```
using BasicAI;
using Final_Project.Properties;
using Microsoft.VisualBasic.Devices;
using System.Transactions;

//programmer: Aaron Yoon
//project: Final Project
//Date: 5/25/23
namespace Final_Project
{
    public partial class Form1 : Form
    {

        private static Form1 instance = new Form1();

        //declare all variables
        int playerSpeed = 10;
        int test = 0;
        String movementKey = "";
        String modifierKey = "";

        int kills = 0;

        List<PictureBox> heartList = new List<PictureBox>();
        int heartCount = 0;

        Label[] projectiles = new Label[5];
        int bulletCount = 0;

        Boolean timeFrozen = false;
        Boolean timeRegen = false;

        System.Random r = new System.Random((int)
          System.DateTime.Now.Ticks);

        //variable to control enemies
        Enemy[] enemies = new Enemy[5];

        Enemy enemy0;
        Enemy enemy1;
        Enemy enemy2;
        Enemy enemy3;
        Enemy enemy4;

        Gunner boss;
        RoundState currentRound = RoundState.ROUND_1;
        WeaponSelected currentWeapon = WeaponSelected.SWORD;

        //a way to show what the current round is
        private enum RoundState
        {
            ROUND_1,
            ROUND_2,
            ROUND_3,
            BOSS
        }

        //a way to show the current weapon selected
        public enum WeaponSelected
        {
            SWORD,
            SHIELD,
        }
        public Form1()
        {

            InitializeComponent();

            //enemy0 = new Enemy(picEnemy0, player, projectiles);

            //private static Form1 instance = new Form1();

            //give values to all variables
            projectiles[0] = projectile1;
            projectiles[1] = projectile2;
            projectiles[2] = projectile3;
            projectiles[3] = projectile4;
            projectiles[4] = projectile5;

            enemy0 = new Enemy(picEnemy0, player, projectiles, 0.015, 3);
            enemy1 = new Enemy(picEnemy1, player, projectiles, 0.015, 3);
            enemy2 = new Enemy(picEnemy2, player, projectiles, 0.015, 3);
            enemy3 = new Enemy(picEnemy3, player, projectiles, 0.015, 3);
            enemy4 = new Enemy(picEnemy4, player, projectiles, 0.015, 3);

            boss = new Gunner(picBoss, player, bossBullet, projectiles,
              bossHealthBar, 0.013, 400, health);

            boss.width = this.Width;
            boss.height = this.Height;

            enemies[0] = enemy0;
            enemies[1] = enemy1;
            enemies[2] = enemy2;
            enemies[3] = enemy3;
            enemies[4] = enemy4;


            //adjustEnemies(1, 0.015);
            initRound();
        }

        public static Form1 getInstance()
        {
            return instance;
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            //boss.respawn(0, 0);

            Update.Enabled = true;
            itemTimer.Enabled = true;

            stamina.Width = 400;
            health.Width = 400;


        }

        private void movePlayer(String key)
        {
            //makes the player move depending on which key and if
              sprinting

            if (modifierKey == "Shift" && stamina.Width != 0)
            {
                staminaTimer.Enabled = false;
                playerSpeed = 20;
                stamina.Width -= 4;
            }
            else if (modifierKey == "None" && stamina.Width != 0)
            {
                staminaTimer.Enabled = true;
                playerSpeed = 10;
            }
            else if (stamina.Width == 0)
            {
                staminaTimer.Enabled = true;
                playerSpeed = 5;
            }


            switch (key)
            {
                case "w":
                    player.Top -= playerSpeed;
                    break;
                case "a":
                    player.Left -= playerSpeed;
                    break;
                case "s":
                    player.Top += playerSpeed;
                    break;
                case "d":
                    player.Left += playerSpeed;
                    break;

                case "W":
                    player.Top -= playerSpeed;
                    break;
                case "A":
                    player.Left -= playerSpeed;
                    break;
                case "S":
                    player.Top += playerSpeed;
                    break;
                case "D":
                    player.Left += playerSpeed;
                    break;


            }
        }

        private double getAngle()
        {
            //used to get the angle of your cursor in relation to the 
              player
            int mouseX = Cursor.Position.X;
            int mouseY = Cursor.Position.Y;

            int playerX = player.Left + (player.Width / 2);
            int playerY = player.Top + (player.Height / 2) + 30;

            double deltaX = mouseX - playerX;
            double deltaY = playerY - mouseY;

            try
            {

            double angle = Math.Atan(deltaY / deltaX) + (Math.PI / 2);

                if (deltaX < 0)
                    return Math.PI + angle;
                else
                    return angle;
            }
            catch
            {
                return 0;
            }

        }

        private void drawArc(double angle)
        {
            //draws an arc (shield) around the player
            double referenceAngle = angle - (Math.PI / 10);

            double x;
            double y;

            try
            {
                for (int i = 0; i < projectiles.Length; i++)
                {
                    x = Math.Sin(referenceAngle);
                    y = Math.Cos(referenceAngle);

                    projectiles[i].Left = player.Left + (player.Width / 2) 
                    + (int)(x * 100);
                    projectiles[i].Top = player.Top + (player.Height / 2)  
                   + (int)((y * 100));

                    referenceAngle += Math.PI / 20;
                }
            }
            catch { }

            //label1.Text = x.ToString() + "\n" + y.ToString();
        }

        private void drawSword(double angle)
        {
            //draws a stick (sword) around the player
            double x = Math.Sin(angle);
            double y = Math.Cos(angle);

            try
            {
                for (int i = 0; i < projectiles.Length; i++)
                {

                    projectiles[i].Left = player.Left + (player.Width / 2) 
                    + (int)(x * 100);
                    projectiles[i].Top = player.Top + (player.Height / 2)
                   + (int)(y * 100);

                    //label1.Text = x.ToString() + "\n" + y.ToString();
                    x *= 1.1;
                    y *= 1.1;
                }
            }
            catch { }
        }

        //
          ================================================================
          ========================================================
        private void timer1_Tick(object sender, EventArgs e)
        {
            //update timer for everything
            try
            {
                movePlayer(movementKey);
                //drawArc(getAngle());

                if(currentWeapon == WeaponSelected.SWORD)
                    drawSword(getAngle());
                else if(currentWeapon == WeaponSelected.SHIELD)
                    drawArc(getAngle());

                heal();


                label1.Text = getKills().ToString();

                if(!timeFrozen)
                    updateEnemies();

                if (currentRound == RoundState.BOSS)
                {

                    boss.Update();
                    boss.dynamicWaypoint(this.Width, this.Height);

                    if (boss.getHealth() == 0)
                    {

                        btnReset.Visible = true;
                        btnReset.Enabled = true;
                        Update.Enabled = false;

                        MessageBox.Show("The tyrant has been slain!");

                    }
                }

                changeRound();
            }

            catch
            {
                MessageBox.Show("How did we get here?");
            }
        }
        //===============================================================================================================================


        private void changeRound()
        {
            //change round based off of how many kills you have
            if(currentRound == RoundState.ROUND_1 && getKills() >= 10)
            {
                currentRound = RoundState.ROUND_2;
                initRound();
            }
            else if(currentRound == RoundState.ROUND_2 && getKills() >= 25)
            {
                currentRound = RoundState.ROUND_3;
                initRound();
            }
            else if (currentRound == RoundState.ROUND_3 && getKills() >= 45)
            {
                currentRound = RoundState.BOSS;
                initRound();
            }


        }

        private void initRound()
        {
            //initialize the round for what it needs
            switch(currentRound)
            {
                case RoundState.ROUND_1:
                    adjustEnemies(2, 0.015);
```

```
                        RoundIndicator.BackColor = Color.Lime;
                        changeImage("happy.png");
                        break;
                    case RoundState.ROUND_2:
                        adjustEnemies(3, 0.020);
                        RoundIndicator.BackColor = Color.Orange;
                        changeImage("meh.png");
                        break;
                    case RoundState.ROUND_3:
                        adjustEnemies(4, 0.03);
                        RoundIndicator.BackColor = Color.Red;
                        changeImage("angy.png");
                        break;
                    case RoundState.BOSS:
                        adjustEnemies(4, 0.03);
                        RoundIndicator.BackColor = Color.Black;
                        changeImage("demon.png");
                        arbi.Visible = true;
                        boss.respawn(0, 0);
                        break;
                }
            }

            private void changeImage(String name)
            {
                //used to mass change the images of all enemies
                for(int i = 0; i < enemies.Length; i++)
                {
                    enemies[i].picEnemy.Image = Image.FromFile(name, true);
                }
            }

            private void adjustEnemies(int cap, double spd)
            {
                //change the speed and health of the enemies
                for(int i = 0; i < enemies.Length; i++)
                {
                    enemies[i].healthcap = cap;
                    enemies[i].speed = spd;
                }
            }

            private void refillStamina()
            {
                //used to refill the stamina
                //eventTimer.Enabled = true;
                staminaTimer.Interval = 40;

                if (stamina.Width < 400)
                    stamina.Width += 10;
                else
                {
                    staminaTimer.Enabled = false;
                    staminaTimer.Interval = 1500;
                }

            }

            private void heal()
            {
                //used to automatically check if you're able to grab a heart to heal
                for(int i = 0; i < heartList.Count; i++)
                {
                    if (player.Bounds.IntersectsWith(heartList.ElementAt(i).Bounds) && health.Width < 400)
                    {
                        health.Width += 40;
                        this.Controls.Remove(heartList.ElementAt(i));
                        heartList.RemoveAt(i);
                        heartCount--;
                    }
                }
            }

            private void updateEnemies()
            {
                //a way to mass update all enemies and to automatically take damage
                for (int i = 0; i < enemies.Length; i++)
                {
                    if (enemies[i].enemyTouch())
                    {
                        health.Width -= 80;
                    }

                    enemies[i].Update();
                }

                if (health.Width <= 0)
                {
                    btnReset.Visible = true;
                    btnReset.Enabled = true;
                    Update.Enabled = false;

                    MessageBox.Show("Game Over!");
                }
            }

            private double getKills()
            {
                //displays the amount of kills you have
                kills = 0;

                for (int i = 0; i < enemies.Length; i++)
                {
                    kills += enemies[i].getDeaths();
                }

                return kills;
                //label1.Text = kills.ToString();//enemy0.getHealth().ToString
                  ();//enemy0.swordTouch().ToString();
            }
            public void spawnHeart()
            {
                //gives a random chance to spawn a heart
                heartList.Add(new PictureBox());

                this.Controls.Add(heartList.ElementAt(heartCount));
                heartList.ElementAt(heartCount).Height = 50;
                heartList.ElementAt(heartCount).Width = 50;
                heartList.ElementAt(heartCount).SizeMode =
                  PictureBoxSizeMode.StretchImage;
                heartList.ElementAt(heartCount).Image = Image.FromFile
                  ("heart.png", true);
                heartList.ElementAt(heartCount).Left = r.Next(0, this.Width -
                  50);
                heartList.ElementAt(heartCount).Top = r.Next(0, this.Height -
                  50);

                heartCount++;
            }
            private void Form1_KeyDown(object sender, KeyEventArgs e)
            {
                modifierKey = Control.ModifierKeys.ToString();


            }

            private void reset()
            {
                //reset everything back to round 1
                currentRound = RoundState.ROUND_1;
                initRound();

                boss.reset();
                arbi.Visible = false;

                health.Width = 400;

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

                btnReset.Visible = false;
                btnReset.Enabled = false;

                Update.Enabled = true;


            }

            private void player_Click(object sender, EventArgs e)
            {

            }

            private void Form1_KeyUp(object sender, KeyEventArgs e)
            {
                //movementKey = 0;
                modifierKey = Control.ModifierKeys.ToString();

                String keyValue = "";

                if (modifierKey == "Shift")
                {

                    switch (e.KeyValue)
                    {
                        case 87:
                            keyValue = "W";
                            break;
                        case 65:
                            keyValue = "A";
                            break;
                        case 83:
                            keyValue = "S";
                            break;
                        case 68:
                            keyValue = "D";
                            break;
                    }
                }

                else if (modifierKey == "None")
                {

                    switch (e.KeyValue)
                    {
                        case 87:
                            keyValue = "w";
                            break;
                        case 65:
                            keyValue = "a";
                            break;
                        case 83:
                            keyValue = "s";
                            break;
                        case 68:
                            keyValue = "d";
                            break;
                    }
                }

                if (movementKey == keyValue)
                {
                    movementKey = "";
                }
            }

            private void Form1_KeyPress(object sender, KeyPressEventArgs e)
            {


                movementKey = e.KeyChar.ToString();

                //label1.Text = movementKey;
                //label1.Text = Control.ModifierKeys.ToString();
            }

            private void Form1_PreviewKeyDown(object sender,
              PreviewKeyDownEventArgs e)
            {

            }

            private void stamina_Click(object sender, EventArgs e)
            {

            }

            private void staminaTimer_Tick(object sender, EventArgs e)
            {
                refillStamina();
            }

            private void Form1_MouseClick(object sender, MouseEventArgs e)
            {
                //change weapons if you click
                if (currentWeapon == WeaponSelected.SWORD)
                {
                    currentWeapon = WeaponSelected.SHIELD;
                    picWeapon.Image = (Image)Resources.shield;

                    for(int i = 0; i < enemies.Length; i++)
                    {
                        enemies[i].currentWeapon = currentWeapon;

                    }

                    boss.currentWeapon = currentWeapon;
                }
                else if (currentWeapon == WeaponSelected.SHIELD)
                {
                    currentWeapon = WeaponSelected.SWORD;
                    picWeapon.Image = (Image)Resources.sword1;
                    for (int i = 0; i < enemies.Length; i++)
                    {
                        enemies[i].currentWeapon = currentWeapon;

                    }

                    boss.currentWeapon = currentWeapon;
                }
            }

            private void itemTimer_Tick(object sender, EventArgs e)
            {
                //used to randomly spawn a heart
                int proc = r.Next(0, 100);

                if(proc == 3)
                {
                    spawnHeart();
                }
            }

            private void quitToolStripMenuItem_Click(object sender, EventArgs
              e)
            {
                this.Close();
            }

            private void fileToolStripMenuItem_Click(object sender, EventArgs
              e)
            {
                //Update.Enabled = false;
            }

            private void pauseUnpauseToolStripMenuItem_Click(object sender,
              EventArgs e)
            {
                //toggle update timer
                if (Update.Enabled)
                    Update.Enabled = false;
                else
                    Update.Enabled = true;
            }

            private void button1_Click(object sender, EventArgs e)
            {

            }

            private void yBox_TextChanged(object sender, EventArgs e)
            {

            }

            private void xBox_TextChanged(object sender, EventArgs e)
            {

            }

            private void Form1_MouseDoubleClick(object sender, MouseEventArgs
              e)
            {
                if (timeIcon.Height >= 150)
                {

                    stopTime.Enabled = true;
                }
            }

            private void stopTime_Tick(object sender, EventArgs e)
            {
                //used to stop time
                if (!timeRegen)
                {
                    timeFrozen = true;

                    timeIcon.Height -= 5;
                }

                if (timeRegen && timeIcon.Height < 150)
                {
                    timeIcon.Height += 2;
                }
                else if(timeIcon.Height >= 150)
                {
                    timeRegen = false;
                    stopTime.Enabled = false;
                }

                if(timeIcon.Height <= 0)
                {
                    timeFrozen = false;
                    timeRegen = true;
                }
            }

            private void bossHealthBar_Click(object sender, EventArgs e)
            {

            }

            private void btnReset_Click(object sender, EventArgs e)
            {
                reset();
            }

            private void btnReset_Click_1(object sender, EventArgs e)
            {
                reset();
            }

            private void newGameToolStripMenuItem_Click(object sender,
              EventArgs e)
            {
                reset();
            }

            private void aboutToolStripMenuItem_Click(object sender, EventArgs
               e)
            {
                //when about is clicked, this message box shows
                MessageBox.Show(
                    "Hello! This game simulates you, a knight armed with a
                      sword and shield, defending his homeland against the
                      evil monsters!" + "\n"
                    + "Use WASD to move and shift to sprint, though becareful!
```

```
                         Once you're out of stamina you become exhausted and 
                        cannot walk as fast." + "\n"
                      + "Use your mouse to control where you sword / shield is, 
                        and left click to swap weapons. Double click to stop 
                        time briefly." + "\n"
                      + "overtime, there are chances for hearts to spawn around 
                        the map. Pick them up for a small health boost." + "\n"
                      + "That's all you need to know, now beat that tyrant!"
                      );
            }
        }
}
```