.png");
    SplashKit.LoadBitmap("Pegasi", "Pegasi.png");
    SplashKit.LoadBitmap("Gliese", "Gliese.png");
    SplashKit.LoadBitmap("Aquarii", "Aquarii.png");
    SplashKit.LoadBitmap("MetalBullet", "MetalBullet.png");
    SplashKit.LoadFont("Font1","font.ttf");
    SplashKit.LoadSoundEffect("HitSound","hit.wav");
}
```

Loading the database and HighScore table which stored our previous player names and scores. We will work with database to retrieve, update, insert new record for the Name and Score filed

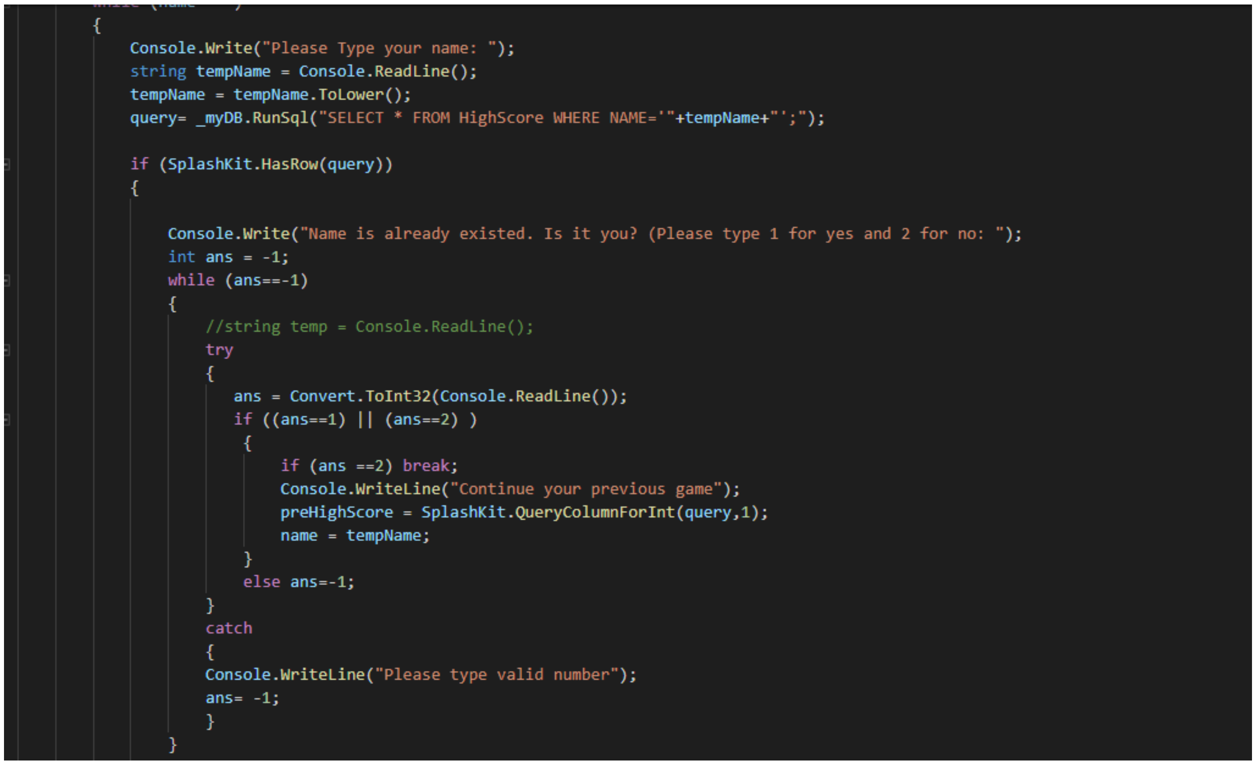


```
// Declare the database and query
Database _myDB;
QueryResult query;
_myDB = SplashKit.OpenDatabase("ScoreAAADatabase","ScoreAAADatabase");
_myDB.RunSql("CREATE TABLE HighScore (NAME STRING PRIMARY KEY, SCORE INTEGER);");
```

Before starting the game, we need to prompt player for his/her name. If the Name is already existed, ask whether user is the old player who want to retrieve and continue from the previous High Score.

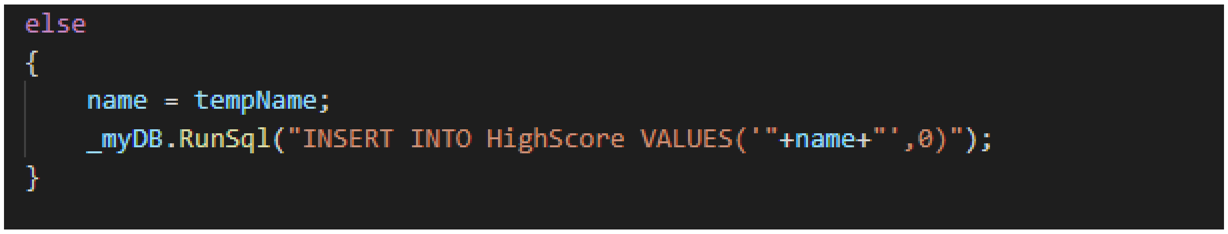
In case the user is the old player then we retrieve the highscore from table and assign into the preHighScore variable (default is 0)

If No, ask user for the new name.



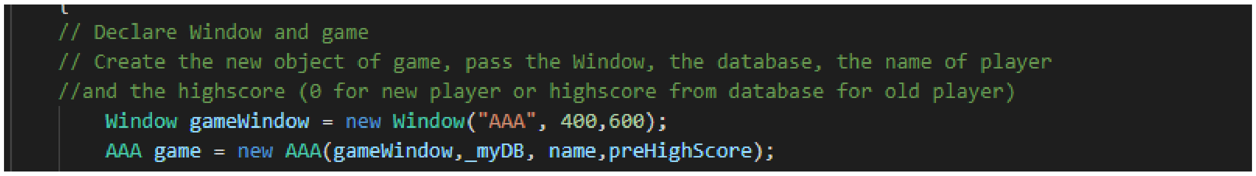
```
{
    Console.Write("Please Type your name: ");
    string tempName = Console.ReadLine();
    tempName = tempName.ToLower();
    query= _myDB.RunSql("SELECT * FROM HighScore WHERE NAME='"+tempName+"';");

    if (SplashKit.HasRow(query))
    {

        Console.Write("Name is already existed. Is it you? (Please type 1 for yes and 2 for no: ");
        int ans = -1;
        while (ans==-1)
        {
            //string temp = Console.ReadLine();
            try
            {
               ans = Convert.ToInt32(Console.ReadLine());
               if ((ans==1) || (ans==2) )
                {
                    if (ans ==2) break;
                    Console.WriteLine("Continue your previous game");
                    preHighScore = SplashKit.QueryColumnForInt(query,1);
                    name = tempName;
                }
                else ans=-1;
            }
            catch
            {
            Console.WriteLine("Please type valid number");
            ans= -1;
            }
        }
```

When the name is not existed, a new record will be added into database with the default preHighScore = 0



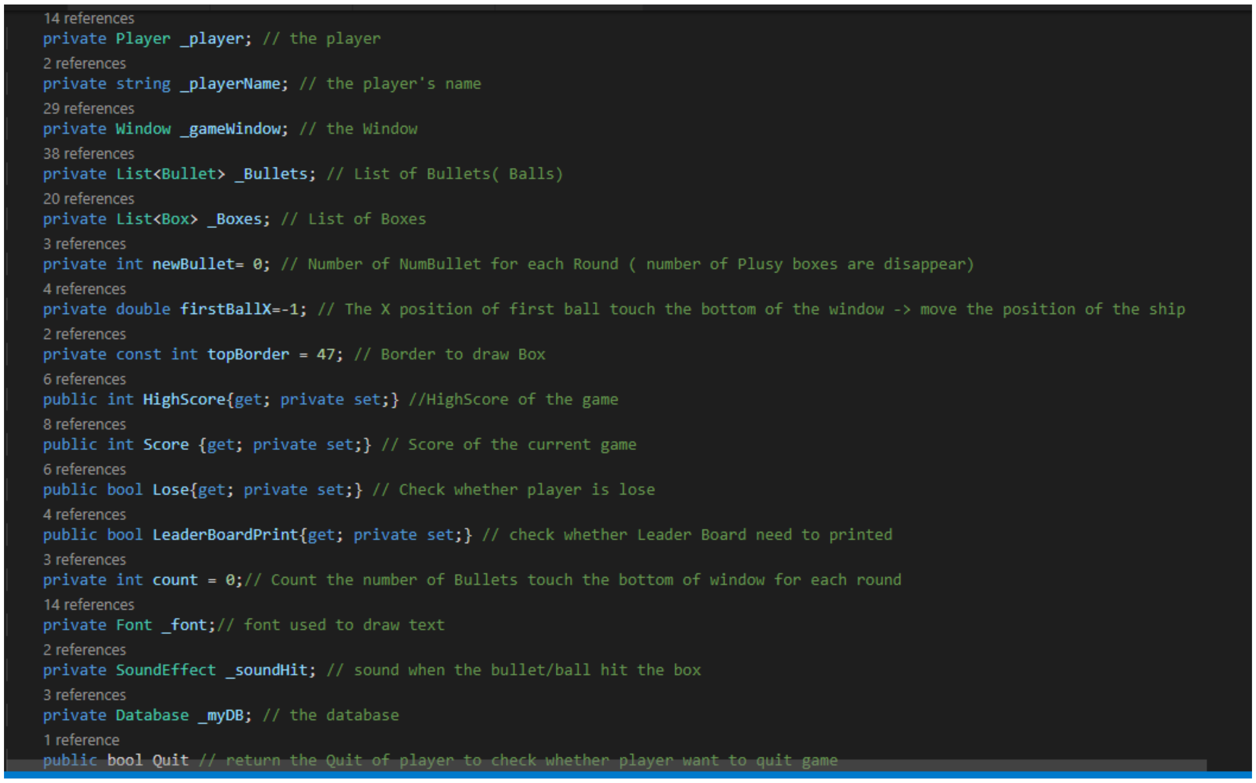
```
else
{
    name = tempName;
    _myDB.RunSql("INSERT INTO HighScore VALUES('"+name+"',0)");
}
```

Starting the game, we need to create new Window to draw to game and create new AAA game.



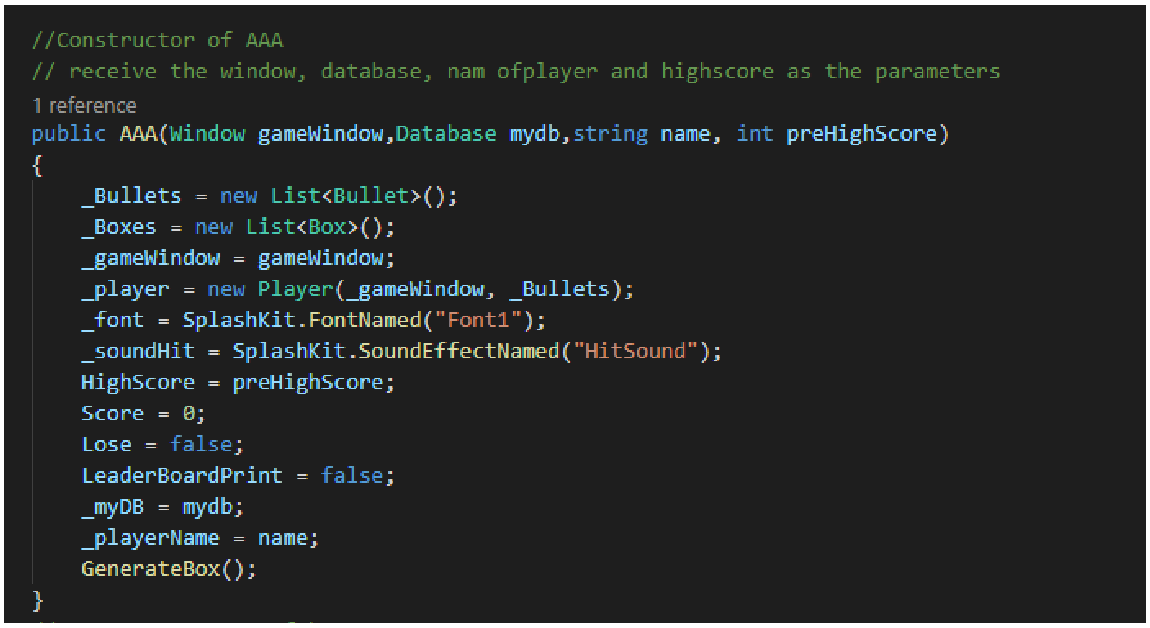
```
// Declare Window and game
// Create the new object of game, pass the Window, the database, the name of player
//and the highscore (0 for new player or highscore from database for old player)
    Window gameWindow = new Window("AAA", 400,600);
    AAA game = new AAA(gameWindow,_myDB, name,preHighScore);
```

For the AAA class, we has those attributes to define the player, List of bullet, List of box, High Score, Score, font, sound database,…



```
14 references
private Player _player; // the player
2 references
private string _playerName; // the player's name
29 references
private Window _gameWindow; // the Window
38 references
private List<Bullet> _Bullets; // List of Bullets( Balls)
20 references
private List<Box> _Boxes; // List of Boxes
3 references
private int newBullet= 0; // Number of NumBullet for each Round ( number of Plusy boxes are disappear)
4 references
private double firstBallX=-1; // The X position of first ball touch the bottom of the window -> move the position of the ship
2 references
private const int topBorder = 47; // Border to draw Box
6 references
public int HighScore{get; private set;} //HighScore of the game
8 references
public int Score {get; private set;} // Score of the current game
6 references
public bool Lose{get; private set;} // Check whether player is lose
4 references
public bool LeaderBoardPrint{get; private set;} // check whether Leader Board need to printed
3 references
private int count = 0;// Count the number of Bullets touch the bottom of window for each round
14 references
private Font _font;// font used to draw text
2 references
private SoundEffect _soundHit; // sound when the bullet/ball hit the box
3 references
private Database _myDB; // the database
1 reference
public bool Quit // return the Quit of player to check whether player want to quit game
```

The constructor of AAA will initialize the starting values for the attributes such as creating the new Player, create List used for Bullets, List used for Box,  Score = 0, HighScore equal to the preHighScore ,…



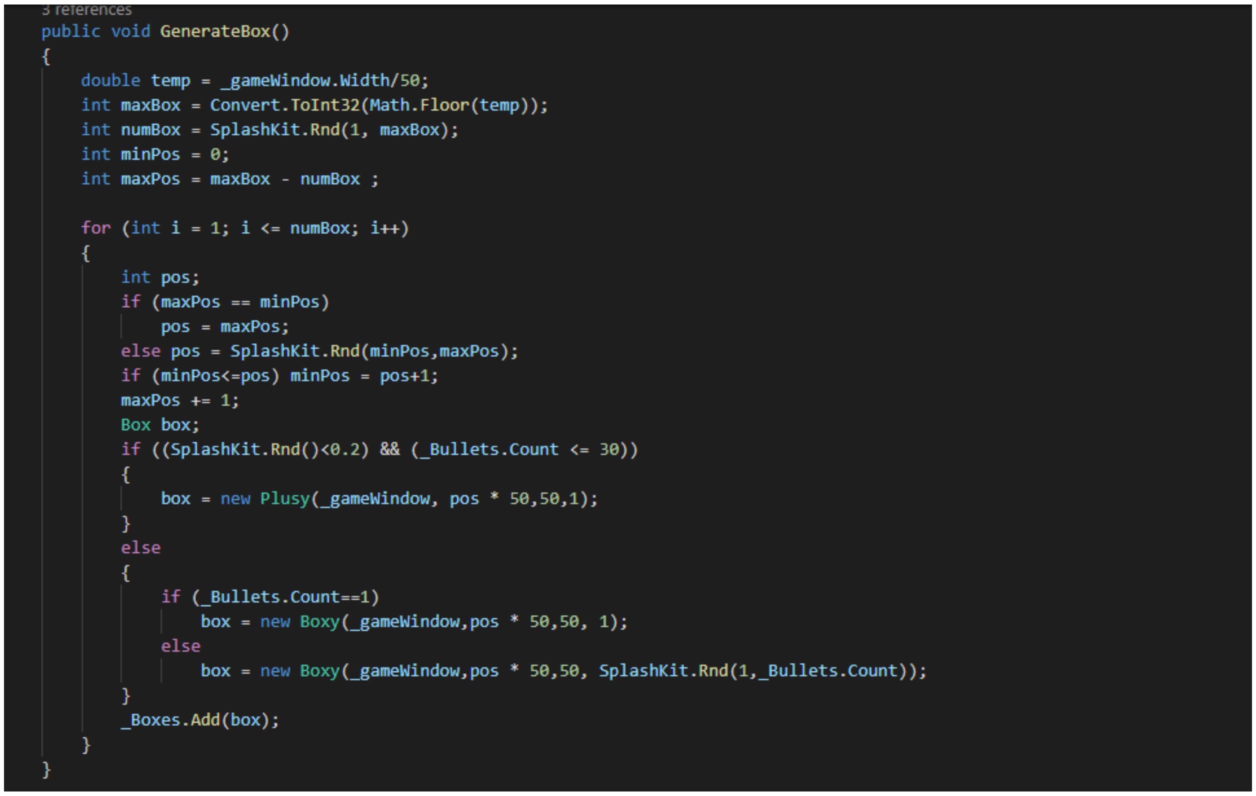
```
//Constructor of AAA
// receive the window, database, nam ofplayer and highscore as the parameters
1 reference
public AAA(Window gameWindow,Database mydb,string name, int preHighScore)
{
    _Bullets = new List<Bullet>();
    _Boxes = new List<Box>();
    _gameWindow = gameWindow;
    _player = new Player(_gameWindow, _Bullets);
    _font = SplashKit.FontNamed("Font1");
    _soundHit = SplashKit.SoundEffectNamed("HitSound");
    HighScore = preHighScore;
    Score = 0;
    Lose = false;
    LeaderBoardPrint = false;
    _myDB = mydb;
    _playerName = name;
    GenerateBox();
}
```

Here we have the GenerateBox() method  used to generate new row of Boxes.

Number of Box for each row is between min (0) and the max (based on the width of Window and the length of Box, here is 8 as the Window width is 400 and length of box is 50)

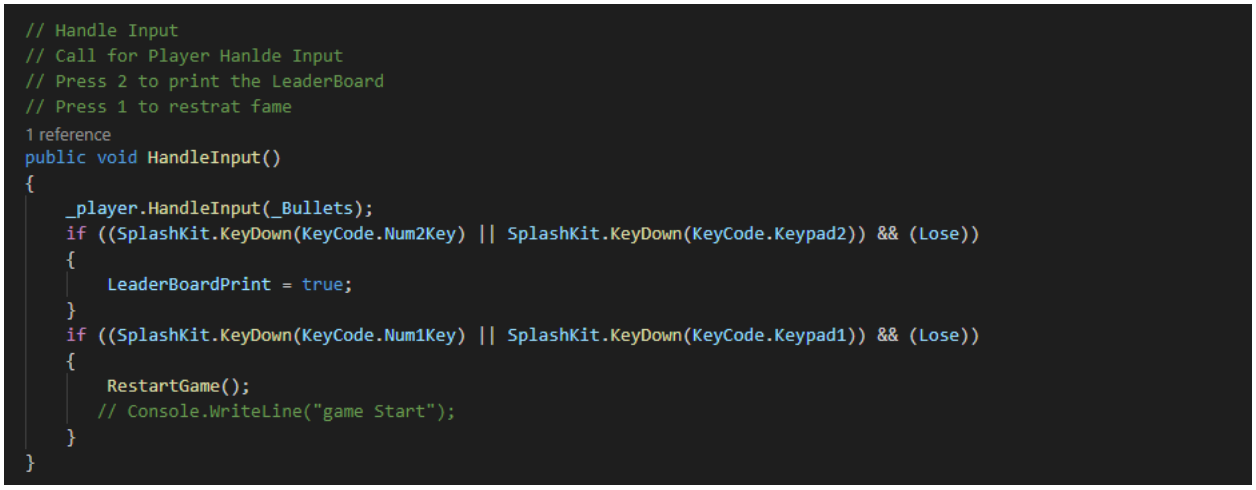
Type of Box is chosen using SPlashKit.Rnd() if the random number <0.2 and the total bullet is no more than 30) then we can allow to create Plusy Box to make changes to increase the bullet number. Or Opposite, we create the Boxy box for player to attach.

As Start, we will have one row of Box. Each round will be generate one more row of box



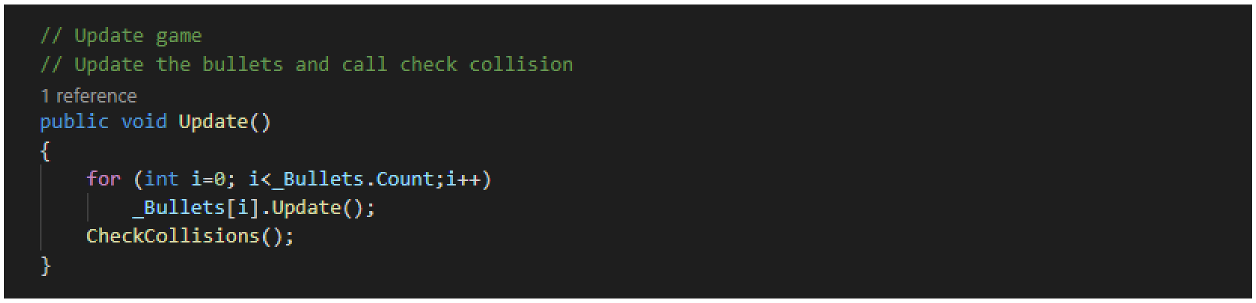
```
3 references
public void GenerateBox()
{
    double temp = _gameWindow.Width/50;
    int maxBox = Convert.ToInt32(Math.Floor(temp));
    int numBox = SplashKit.Rnd(1, maxBox);
    int minPos = 0;
    int maxPos = maxBox - numBox ;

    for (int i = 1; i <= numBox; i++)
    {
        int pos;
        if (maxPos == minPos)
            pos = maxPos;
        else pos = SplashKit.Rnd(minPos,maxPos);
        if (minPos<=pos) minPos = pos+1;
        maxPos += 1;
        Box box;
        if ((SplashKit.Rnd()<0.2) && (_Bullets.Count <= 30))
        {
            box = new Plusy(_gameWindow, pos * 50,50,1);
        }
        else
        {
            if (_Bullets.Count==1)
                box = new Boxy(_gameWindow,pos * 50,50, 1);
            else
                box = new Boxy(_gameWindow,pos * 50,50, SplashKit.Rnd(1,_Bullets.Count));
        }
        _Boxes.Add(box);
    }
}
```

HandleInput is used to trigger input of player includes press 1 to restart game, press 2 to print Leader Board or input related to the gameplayer as shooting or quit game



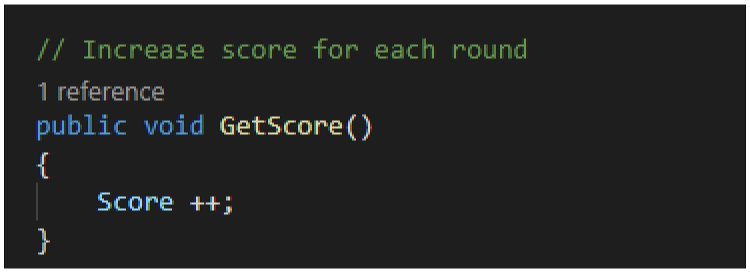
```
// Handle Input
// Call for Player Hanlde Input
// Press 2 to print the LeaderBoard
// Press 1 to restrat fame
1 reference
public void HandleInput()
{
    _player.HandleInput(_Bullets);
    if ((SplashKit.KeyDown(KeyCode.Num2Key) || SplashKit.KeyDown(KeyCode.Keypad2)) && (Lose))
    {
        LeaderBoardPrint = true;
    }
    if ((SplashKit.KeyDown(KeyCode.Num1Key) || SplashKit.KeyDown(KeyCode.Keypad1)) && (Lose))
    {
        RestartGame();
       // Console.WriteLine("game Start");
    }
}
```

Update method is used to update the position of bullet and check the collision of Bullet and Box to its surrounding



```
// Update game
// Update the bullets and call check collision
1 reference
public void Update()
{
    for (int i=0; i<_Bullets.Count;i++)
        _Bullets[i].Update();
    CheckCollisions();
}
```

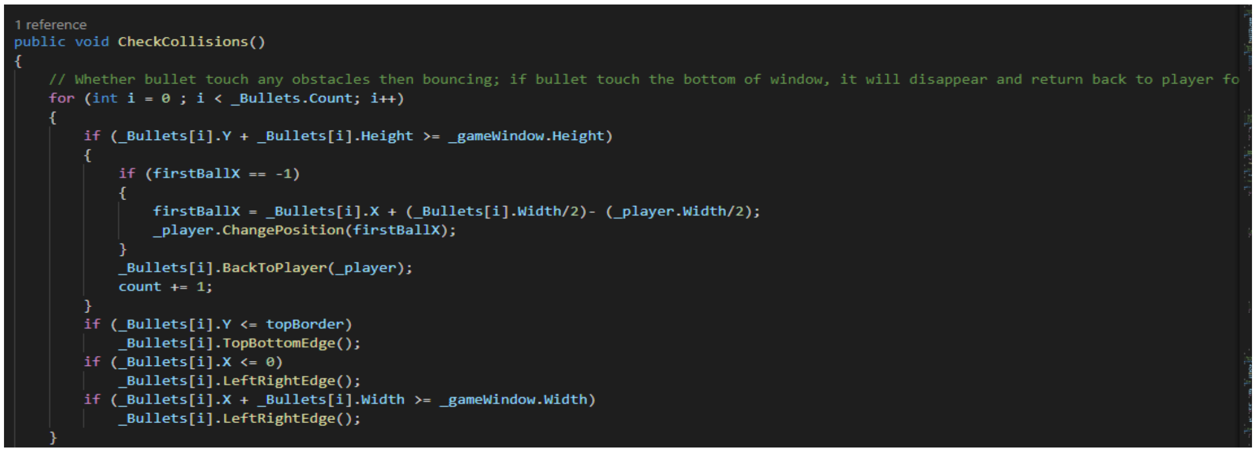
GetScore for adding one more point in score



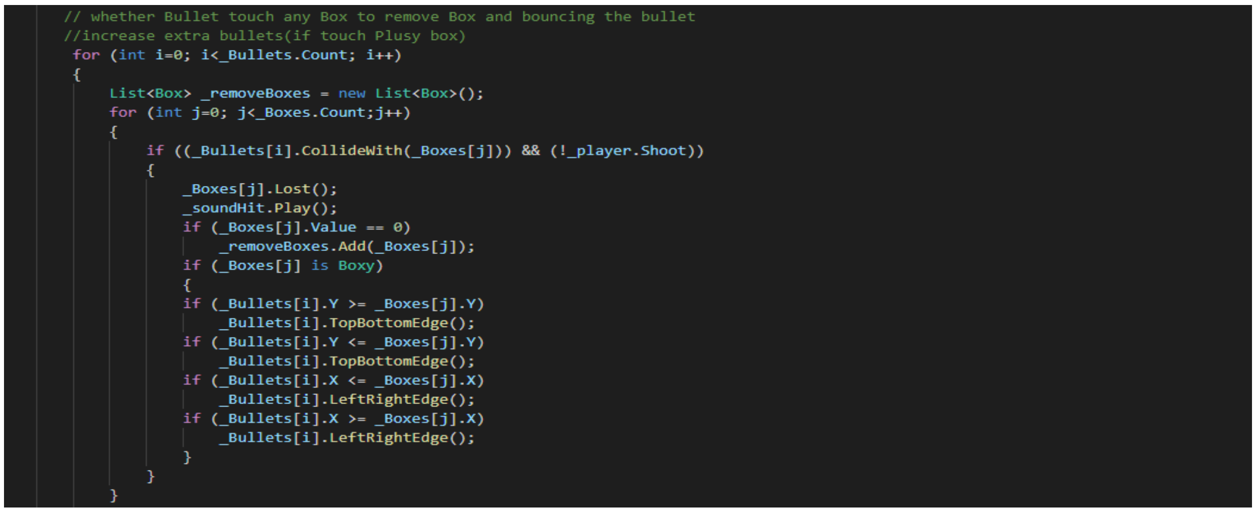
```
// Increase score for each round
1 reference
public void GetScore()
{
    Score ++;
}
```

CheckCollision will detect whether bullet touch the Window edge (except bottom of Window where bullet will go back to the player position) or Box side so that the bullet can bouncing and change its velocity; whether bullet touches the box to remove 1 point of Box's value. When Box's value =0, the box will disappear.

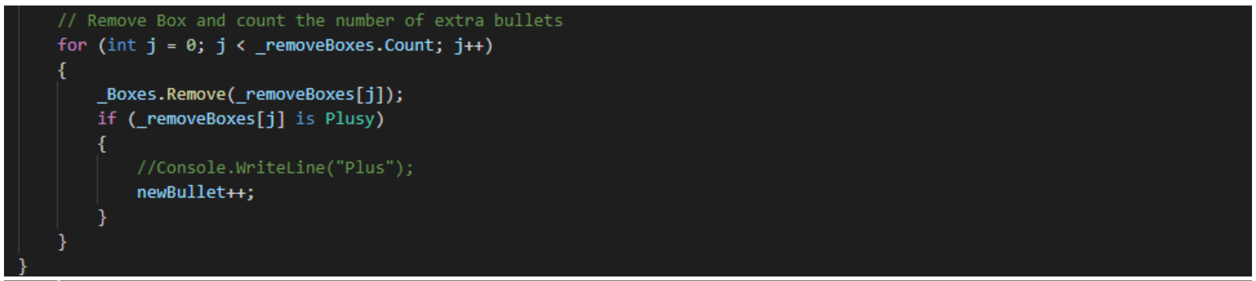
For each shooting, when all the bullets go back to the player, new round starts by generating new row of box and increase the Score (getScore) and add new bullets ( the number is based on how many plusy Box is hit from the last round)



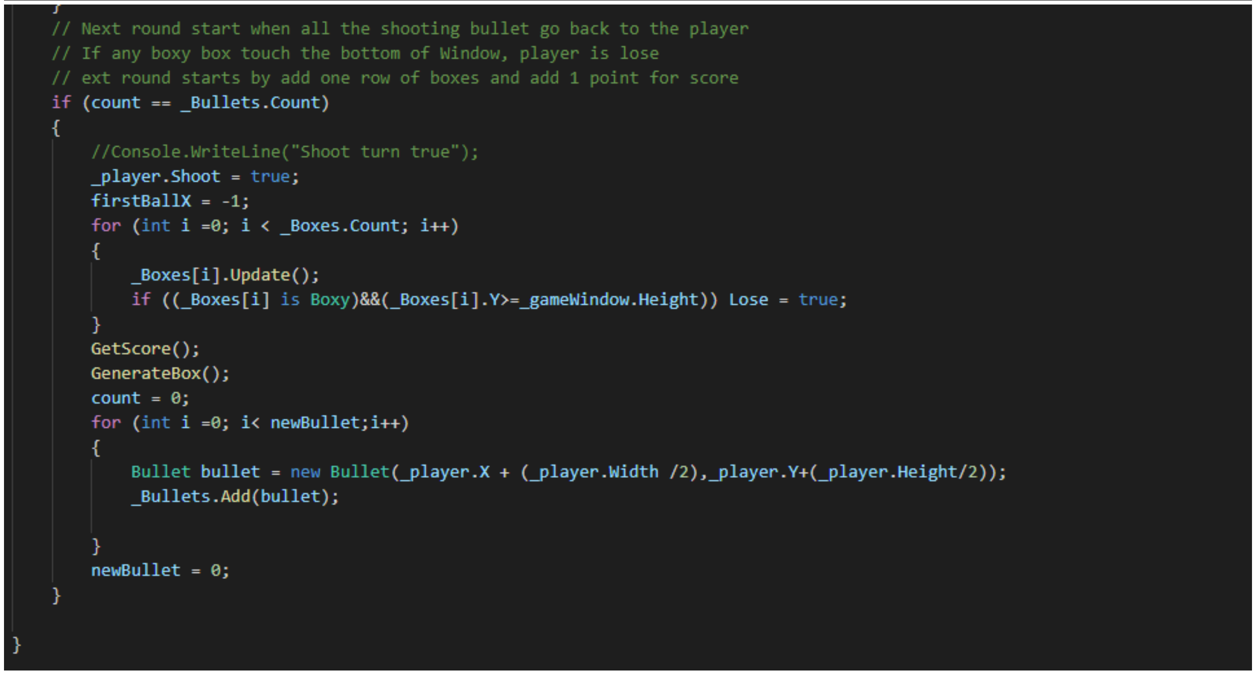
```
1 reference
public void CheckCollisions()
{
    // Whether bullet touch any obstacles then bouncing; if bullet touch the bottom of window, it will disappear and return back to player fo
    for (int i = 0 ; i < _Bullets.Count; i++)
    {
        if (_Bullets[i].Y + _Bullets[i].Height >= _gameWindow.Height)
        {
            if (firstBallX == -1)
            {
                firstBallX = _Bullets[i].X + (_Bullets[i].Width/2)- (_player.Width/2);
                _player.ChangePosition(firstBallX);
            }
            _Bullets[i].BackToPlayer(_player);
            count += 1;
        }
        if (_Bullets[i].Y <= topBorder)
            _Bullets[i].TopBottomEdge();
        if (_Bullets[i].X <= 0)
            _Bullets[i].LeftRightEdge();
        if (_Bullets[i].X + _Bullets[i].Width >= _gameWindow.Width)
            _Bullets[i].LeftRightEdge();
    }
```



```
    // whether Bullet touch any Box to remove Box and bouncing the bullet
    //increase extra bullets(if touch Plusy box)
     for (int i=0; i<_Bullets.Count; i++)
     {
         List<Box> _removeBoxes = new List<Box>();
         for (int j=0; j<_Boxes.Count;j++)
         {
             if ((_Bullets[i].CollideWith(_Boxes[j])) && (!_player.Shoot))
             {
                 _Boxes[j].Lost();
                 _soundHit.Play();
                 if (_Boxes[j].Value == 0)
                     _removeBoxes.Add(_Boxes[j]);
                 if (_Boxes[j] is Boxy)
                 {
                 if (_Bullets[i].Y >= _Boxes[j].Y)
                     _Bullets[i].TopBottomEdge();
                 if (_Bullets[i].Y <= _Boxes[j].Y)
                     _Bullets[i].TopBottomEdge();
                 if (_Bullets[i].X <= _Boxes[j].X)
                     _Bullets[i].LeftRightEdge();
                 if (_Bullets[i].X >= _Boxes[j].X)
                     _Bullets[i].LeftRightEdge();
                 }
             }
         }
```

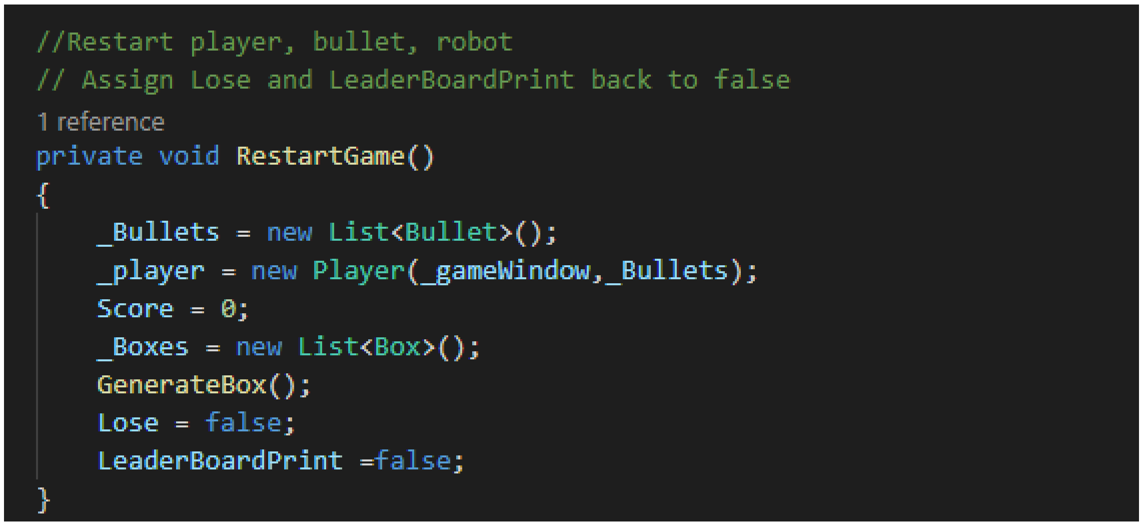


```
    // Remove Box and count the number of extra bullets
    for (int j = 0; j < _removeBoxes.Count; j++)
    {
        _Boxes.Remove(_removeBoxes[j]);
        if (_removeBoxes[j] is Plusy)
        {
            //Console.WriteLine("Plus");
            newBullet++;
        }
    }
}
```



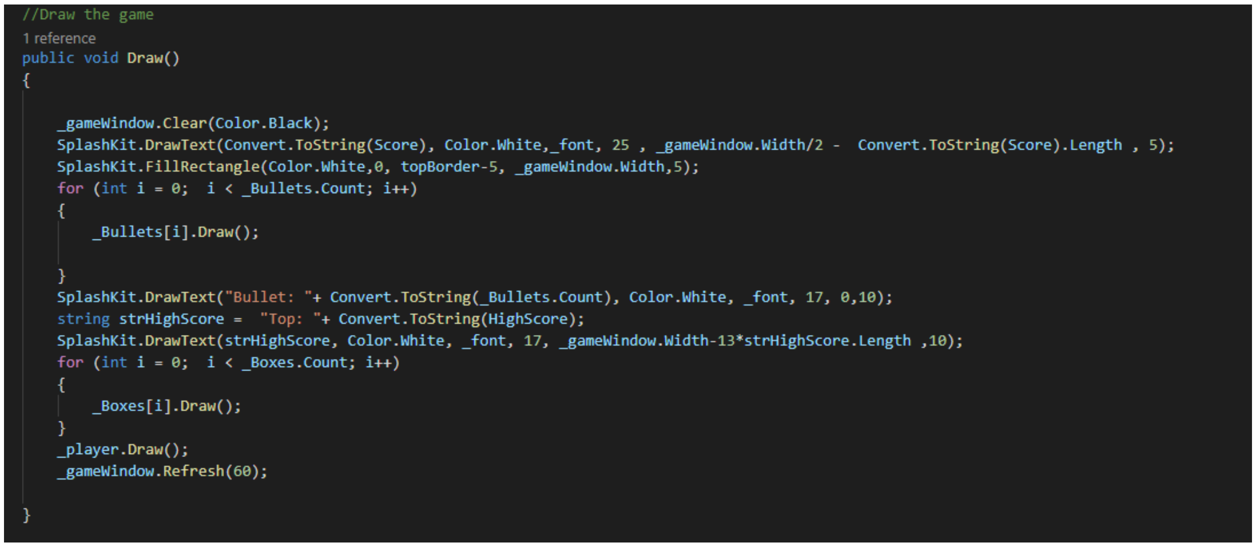
```
    // Next round start when all the shooting bullet go back to the player
    // If any boxy box touch the bottom of Window, player is lose
    // ext round starts by add one row of boxes and add 1 point for score
    if (count == _Bullets.Count)
    {
        //Console.WriteLine("Shoot turn true");
        _player.Shoot = true;
        firstBallX = -1;
        for (int i =0; i < _Boxes.Count; i++)
        {
            _Boxes[i].Update();
            if ((_Boxes[i] is Boxy)&&(_Boxes[i].Y>=_gameWindow.Height)) Lose = true;
        }
        GetScore();
        GenerateBox();
        count = 0;
        for (int i =0; i< newBullet;i++)
        {
            Bullet bullet = new Bullet(_player.X + (_player.Width /2),_player.Y+(_player.Height/2));
            _Bullets.Add(bullet);

        }
        newBullet = 0;
    }

}
```

RestartGame is used to restart the player, score, bullets,…



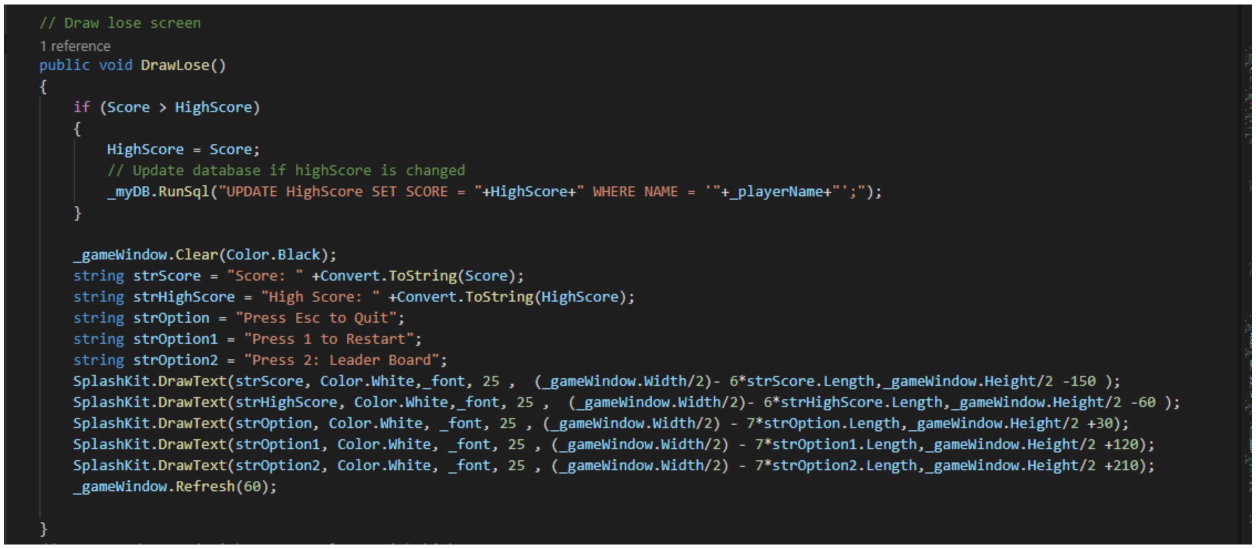
```
//Restart player, bullet, robot
// Assign Lose and LeaderBoardPrint back to false
1 reference
private void RestartGame()
{
    _Bullets = new List<Bullet>();
    _player = new Player(_gameWindow,_Bullets);
    Score = 0;
    _Boxes = new List<Box>();
    GenerateBox();
    Lose = false;
    LeaderBoardPrint =false;
}
```

We have draw method to draw the game element include the player, bullets, boxes and texts related to score, number of Bullet, highScore



```
//Draw the game
1 reference
public void Draw()
{

    _gameWindow.Clear(Color.Black);
    SplashKit.DrawText(Convert.ToString(Score), Color.White,_font, 25 , _gameWindow.Width/2 -  Convert.ToString(Score).Length , 5);
    SplashKit.FillRectangle(Color.White,0, topBorder-5, _gameWindow.Width,5);
    for (int i = 0;  i < _Bullets.Count; i++)
    {
        _Bullets[i].Draw();

    }
    SplashKit.DrawText("Bullet: "+ Convert.ToString(_Bullets.Count), Color.White, _font, 17, 0,10);
    string strHighScore =  "Top: "+ Convert.ToString(HighScore);
    SplashKit.DrawText(strHighScore, Color.White, _font, 17, _gameWindow.Width-13*strHighScore.Length ,10);
    for (int i = 0;  i < _Boxes.Count; i++)
    {
        _Boxes[i].Draw();
    }
    _player.Draw();
    _gameWindow.Refresh(60);

}
```

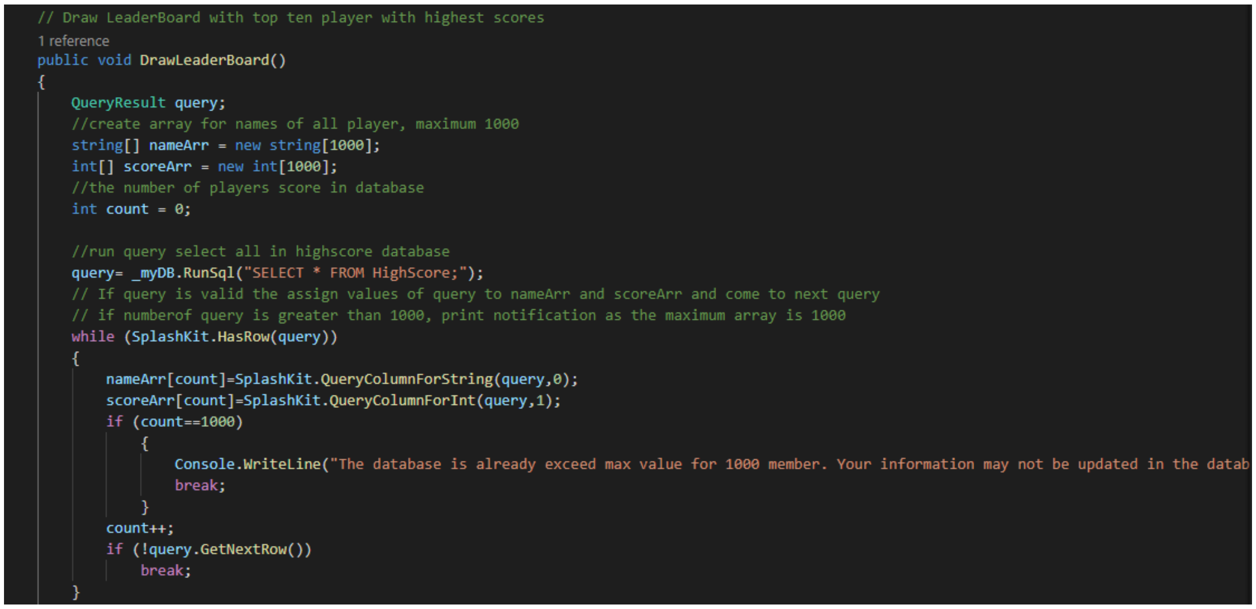
DrawLose method draws the lose screen where user can choose to restart , quit the game or check out the leaderboard



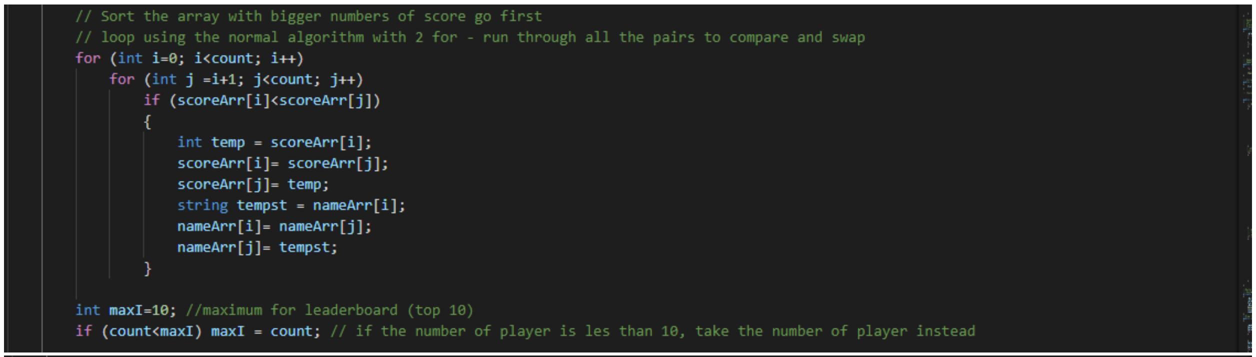
```
// Draw lose screen
1 reference
public void DrawLose()
{
    if (Score > HighScore)
    {
        HighScore = Score;
        // Update database if highScore is changed
        _myDB.RunSql("UPDATE HighScore SET SCORE = "+HighScore+" WHERE NAME = '"+_playerName+"';");
    }

    _gameWindow.Clear(Color.Black);
    string strScore = "Score: " +Convert.ToString(Score);
    string strHighScore = "High Score: " +Convert.ToString(HighScore);
    string strOption = "Press Esc to Quit";
    string strOption1 = "Press 1 to Restart";
    string strOption2 = "Press 2: Leader Board";
    SplashKit.DrawText(strScore, Color.White,_font, 25 ,  (_gameWindow.Width/2)- 6*strScore.Length,_gameWindow.Height/2 -150 );
    SplashKit.DrawText(strHighScore, Color.White,_font, 25 ,  (_gameWindow.Width/2)- 6*strHighScore.Length,_gameWindow.Height/2 -60 );
    SplashKit.DrawText(strOption, Color.White, _font, 25 , (_gameWindow.Width/2) - 7*strOption.Length,_gameWindow.Height/2 +30);
    SplashKit.DrawText(strOption1, Color.White, _font, 25 , (_gameWindow.Width/2) - 7*strOption1.Length,_gameWindow.Height/2 +120);
    SplashKit.DrawText(strOption2, Color.White, _font, 25 , (_gameWindow.Width/2) - 7*strOption2.Length,_gameWindow.Height/2 +210);
    _gameWindow.Refresh(60);

}
```

DrawLeaderBoard draws the top players and scores (maximum 10)

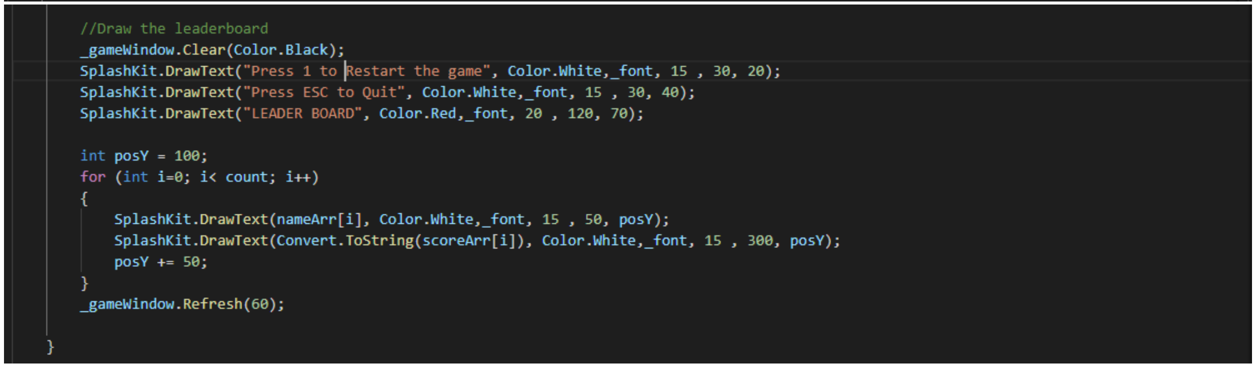
For the LeaderBoard drawing, we first create 2 array to retrieve all values of name and score in the our database then sort 2 arrays so that the higher scores and their player name according go first. To print, we simply just take 10 first values or if the number of record is less than 10 then print all the array



```
// Draw LeaderBoard with top ten player with highest scores
1 reference
public void DrawLeaderBoard()
{
    QueryResult query;
    //create array for names of all player, maximum 1000
    string[] nameArr = new string[1000];
    int[] scoreArr = new int[1000];
    //the number of players score in database
    int count = 0;

    //run query select all in highscore database
    query= _myDB.RunSql("SELECT * FROM HighScore;");
    // If query is valid the assign values of query to nameArr and scoreArr and come to next query
    // if numberof query is greater than 1000, print notification as the maximum array is 1000
    while (SplashKit.HasRow(query))
    {
        nameArr[count]=SplashKit.QueryColumnForString(query,0);
        scoreArr[count]=SplashKit.QueryColumnForInt(query,1);
        if (count==1000)
            {
                Console.WriteLine("The database is already exceed max value for 1000 member. Your information may not be updated in the datab
                break;
            }
        count++;
        if (!query.GetNextRow())
            break;
    }
```

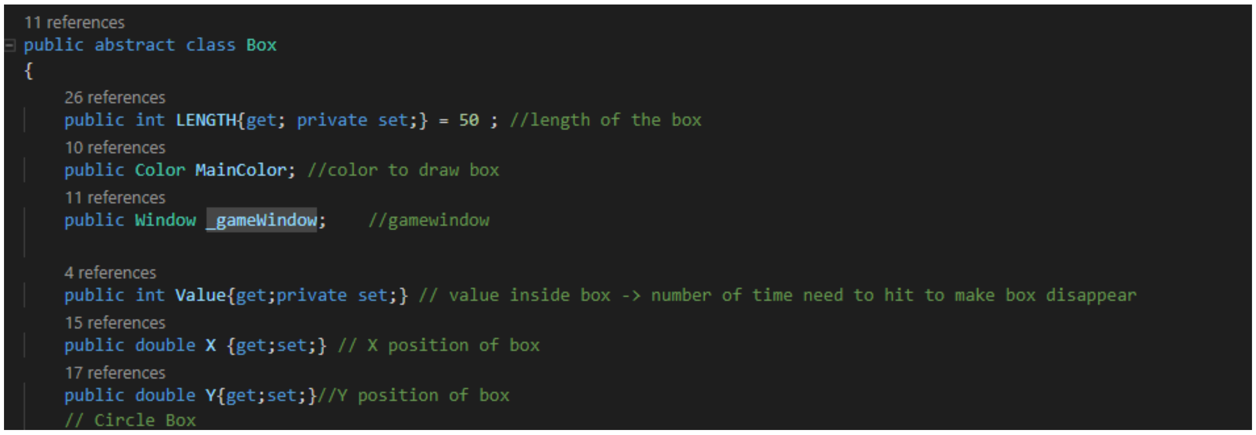


```
// Sort the array with bigger numbers of score go first
// loop using the normal algorithm with 2 for - run through all the pairs to compare and swap
for (int i=0; i<count; i++)
    for (int j =i+1; j<count; j++)
        if (scoreArr[i]<scoreArr[j])
        {
            int temp = scoreArr[i];
            scoreArr[i]= scoreArr[j];
            scoreArr[j]= temp;
            string tempst = nameArr[i];
            nameArr[i]= nameArr[j];
            nameArr[j]= tempst;
        }

int maxI=10; //maximum for leaderboard (top 10)
if (count<maxI) maxI = count; // if the number of player is les than 10, take the number of player instead
```



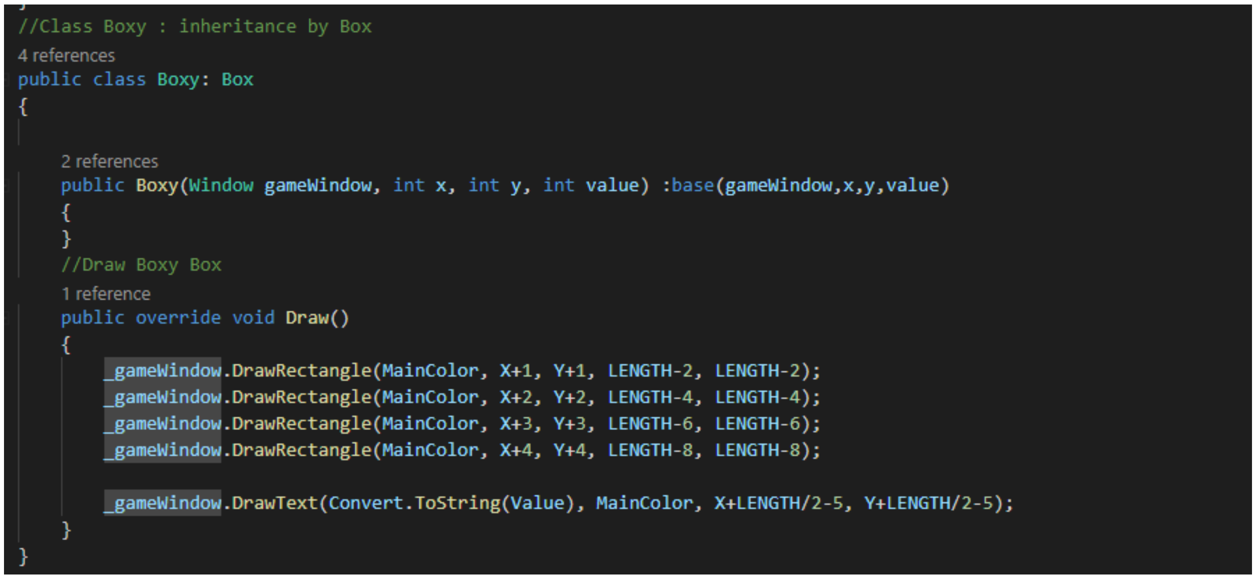
```
    //Draw the leaderboard
    _gameWindow.Clear(Color.Black);
    SplashKit.DrawText("Press 1 to Restart the game", Color.White,_font, 15 , 30, 20);
    SplashKit.DrawText("Press ESC to Quit", Color.White,_font, 15 , 30, 40);
    SplashKit.DrawText("LEADER BOARD", Color.Red,_font, 20 , 120, 70);

    int posY = 100;
    for (int i=0; i< count; i++)
    {
        SplashKit.DrawText(nameArr[i], Color.White,_font, 15 , 50, posY);
        SplashKit.DrawText(Convert.ToString(scoreArr[i]), Color.White,_font, 15 , 300, posY);
        posY += 50;
    }
    _gameWindow.Refresh(60);

}
```

For the Box class, we have Box as the abstract class which declares the variables and method for the Plusy and Boxy to inherit such as Update for increase the Y position to move towards to the bottom of Window



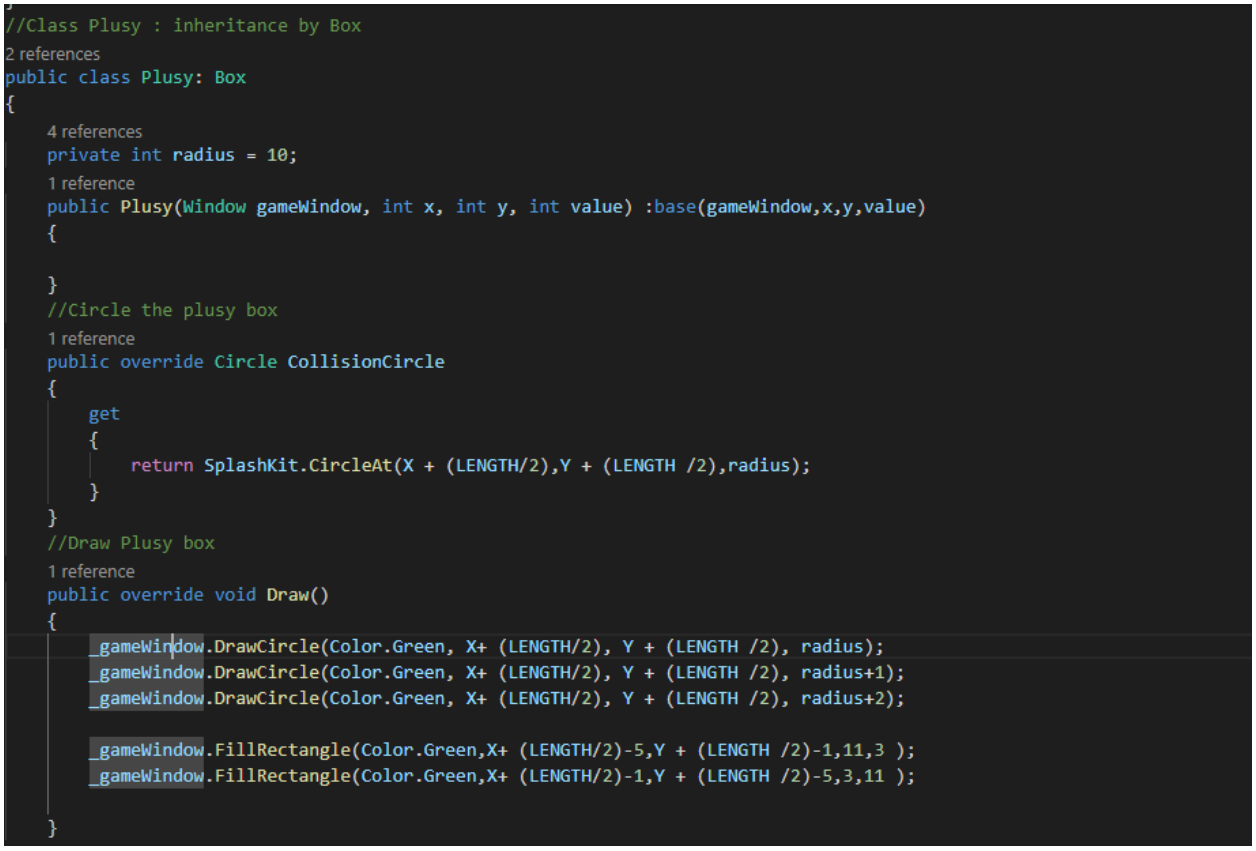
```
11 references
public abstract class Box
{
    26 references
    public int LENGTH{get; private set;} = 50 ; //length of the box
    10 references
    public Color MainColor; //color to draw box
    11 references
    public Window _gameWindow;     //gamewindow

    4 references
    public int Value{get;private set;} // value inside box -> number of time need to hit to make box disappear
    15 references
    public double X {get;set;} // X position of box
    17 references
    public double Y{get;set;}//Y position of box
    // Circle Box
```

Boxy is the child of Box class, which is the box the player need to attach. I any Boxy moves beyond the bottom of Window, the player will lose



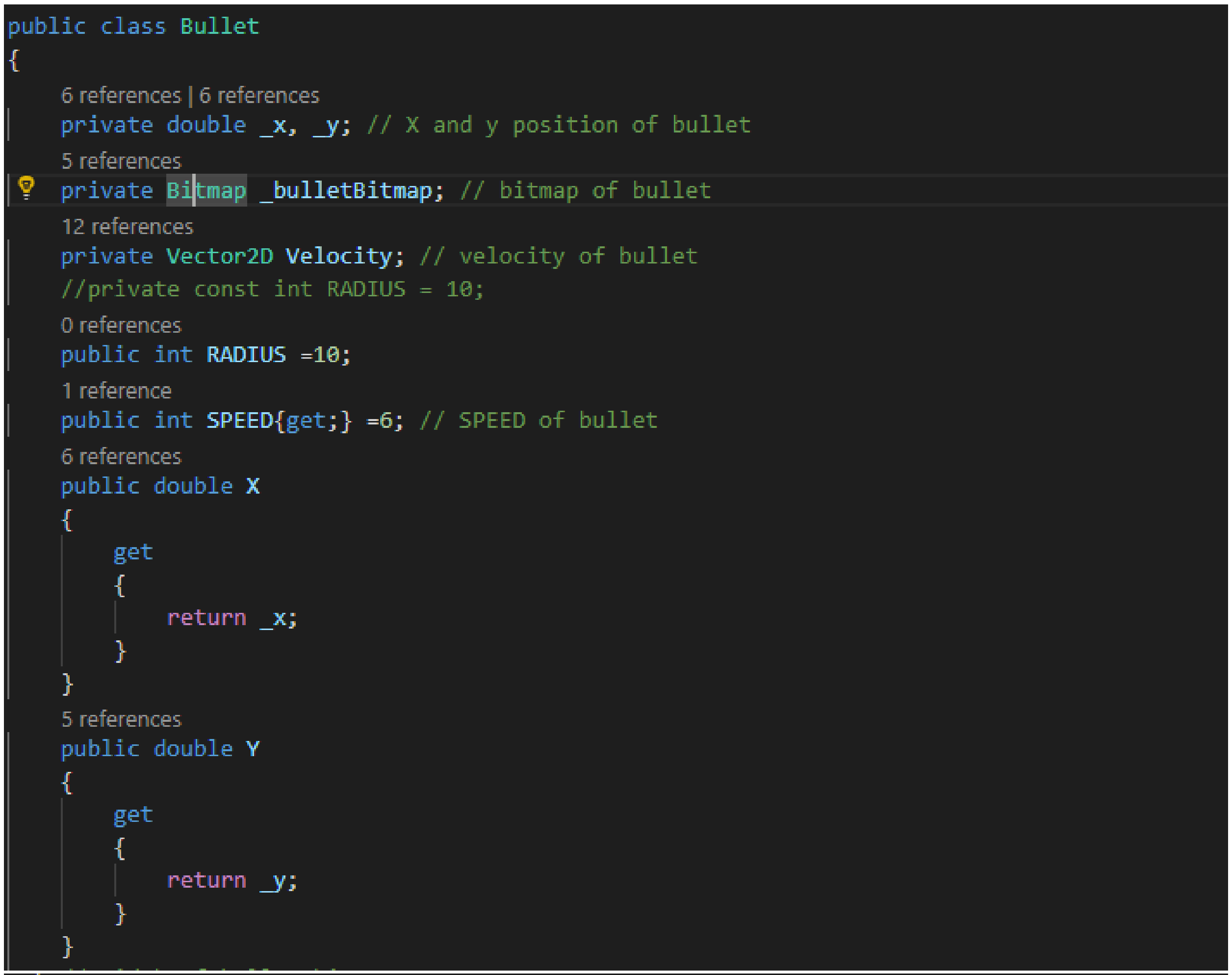
```
//Class Boxy : inheritance by Box
4 references
public class Boxy: Box
{

    2 references
    public Boxy(Window gameWindow, int x, int y, int value) :base(gameWindow,x,y,value)
    {
    }
    //Draw Boxy Box
    1 reference
    public override void Draw()
    {
        _gameWindow.DrawRectangle(MainColor, X+1, Y+1, LENGTH-2, LENGTH-2);
        _gameWindow.DrawRectangle(MainColor, X+2, Y+2, LENGTH-4, LENGTH-4);
        _gameWindow.DrawRectangle(MainColor, X+3, Y+3, LENGTH-6, LENGTH-6);
        _gameWindow.DrawRectangle(MainColor, X+4, Y+4, LENGTH-8, LENGTH-8);

        _gameWindow.DrawText(Convert.ToString(Value), MainColor, X+LENGTH/2-5, Y+LENGTH/2-5);
    }
}
```

The plusy Box ( + symbol) creates chances for player to earn mor bullet if they can shoot the Plusy

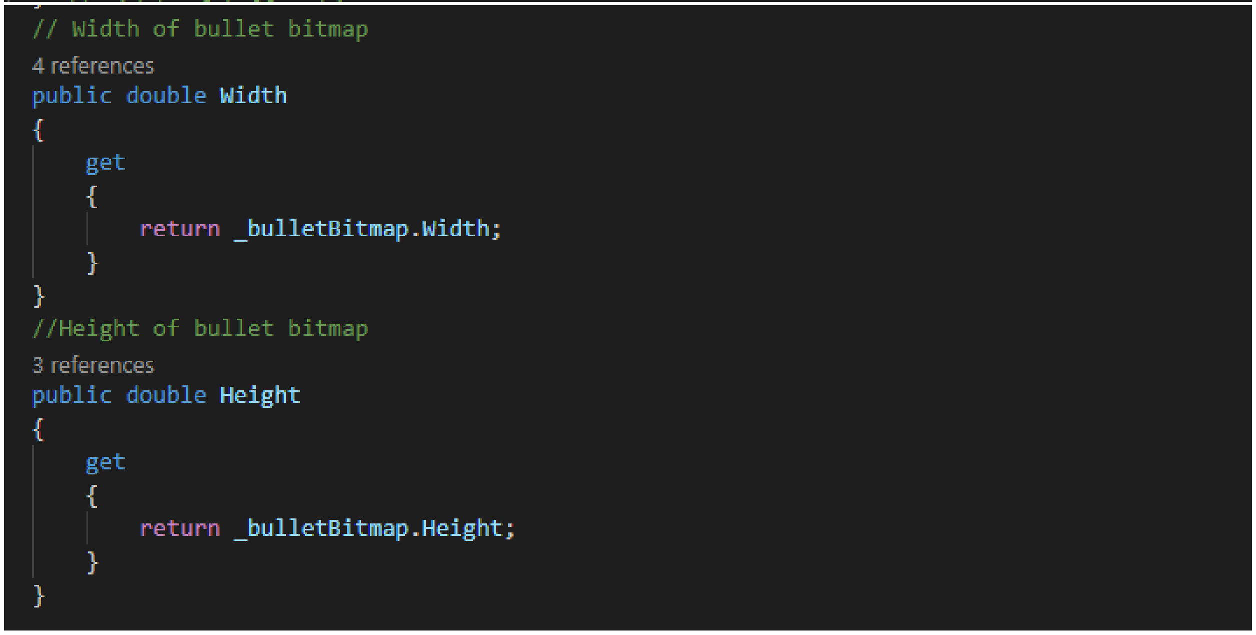


```
//Class Plusy : inheritance by Box
2 references
public class Plusy: Box
{
    4 references
    private int radius = 10;
    1 reference
    public Plusy(Window gameWindow, int x, int y, int value) :base(gameWindow,x,y,value)
    {

    }
    //Circle the plusy box
    1 reference
    public override Circle CollisionCircle
    {
        get
        {
            return SplashKit.CircleAt(X + (LENGTH/2),Y + (LENGTH /2),radius);
        }
    }
    //Draw Plusy box
    1 reference
    public override void Draw()
    {
        _gameWindow.DrawCircle(Color.Green, X+ (LENGTH/2), Y + (LENGTH /2), radius);
        _gameWindow.DrawCircle(Color.Green, X+ (LENGTH/2), Y + (LENGTH /2), radius+1);
        _gameWindow.DrawCircle(Color.Green, X+ (LENGTH/2), Y + (LENGTH /2), radius+2);

        _gameWindow.FillRectangle(Color.Green,X+ (LENGTH/2)-5,Y + (LENGTH /2)-1,11,3 );
        _gameWindow.FillRectangle(Color.Green,X+ (LENGTH/2)-1,Y + (LENGTH /2)-5,3,11 );

    }
```

For the Bullet class, we have attribute about its position, velocity, speed, Height and Width…

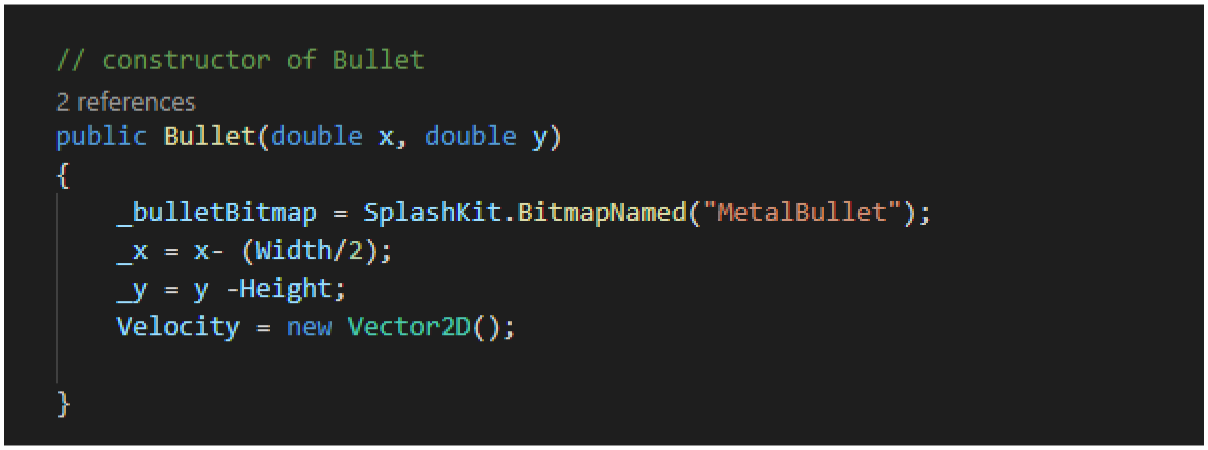


```
public class Bullet
{
    6 references | 6 references
    private double _x, _y; // X and y position of bullet
    5 references
    private Bitmap _bulletBitmap; // bitmap of bullet
    12 references
    private Vector2D Velocity; // velocity of bullet
    //private const int RADIUS = 10;
    0 references
    public int RADIUS =10;
    1 reference
    public int SPEED{get;} =6; // SPEED of bullet
    6 references
    public double X
    {
        get
        {
            return _x;
        }
    }
    5 references
    public double Y
    {
        get
        {
            return _y;
        }
    }
```



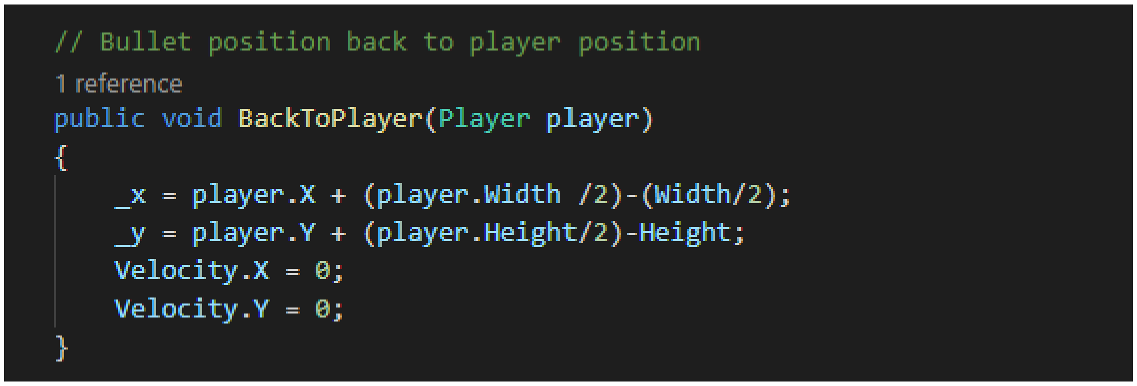
```
    // Width of bullet bitmap
    4 references
    public double Width
    {
        get
        {
            return _bulletBitmap.Width;
        }
    }
    //Height of bullet bitmap
    3 references
    public double Height
    {
        get
        {
            return _bulletBitmap.Height;
        }
    }
```

Bullet starting position is at the player position.



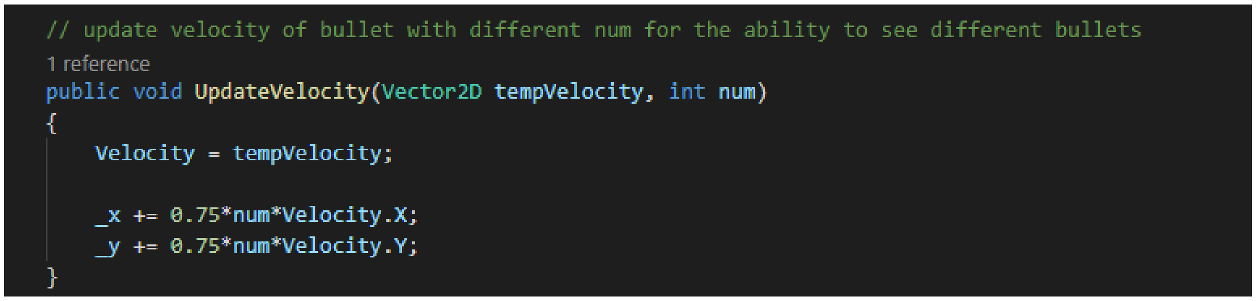
```
// constructor of Bullet
2 references
public Bullet(double x, double y)
{
    _bulletBitmap = SplashKit.BitmapNamed("MetalBullet");
    _x = x- (Width/2);
    _y = y -Height;
    Velocity = new Vector2D();

}
```

BackToPlayer method changes the position of Bullet back to the current position of Player and assign as not moving as the vector of velocity equal 0



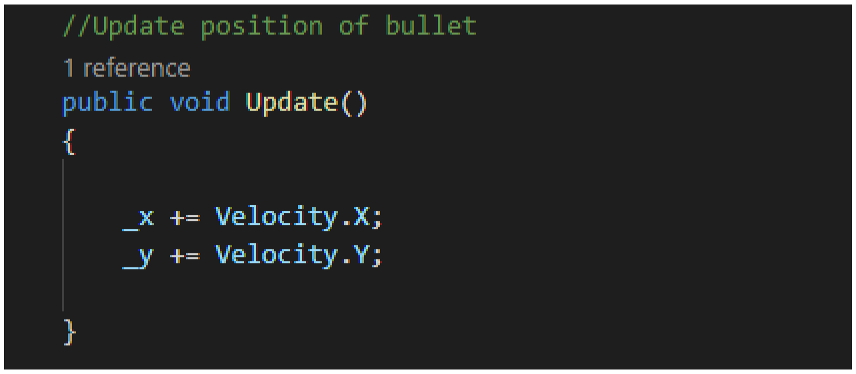
```
// Bullet position back to player position
1 reference
public void BackToPlayer(Player player)
{
    _x = player.X + (player.Width /2)-(Width/2);
    _y = player.Y + (player.Height/2)-Height;
    Velocity.X = 0;
    Velocity.Y = 0;
}
```

UpdateVelocity changes the velocity vector of bullet so that bullet can move towards the according angle and speed



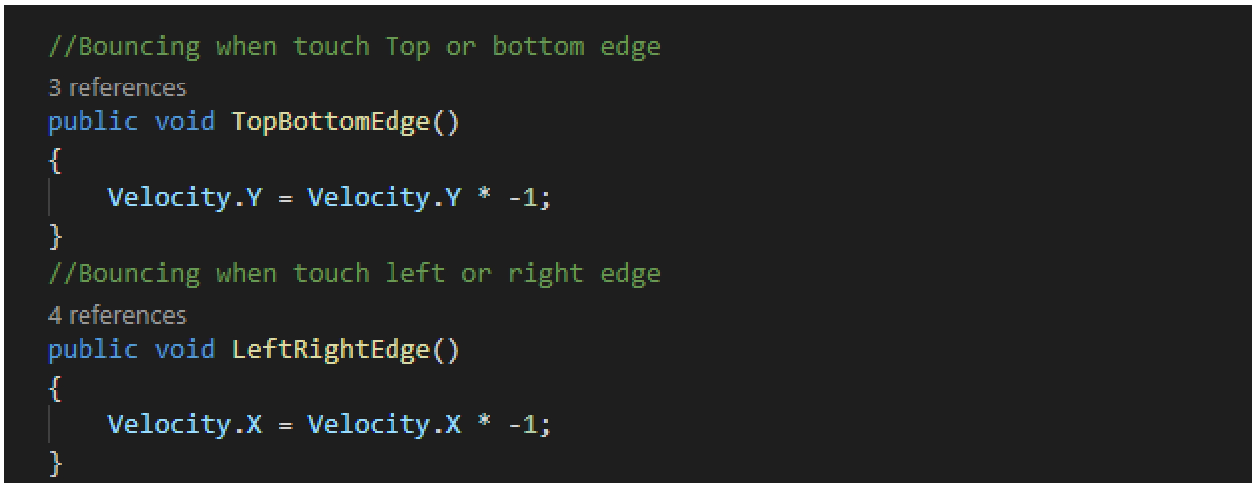
```
// update velocity of bullet with different num for the ability to see different bullets
1 reference
public void UpdateVelocity(Vector2D tempVelocity, int num)
{
    Velocity = tempVelocity;

    _x += 0.75*num*Velocity.X;
    _y += 0.75*num*Velocity.Y;
}
```

Update method updates the position of bullet



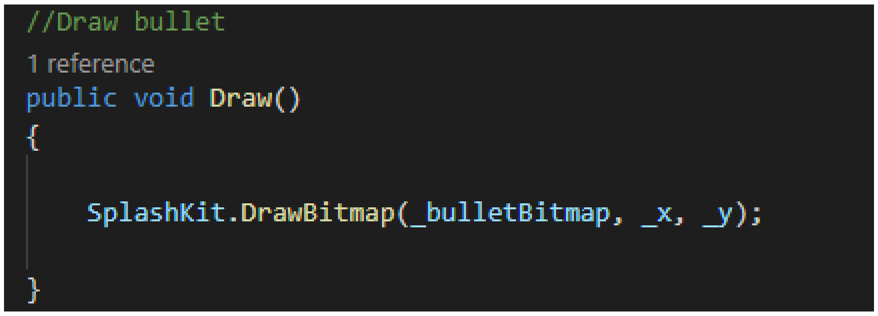
```
//Update position of bullet
1 reference
public void Update()
{

    _x += Velocity.X;
    _y += Velocity.Y;

}
```

TopBottomEdge and LeftRightEdge changes the velocity of Bullet so that bullet can bouncing when it reach the corresponding edge



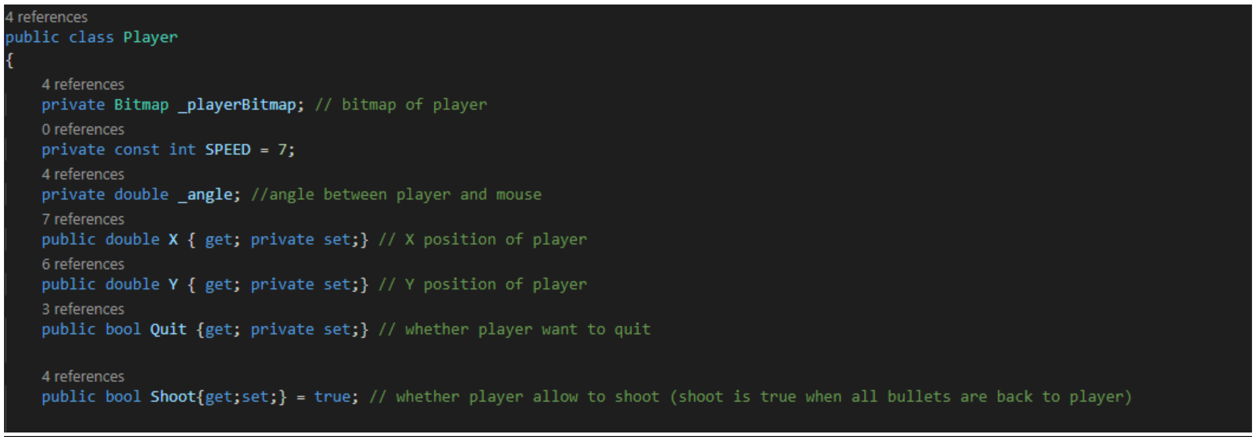
```
//Bouncing when touch Top or bottom edge
3 references
public void TopBottomEdge()
{
    Velocity.Y = Velocity.Y * -1;
}
//Bouncing when touch left or right edge
4 references
public void LeftRightEdge()
{
    Velocity.X = Velocity.X * -1;
}
```

And we have Draw method to draw the bullet on the screen

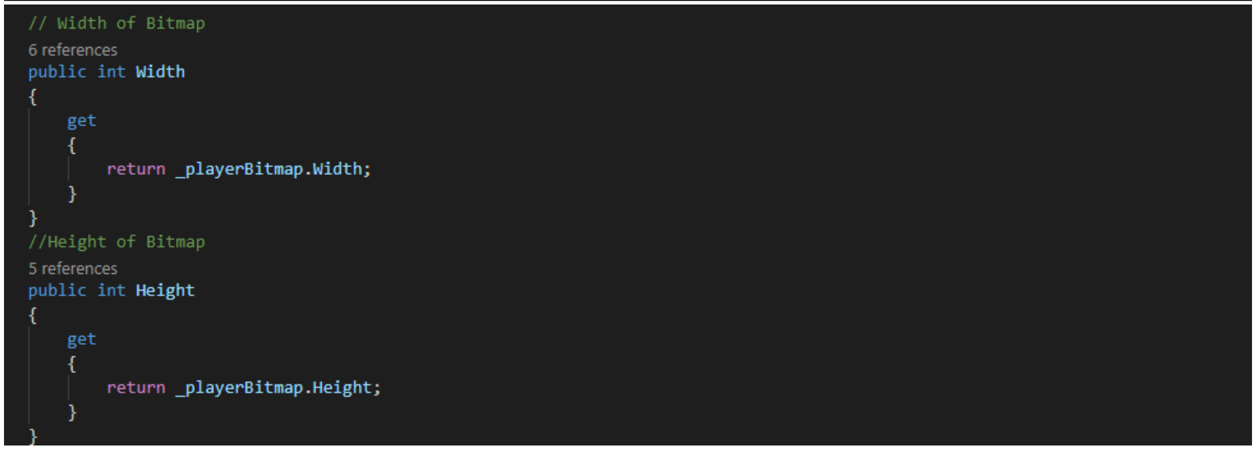


```
//Draw bullet
1 reference
public void Draw()
{

    SplashKit.DrawBitmap(_bulletBitmap, _x, _y);

}
```

In the player class, attributes of player such as the bitmap, the _angle (to draw angle for the ship), the position, Width and Height,… is defined.

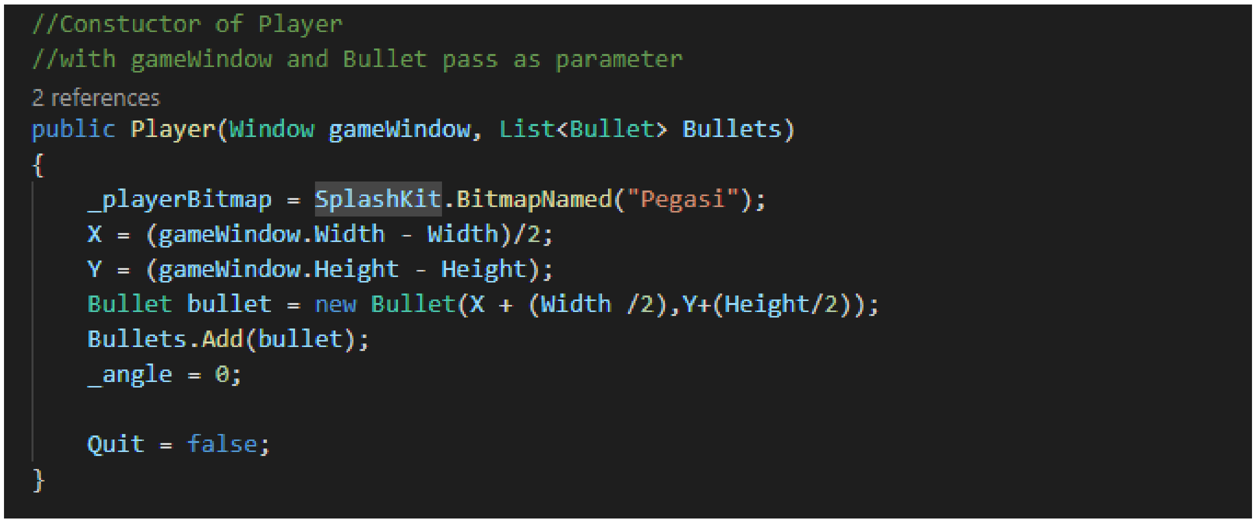


```
4 references
public class Player
{
    4 references
    private Bitmap _playerBitmap; // bitmap of player
    0 references
    private const int SPEED = 7;
    4 references
    private double _angle; //angle between player and mouse
    7 references
    public double X { get; private set;} // X position of player
    6 references
    public double Y { get; private set;} // Y position of player
    3 references
    public bool Quit {get; private set;} // whether player want to quit

    4 references
    public bool Shoot{get;set;} = true; // whether player allow to shoot (shoot is true when all bullets are back to player)
```



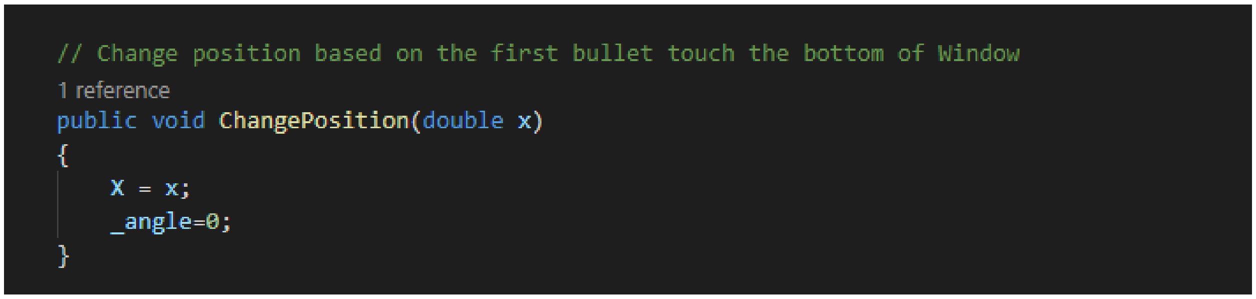
```
  // Width of Bitmap
  6 references
  public int Width
  {
      get
      {
          return _playerBitmap.Width;
      }
  }
  //Height of Bitmap
  5 references
  public int Height
  {
      get
      {
          return _playerBitmap.Height;
      }
  }
```

Player starts with the position in the middle of the screen at the bottom of window, and the angle 0 which points to the top edge



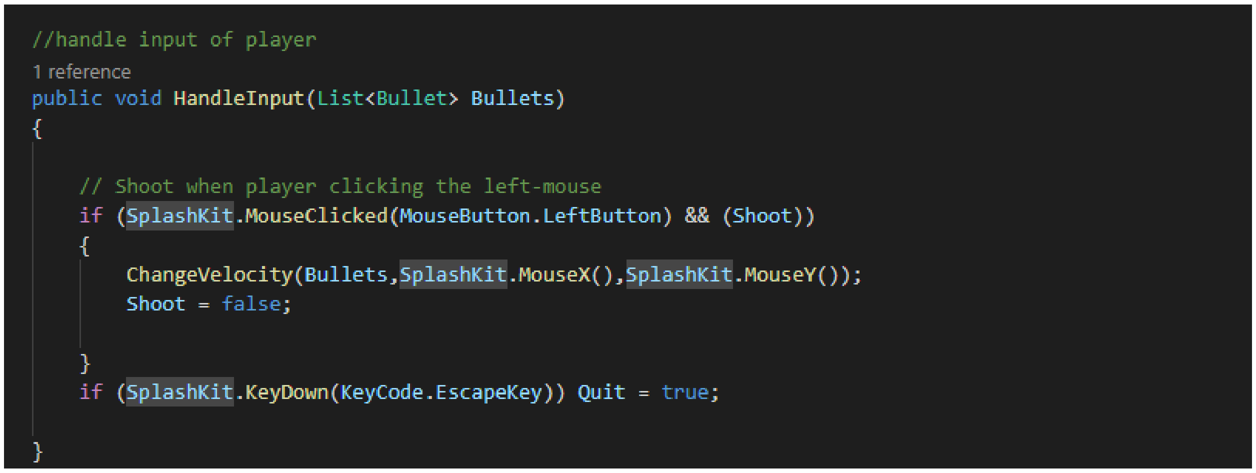
```
//Constuctor of Player
//with gameWindow and Bullet pass as parameter
2 references
public Player(Window gameWindow, List<Bullet> Bullets)
{
    _playerBitmap = SplashKit.BitmapNamed("Pegasi");
    X = (gameWindow.Width - Width)/2;
    Y = (gameWindow.Height - Height);
    Bullet bullet = new Bullet(X + (Width /2),Y+(Height/2));
    Bullets.Add(bullet);
    _angle = 0;

    Quit = false;
}
```

ChangePosition changes the x position of the player based on the X position which first bullet touch the bottom of screen



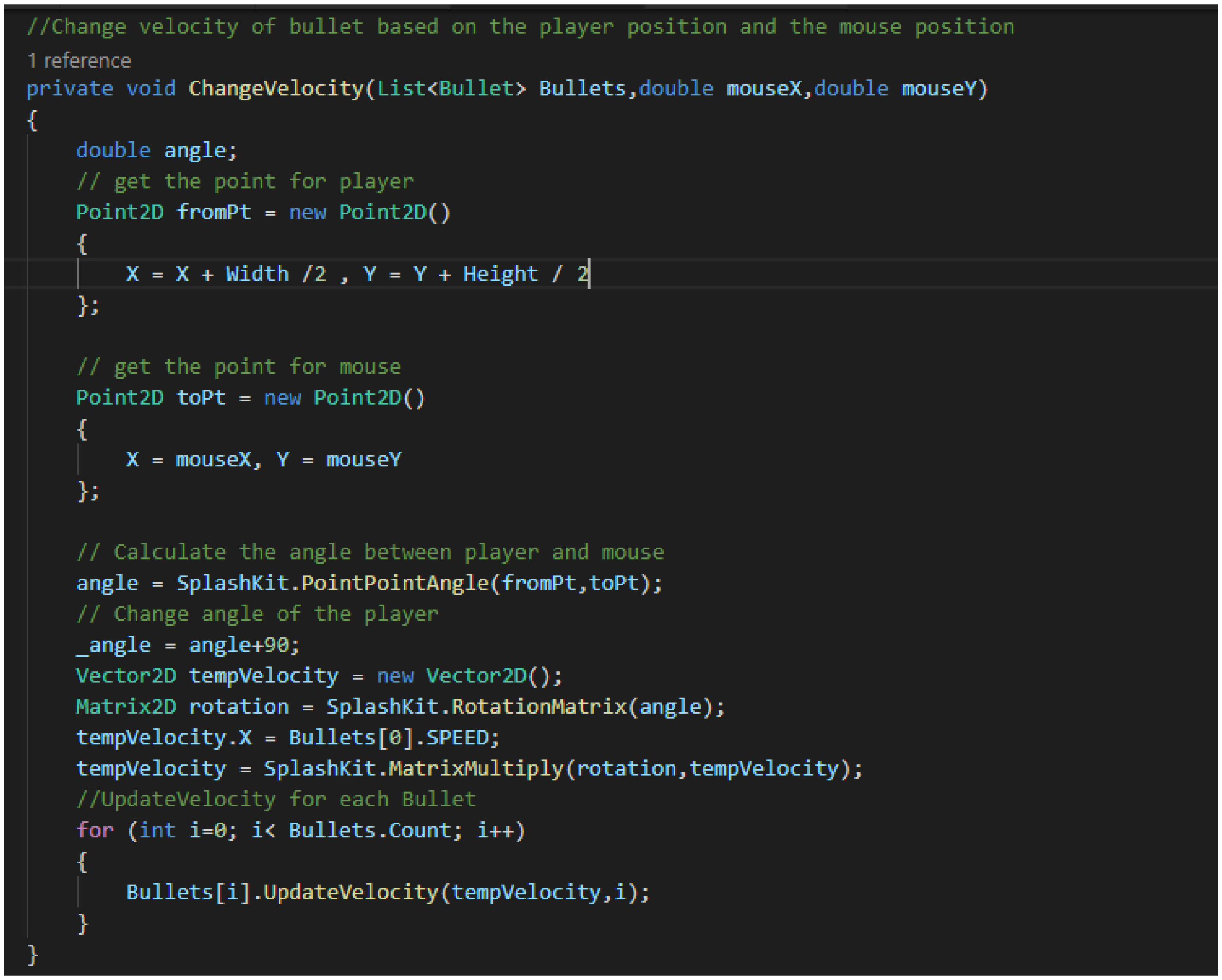
```
// Change position based on the first bullet touch the bottom of Window
1 reference
public void ChangePosition(double x)
{
    X = x;
    _angle=0;
}
```

HandleInput to trigger the input of user for shooting or quit the game



```
//handle input of player
1 reference
public void HandleInput(List<Bullet> Bullets)
{

    // Shoot when player clicking the left-mouse
    if (SplashKit.MouseClicked(MouseButton.LeftButton) && (Shoot))
    {
        ChangeVelocity(Bullets,SplashKit.MouseX(),SplashKit.MouseY());
        Shoot = false;

    }
    if (SplashKit.KeyDown(KeyCode.EscapeKey)) Quit = true;

}
```

ChangeVelocity calculates the angle between player and mouse clicked to change the velocity of bullet towards the mouse clicked, at the same time change the angle when draw bitmap player



```
//Change velocity of bullet based on the player position and the mouse position
1 reference
private void ChangeVelocity(List<Bullet> Bullets,double mouseX,double mouseY)
{
    double angle;
    // get the point for player
    Point2D fromPt = new Point2D()
    {
        X = X + Width /2 , Y = Y + Height / 2
    };

    // get the point for mouse
    Point2D toPt = new Point2D()
    {
        X = mouseX, Y = mouseY
    };

    // Calculate the angle between player and mouse
    angle = SplashKit.PointPointAngle(fromPt,toPt);
    // Change angle of the player
    _angle = angle+90;
    Vector2D tempVelocity = new Vector2D();
    Matrix2D rotation = SplashKit.RotationMatrix(angle);
    tempVelocity.X = Bullets[0].SPEED;
    tempVelocity = SplashKit.MatrixMultiply(rotation,tempVelocity);
    //UpdateVelocity for each Bullet
    for (int i=0; i< Bullets.Count; i++)
    {
        Bullets[i].UpdateVelocity(tempVelocity,i);
    }
}
```

So in summary, the code will execution as:

When Player clicks the mouse, all the bullets are shoot towards the mouse clicked position. If bullets touch any obstacle, they will bouncing.

If the bullets touch the Box, the value of box will decrease based on how many times bullets touch the box. When the box turns 0, the box disappears

If the box touches the plus symbol, next round, each new bullet will be added for any collision between bullet and plus symbol

When all bullet are back to player, the next round starts with add one row of box and increase the score

If any square box goes beyond the bottom screen, the game will finish