The Kinect Application using the Human Skeleton tracking

SM3603 Assignment 2

Tam Chin Pang
SID: 56226481
27 March 2022

Introduction

Motivation

Human skeleton tracking plays a significant role in the development of NUI (Nature User Interface). The Gesture, the posture or even the activity can be recognised using this skeleton-based detection. The mission in this assignment is to explore and develop an application based on this technique. Regarding the concept of the application, I was inspired by the current game called Vampire Survivors which is a time survival game with roguelite elements. Combing the simplified game strategy from it with the theme of battling with covid, I created a Kinect game called “KillVirus”.

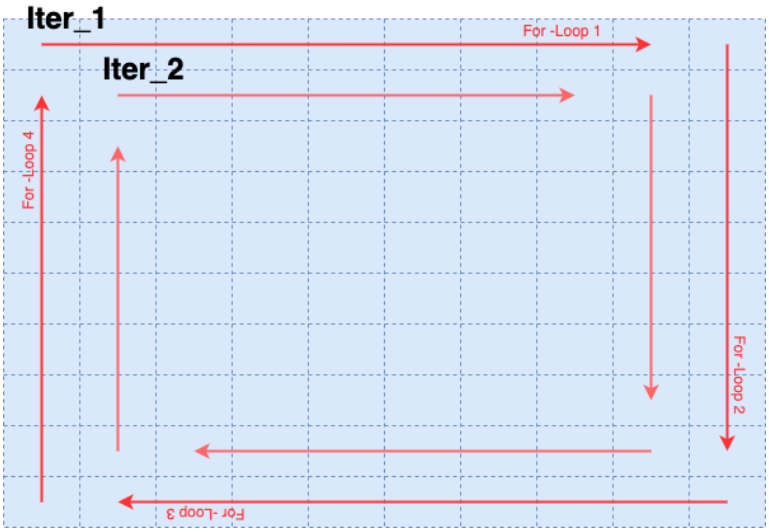
What-to-used

Kinect and visual studio are used as the hardware and the software in this project respectively.

Solution

Virus

Generating the virus




```
private void virusGeneration()   //respawn a customized no. of virus in customized order per second
{
    int generatedVirus = 0;
    for (int iter = 0; iter < row / 2; iter++)
    {
        if (generatedVirus >= GeneratePerSec)
        {
            break;
        }

        for (int x = iter; x < 9 - iter; x++)
        {
            if (generatedVirus >= GeneratePerSec)
            {
                break;
            }
            if (virus[iter * 10 + x].isGenerated == false)
            {
                virus[iter * 10 + x].color = ColorSelector();
                virus[iter * 10 + x].AbilityReset();
                virus[iter * 10 + x].isGenerated = true;
                generatedVirus++;
            }
        }

        for(int y = iter; y < 9 - iter; y++)
        {
            if (generatedVirus >= GeneratePerSec)
            {
                break;
            }
            if (virus[y * 10 + 9 - iter].isGenerated == false)
            {
                virus[y * 10 + 9 - iter].color = ColorSelector();
                virus[y * 10 + 9 - iter].AbilityReset();
                virus[y * 10 + 9 - iter].isGenerated = true;
                generatedVirus++;
            }
        }

        for(int x = 9 - iter; x > iter; x--)
        {
            if (generatedVirus >= GeneratePerSec)
            {
                break;
            }
            if (virus[(9 - iter) * 10 + x].isGenerated == false)
            {
                virus[(9 - iter) * 10 + x].color = ColorSelector();
                virus[(9 - iter) * 10 + x].AbilityReset();
                virus[(9 - iter) * 10 + x].isGenerated = true;
                generatedVirus++;
            }
        }

        for(int y = 9 - iter; y > iter; y--)
        {
            if (generatedVirus >= GeneratePerSec)
            {
                break;
            }
            if (virus[y * 10 + iter].isGenerated == false)
            {
                virus[y * 10 + iter].color = ColorSelector();
                virus[y * 10 + iter].AbilityReset();
                virus[y * 10 + iter].isGenerated = true;
                generatedVirus++;
            }
        }
    }
}
```

The position of every virus is evenly distributed in the 10*10 grid. As the diagram shown above, the virus will be generated in the order of the concentrated spheres' shape. In the virusGeneration(), the given number of virus will be selected to be spawned. Their attributes of color, isGenerated will be initialised.

Virus Spawn rate

```
VirusGenTimer.Interval = new TimeSpan(0, 0, 0, 3);
VirusGenTimer.Tick += VirusGenTimer_Tick;
VirusGenTimer.Start();
```

```
private void VirusGenTimer_Tick(object sender, EventArgs e)
{
    if (virus != null && IsStart)
    {
        virusGeneration();
    }
}
```

The generation of virus is controlled by the timer initialised once the Kinect is initialised properly. The virusGeneration() motioned above will be activated once in every 3 second if the array of virus is set up and the game is started.

Initialising the Virus

```
if(virus[10 * y + x].isGenerated)
{
    //dc.DrawRectangle(Brushes.Green, new Pen(Brushes.Green, 10), virus[10 * y + x].detectionRegion);
    virus[10 * y + x].Loadimage();
    virus[10 * y + x].drawVirus(dc);
    ability(dc, virus[10 * y + x], MapCameraPointToScreenSpace(body, JointType.HandLeft), MapCameraPointToScreenSpace(body, JointType.HandRight));
    virus[10 * y + x].updateState();
    virus[10 * y + x].updateRegionSize();
    virus[y * 10 + x].detectionRegion.X = x_interval * x + x_interval / 2 - virus[y * 10 + x].detectionRegion.Width/2;
    virus[y * 10 + x].detectionRegion.Y = y_interval * y + y_interval / 2 - virus[y * 10 + x].detectionRegion.Height/2;
   // DrawText(dc, virus[10 * y + x].color, new Point(x_interval * x + x_interval / 2 - virus[y * 10 + x].detectionRegion.Width / 2, y_interval * y
}
```

Initialisation of virus is done in every frame. If the virus' attribute of isGenerated is true, it will be initialised with a couple of functions: loading its image according to their color, drawing them on the canvas, updating their states, updating their detection regions according to their sizes and corresponded position.

States of Virus

```
1 reference
public void updateState()
{
    if(isGenerated)
    {
        if (size < 10)
        {
            size = 45;
            killed++;
            isGenerated = false;
        } else if(size >= 10  && size < max_size)
        {
            size += 1;
        } else if(size >= max_size)
        {
            size = max_size;
        }
    }
}
```

The state of virus is determined by its size. If its size is smaller than 10, it will be considered as “dead” and its isGenerated will return to false. If its size is larger than 10, it will be considered as “growing” and increase its size til its maximum size.

Types of Virus

```
private Random rand = new Random();
4 references
private string ColorSelector()
{
    double prob = rand.NextDouble();
    if (prob < 0.6)
    {
        return "green";
    }
    if(prob >= 0.6 && prob < 0.8)
    {
        return "red";
    }

    return "blue";
}
```

The type of the virus can be distinguished by their color. Different types of virus own different spawning probability. Green virus generated the most, having 60% chance to be spawned. Red and blue virus both own a 20% chance to be spawned.

Ability of Virus

```
1 reference
private void ability(DrawingContext dc, Virus virus, Point left, Point right)
{
    if(virus.color == "red")
    {
        virus.Emitlaser(dc, drawingImgWidth, heart);

    }
    if(virus.color == "blue")
    {
        virus.SpreadIce(dc, sprays, left, right);
    }
}
```

Different types of virus have different ability. The green one does not own any ability. The red one can emit red laser. The blue one can release ice region.

Laser Ability

```
1 reference
public void Emitlaser(DrawingContext dc, double width, Heart heart)
{

    left_laser.Height = 5;
    left_laser.Width += 10;
    left_laser.X = detectionRegion.X + size / 2 - left_laser.Width - detectionRegion.Width/2;
    left_laser.Y = detectionRegion.Y + size / 2 - right_laser.Height / 2;
    if (left_laser.Width >= 300)
    {
        left_laser.X = detectionRegion.X + size / 2;
        left_laser.Width = 0;
    }
    dc.DrawRectangle(Brushes.Red, new Pen(Brushes.Red, 1), left_laser);

    right_laser.Height = 5;
    right_laser.Width += 10;
    right_laser.X = detectionRegion.X + size / 2 + detectionRegion.Width/2;
    right_laser.Y = detectionRegion.Y + size / 2;
    if (right_laser.Width >= 300)
    {
        right_laser.Width = 0;
    }
    dc.DrawRectangle(Brushes.Red, new Pen(Brushes.Red, 1), right_laser);

    up_laser.Width = 5;
    up_laser.Height += 10;
    up_laser.X = detectionRegion.X + size / 2 - up_laser.Width / 2;
    up_laser.Y = detectionRegion.Y + size / 2 - up_laser.Height - detectionRegion.Height/2;
    if(up_laser.Height >= 300)
    {
        up_laser.Y = detectionRegion.Y + size / 2;
        up_laser.Height = 0;
    }
    dc.DrawRectangle(Brushes.Red, new Pen(Brushes.Red, 1), up_laser);

    down_laser.Width = 5;
    down_laser.Height += 10;
    down_laser.X = detectionRegion.X + size / 2 - up_laser.Width / 2;
    down_laser.Y = detectionRegion.Y + size / 2 + detectionRegion.Height / 2;
    if (down_laser.Height >= 300)
    {
        down_laser.Y = detectionRegion.Y + size / 2;
        down_laser.Height = 0;
    }
    dc.DrawRectangle(Brushes.Red, new Pen(Brushes.Red, 0), down_laser);

    laserIntersectHeart(heart);
}
```

```
1 reference
public void laserIntersectHeart(Heart heart)
{

    if ( (up_laser.IntersectsWith(heart.detectionRegion) || down_laser.IntersectsWith(heart.detectionRegion) || left_laser.IntersectsWith(heart.detectionRegion) || right_laser.IntersectsWith(heart.detectionRegion) ) && isHiting == false){
        heart.hp--;
        isHiting = true;
    }
    if(up_laser.IntersectsWith(heart.detectionRegion) == false && down_laser.IntersectsWith(heart.detectionRegion) == false && left_laser.IntersectsWith(heart.detectionRegion) == false && right_laser.IntersectsWith(heart.detectionRegion) == false)
    {
        isHiting = false;
    }
}
```

Every red virus will emit a 600*600 cross-shaped laser. If the heart of the player is hit by any of the laser, its hp will be deceased by one.

Ice Ability

```
1 reference
public void SpreadIce(DrawingContext dc, Spray[] sprays, Point left, Point right)
{
    iceArea.X = detectionRegion.X+size/2 - iceArea.Width/2;
    iceArea.Y = detectionRegion.Y+size/2 - iceArea.Height/2;
    iceArea.Height += 2;
    iceArea.Width += 2;
    if(iceArea.Height > 500)
    {
        iceArea.Height = 0;
        iceArea.Width = 0;
    }
    //dc.DrawRectangle(Brushes.Transparent, new Pen(Brushes.LightBlue, 5), iceArea);
    dc.DrawImage(iceCircle, iceArea);
    IceIntersectSpray(left, right, sprays);
}
```

```
1 reference
public void IceIntersectSpray(Point left, Point right, Spray[] sprays)
{
    if (iceArea.Contains(left))
    {
        sprays[0].IsFreezed = true;

    }
    if (iceArea.Contains(right))
    {
        sprays[1].IsFreezed = true;
    }
}
```

Every blue virus will release a 500*500 circular ice region. If any of the player's hand enter the region, the corresponded hand will be frozen and can't do any shooting.

```
1 reference
private void checkFreeze()
{
    if (sprays[0].IsFreezed && left_t_activated == false)
    {
        left_t_activated = true;
        System.Timers.Timer Left_t = new System.Timers.Timer(3000);
        Left_t.Elapsed += new System.Timers.ElapsedEventHandler(left_endFreeze);
        Left_t.AutoReset = false;
        Left_t.Enabled = true;
    }
    if (sprays[1].IsFreezed && right_t_activated == false)
    {
        right_t_activated = true;
        System.Timers.Timer right_t = new System.Timers.Timer(3000);
        right_t.Elapsed += new System.Timers.ElapsedEventHandler(right_endFreeze);
        right_t.AutoReset = false;
        right_t.Enabled = true;
    }
}
```

```
1 reference
private void left_endFreeze(object source, System.Timers.ElapsedEventArgs e)
{
    sprays[0].IsFreezed = false;
    left_t_activated = false;
}

1 reference
private void right_endFreeze(object source, System.Timers.ElapsedEventArgs e)
{
    sprays[1].IsFreezed = false;
    right_t_activated = false;
}
```

If any of the hands is detected as "frozen", a timer will be set up for it. After 3 second, the state of the corresponded hand will return to normal.

Spray

Drawing the Sprays

```
1 reference
private void drawSprays(Body body, DrawingContext dc)
{
    Point pt_lefthand = MapCameraPointToScreenSpace(body, JointType.HandLeft);
    sprays[0].loadImage(0);
    double left_ang = Dir2Angle(left_EW);
    if (sprays[0].spray != null)
    {
        //dc.DrawLine(new Pen(Brushes.Black, 50), pt_lefthand, pt_lefthand + left_hand_div*500);
        DrawRotatedImage(dc, sprays[0].spray, pt_lefthand, sprays[0].spray.Width, sprays[0].spray.Height, left_ang);
    }

    Point pt_righthand = MapCameraPointToScreenSpace(body, JointType.HandRight);
    sprays[1].loadImage(1);
    double right_ang = Dir2Angle(right_EW) - 180;
    if (sprays[1].spray != null)
    {
        //dc.DrawLine(new Pen(Brushes.Black, 50), pt_righthand, pt_righthand + right_hand_div * 500);
        DrawRotatedImage(dc, sprays[1].spray, pt_righthand, sprays[1].spray.Width, sprays[1].spray.Height, right_ang);
    }
}
```

Rotated Sprays will be drawn on the both hands in every frame. The sprays can rotate in relation to the moving angle of wrists.

Shooting of Sprays

```
1 reference
private void shoot(Body body, DrawingContext dc)
{
    if(sprays[0].IsFreezed == false)
    {
        //spray 0

        if (body.HandLeftState == HandState.Open && isShooting_left == false)
        {
            L_Shoot_t.Enabled = true;
            isShooting_left = true;
        }
        if (body.HandLeftState != HandState.Open)
        {
            L_Shoot_t.Enabled = false;
            isShooting_left = false;
        }
    }

    if(sprays[1].IsFreezed == false)
    {
        //spray 1

        if (body.HandRightState == HandState.Open && isShooting_right == false)
        {
            R_Shoot_t.Enabled = true;
            isShooting_right = true;
        }
        if (body.HandRightState != HandState.Open)
        {
            R_Shoot_t.Enabled = false;
            isShooting_right = false;
        }
    }
}
```

If the corresponded hand is not frozen and the player open that hand, the “shooting mode” will be enabled in the spray on corresponded hand.

Shooting Mode

```
L_Shoot_t.Elapsed += new System.Timers.ElapsedEventHandler(L_Shoot);
L_Shoot_t.AutoReset = true;
L_Shoot_t.Enabled = false;


R_Shoot_t.Elapsed += new System.Timers.ElapsedEventHandler(R_Shoot);
R_Shoot_t.AutoReset = true;
R_Shoot_t.Enabled = false;
```

```
1 reference
private void L_Shoot(object source, System.Timers.ElapsedEventArgs e)
{
    if (sprays[0].IsFreezed == false)
    {
        sprays[0].ShootAvailableBullet(pt_lefthand, left_hand_div);
    }

}
1 reference
private void R_Shoot(object source, System.Timers.ElapsedEventArgs e)
{
    if(sprays[1].IsFreezed == false)
    {
       sprays[1].ShootAvailableBullet(pt_righthand, right_hand_div);
    }

}
```

The timers is firstly initialised once the Kinect is initialised properly as the global objects. They will be activated only when the sprays enter their shooting modes. In every time loop, the shootAvailableBullet() is used to shoot the drops of alcohol out of the sprays automatically.

Bullet

In every spray, an array of bullets is initialised as its variable.

Available Bullet

```
public void ShootAvailableBullet(Point shootingPoint, Vector div)
{
    for(int i=0; i<bullet.Length; i++)
    {
        if (bullet[i].isShooted == false)
        {
            bullet[i].isShooted = true;
            bullet[i].location.X = shootingPoint.X;
            bullet[i].location.Y = shootingPoint.Y;
            bullet[i].setDir(div);
            availableBullet--;
            break;
        }
    }
}
```

In the function, an available (isShooted is false) bullet will be searched from its array to shoot. If no available bullet is found, nothing will be shot.

Update of Bullets

```
1 reference
private void updateBullets()
{
    sprays[0].VirusIntersectBullet(virus);
    sprays[1].VirusIntersectBullet(virus);

    sprays[0].updateBullet(drawingImgWidth, drawingImgHeight);
    sprays[1].updateBullet(drawingImgWidth, drawingImgHeight);
}
```

```
2 references
public void updateBullet(double drawingImgWidth, double drawingImgHeight)
{
    for(int i = 0; i<bullet.Length; i++)
    {

        if (bullet[i].isShooted)
        {
            if (bullet[i].location.X < 0 || bullet[i].location.X > drawingImgWidth || bullet[i].location.Y < 0 || bullet[i].location.Y > drawingImgHeight)
            {
                bullet[i].reset();
                bullet[i].isShooted = false;
                availableBullet++;
            }
            else
            {
                bullet[i].location += bullet[i].moving_dir * 40;
            }
        }
    }
}
```

If a bullet moves out of the screen or hits any of the virus, it will be reset and thus returns to an available bullet to be shot. If not, the bullet will continue to move with its original direction.

Intersection of Bullet and Virus

```
2 references
public void VirusIntersectBullet(Virus[] virus)
{
    for(int i=0; i<virus.Length; i++)
    {
        if (virus[i].isGenerated)
        {
            for(int j=0; j<bullet.Length; j++)
            {
                if (virus[i].detectionRegion.Contains(bullet[j].location))
                {
                    virus[i].isShooted();
                    bullet[j].reset();
                    bullet[j].isShooted = false;
                    availableBullet++;
                }
            }
        }
    }
```

If the bullet hits the detection region of the virus, isShooted() is activated to reduce the virus' size and reset the corresponded bullet.

Draw the Bullets

```
1 reference
private void drawBullets(DrawingContext dc)
{
    sprays[0].drawBullet(dc);
    sprays[1].drawBullet(dc);
}
```

After doing all the checking and updates, the program will draw all the bullets on their corresponded locations on the canvas.

Objective of the Game

Game Level & Wining

```
1 reference
private void updateLv()
{
    NumOfKilledVirus = 0;
    for(int i = 0; i < virus.Length; i++)
    {
        NumOfKilledVirus += virus[i].killed;
    }

    if(NumOfKilledVirus < 5)
    {
        gameLevel = 0;
        GeneratePerSec = 1;
    } else if (NumOfKilledVirus >= 5 && NumOfKilledVirus < 15)
    {
        gameLevel = 1;
        GeneratePerSec = 2;
    } else if(NumOfKilledVirus >= 15 && NumOfKilledVirus < 30)
    {
        gameLevel = 2;
        GeneratePerSec = 3;
    } else if(NumOfKilledVirus >= 30 && NumOfKilledVirus < 60)
    {
        gameLevel = 3;
        GeneratePerSec = 5;
    } else if(NumOfKilledVirus >= 60 && NumOfKilledVirus < 100)
    {
        gameLevel = 4;
        GeneratePerSec = 7;
    } else if(NumOfKilledVirus >= 100)
    {
        IsWin = true;
    }
}
```

In the function, the program will find the latest total number of killed virus. The game level and spawning rate of virus will increase if the player kills enough amount of virus. The player can win the game once he/she kill 100 virus.

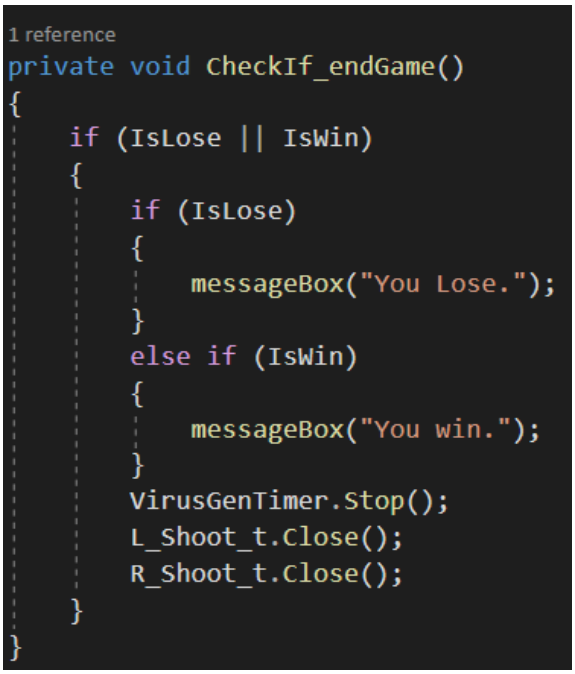
Losing

```
if (IsLose == false)
{
    heart.HeartInit(dc, MapCameraPointToScreenSpace(body, JointType.SpineMid));
    IsLose = heart.VirusIntersectHeart(virus);
}
else
{
    heart.reset();
}
```

```
if(heart.hp <= 0)
{
    IsLose = true;
}
CheckIf_endGame();
```

There are two ways to lose the game. First, the player immediately lose the game if his/her heart touches any of the virus. Another way to lose is that the player's heart has been hit by laser for several times, meaning that the size of the heart is reduced to zero.

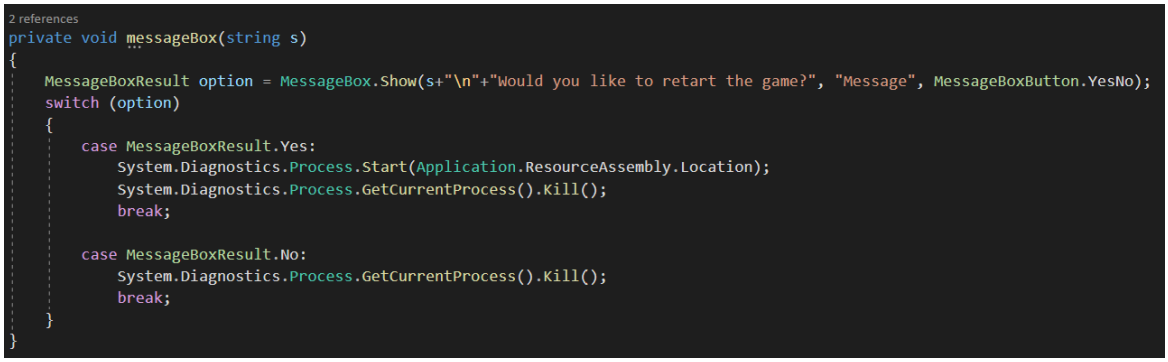
Checking State of the Game



```
1 reference
private void CheckIf_endGame()
{
    if (IsLose || IsWin)
    {
        if (IsLose)
        {
            messageBox("You Lose.");
        }
        else if (IsWin)
        {
            messageBox("You win.");
        }
        VirusGenTimer.Stop();
        L_Shoot_t.Close();
        R_Shoot_t.Close();
    }
}
```

Once a player win or lose the game, a message box will be popped out and all the timers will be closed.

MessageBox



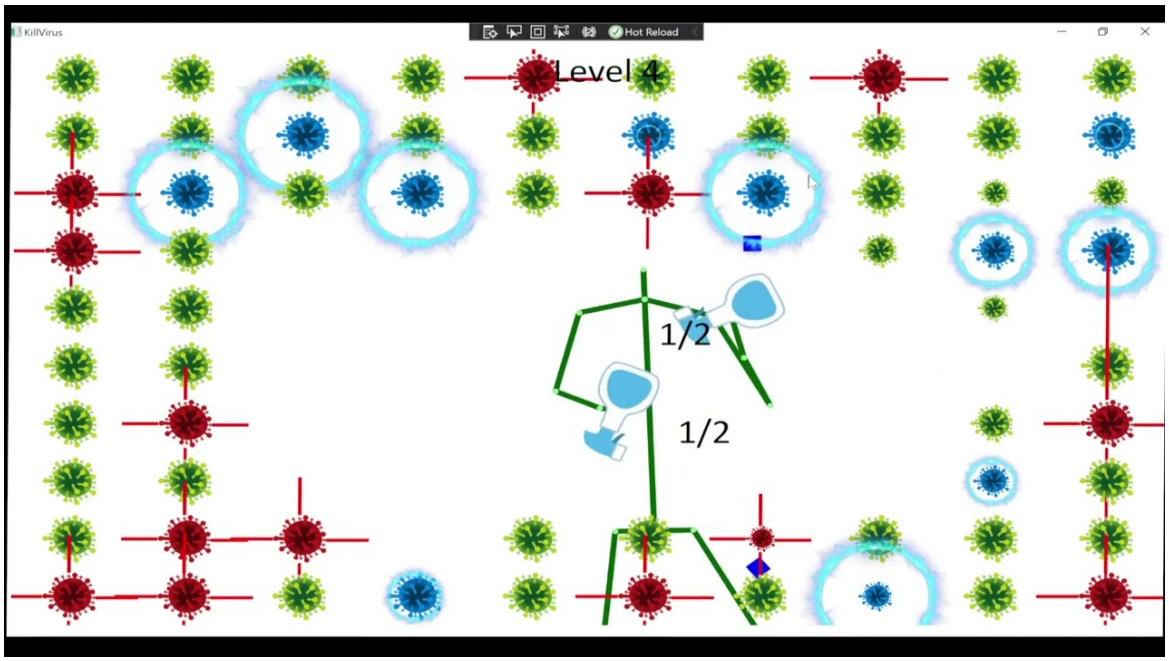
```
2 references
private void messageBox(string s)
{
    MessageBoxResult option = MessageBox.Show(s+"\n"+"Would you like to retart the game?", "Message", MessageBoxButton.YesNo);
    switch (option)
    {
        case MessageBoxResult.Yes:
            System.Diagnostics.Process.Start(Application.ResourceAssembly.Location);
            System.Diagnostics.Process.GetCurrentProcess().Kill();
            break;

        case MessageBoxResult.No:
            System.Diagnostics.Process.GetCurrentProcess().Kill();
            break;
    }
}
```

The message box will ask the player if they want to close or restart the game.

Demo Video

Video Link

youtu.be/HeCPP9Rrm7Y



Reference

Code

T7_AugmentedHuman

T8_PoseMatching

Image

https://www.irasutoya.com/