Game Framework Project

Submitted To:

Dr. Awais Hassan

Submitted By:

Muhammad Safee ullah, 2020-CS-13

Department of Computer Sicence

University of Engineering and Technonlogy Lahore

Framework for Making Platformer Games

Table of Contents

Problem Statement:

We need to add dynamic movement behavior. Such that we can make a class for movement and add it to our pobejct and the behavior is applied to the object.

Previous Solution:

This code defines the movement of the player and enemies in their own classes and we cannot change them outside of the object.

Demerits:

The only demerit I can think of is that you have to add a new enemy type if you ever want to change the behaviour.

Code:

Player Movement Code:

```
public void update()
        {

            double elapsedTime = (DateTime.Now - previousTime).TotalMilliseconds/100;
            if (elapsedTime > 2) elapsedTime = 2;

            if(walkingLeft && this.Left > 10)
            {
                Left -= (int)(PlayerSpeed * elapsedTime);
            }
            if (walkingRight && this.Right < gameForm.Bounds.Right - 10)
            {
                Left += (int)(PlayerSpeed * elapsedTime);
            }
            if(jumping && jumpSpeed > -35)
            {
                jumpSpeed = -35;
                jumping = false;
            }
            else
            {
                if (fallingTrigger && jumpSpeed != 0)
                {
                    if(walkingLeft) Image = 
dragonSlayer.Properties.Resources.KnightFallingBackward;
                    else Image = dragonSlayer.Properties.Resources.KnightStartFalling;
                    fallingTrigger = false;
                }
                jumpSpeed += gAcceleration;
                gAcceleration += 5;
                if (jumpSpeed > 40) jumpSpeed = 40;
            }
            Top += jumpSpeed;
            detectCollision();
            if (slash.Right < gameForm.Right - 10) slash.Left += 40;
            else slash.Top = gameForm.Bottom;

            if (blast.Right < gameForm.Right - 10) blast.Left += 30;
            else blast.Top = gameForm.Bottom;
            if(health<=0) ((MainGame)gameForm).end("You Died!", this.score);
            if (blastCounter > 2 && ((MainGame)gameForm).EnemyHealth > 2)
((MainGame)gameForm).end("You Cannot beat the wizard anymore!", this.score);
            previousTime = DateTime.Now;
        }
        public bool detectCollision()
        {
            foreach (Control control in gameForm.Controls)
            {
                if (control is PictureBox && (string)control.Tag == "ground" &&
control.Bounds.IntersectsWith(this.Bounds) && jumping == false)
                {
                    gAcceleration = 0;
                    Top = control.Bounds.Top - Height;
                    if (jumping) jumping = false;
                    jumpSpeed = 0;
                    fallingTrigger = true;
                    walkingTrigger = true;
                    return true;
                }
                if(control is PathFinder && control.Bounds.IntersectsWith(slash.Bounds))
                {
                    slash.Top = gameForm.Bottom;
                    score += 50;
                    ((PathFinder)control).Health--;
                    ((PathFinder)control).Image =
dragonSlayer.Properties.Resources.DragonHit;
                }
                if (control is PathFinder && control.Bounds.IntersectsWith(Bounds))
                {
                    health--;
                }
                if (control is PictureBox && control.Name == "Wizard" &&
control.Bounds.IntersectsWith(blast.Bounds))
                {
                    score += 200;
                    blast.Top = gameForm.Bottom;
                    ((MainGame)control.FindForm()).EnemyHealth--;
                }
                if (control is PictureBox && control.Name.Contains("power") &&
control.Bounds.IntersectsWith(Bounds))
                {
                    blastCounter++;
                    gameForm.Controls.Remove(control);
                    canBlast = true;
                }
            }
            return false;
        }
```

EnemyMovement Code (Excluding the pathFinding Algorithm):

```
public void followPlayer()
        {

            if (Health <= 0)
            {
                this.Enable = false;
                MapForm.Controls.Remove(this);
                this.Dispose();
            }
            if(followCounter++ % 4 == 0)
            {
                List<Point> path = runAstar();
                if(path.Count > 1)
                {

                    if(this.Left < Target.Left) Image = 
dragonSlayer.Properties.Resources.DragonFlyingRight;
                    else Image = dragonSlayer.Properties.Resources.DragonFlyingLeft;

                    this.SetBounds(path[path.Count - 2].X * CellSideLength, 
path[path.Count - 2].Y * CellSideLength, Width, Height);
                }
            }
        }
```

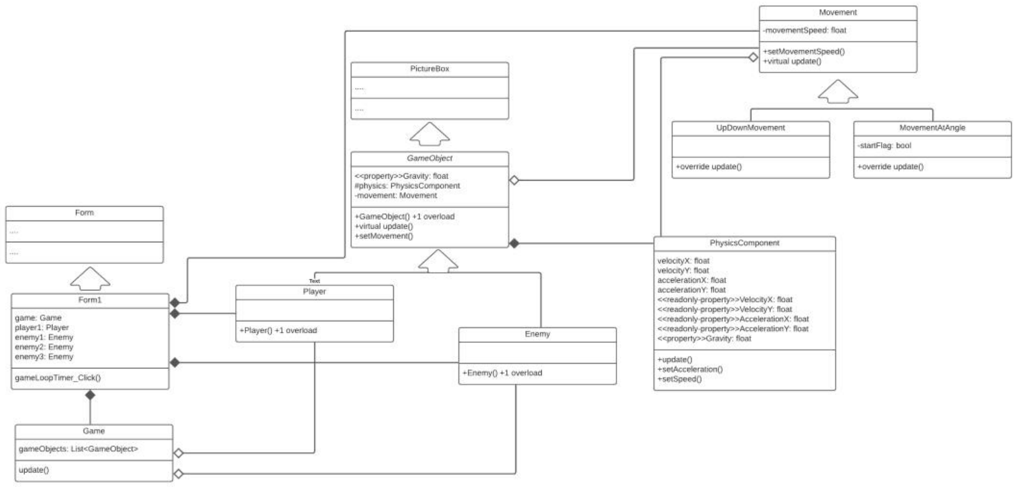
Current Solution:

Design Decision:

Here we have 2 choices:

1. Make a virtual function for updating movement in GameObject and write our movement code in there.
   But the demerits of this approach is that we cannot reuse our movement code and have to write it separately for every class.
2. Second solution makes much more sense it is to make separate classes for movement and pass the object of these classes to our gameobjects for handling the behavior.
   In this scenario we can just make change to our movement classes without touching the code I the gameObject classes and it'll change everywhere automatically.

UML Diagram:



Merits/Demerits:

1. This approach saves us from a bit of repetitive code
2. The code is a lot more manageable since we can just modify the movement classes without having to go through the code of gameObjects.
3. We can add complex behaviuor to out objects without having to add all the code to them
4. The code is more readable and understandable.

Code:

Form1:

```
public partial class Form1 : Form
    {
        Game game;
        public Form1()
        {
            InitializeComponent();
            game = new Game();
            game.addGameObject(player1);
            game.addGameObject(enemy1);
            game.addGameObject(enemy2);
            game.addGameObject(enemy3);
        }

        private void gameLoopTimer_Tick(object sender, EventArgs e)
        {
            game.update();
        }
    }
```

Game:

```
class Game
    {
        List<GameObject> gameObjects = new List<GameObject>();
        public void addGameObject(GameObject gameObject) => gameObjects.Add(gameObject);
        public void update()
        {
            foreach (GameObject gameObject in gameObjects)
                gameObject.update();
        }
    }
```

PhysicsComponent:

```
class PhysicsComponent
    {
        float velocityX, velocityY;
        float accelerationX, accelerationY;
        float gravity;
        Control objectToAttach;

        public float VelocityX { get => velocityX;}
        public float VelocityY { get => velocityY;}
        public float AccelerationX { get => accelerationX;}
        public float AccelerationY { get => accelerationY;}
        public float Gravity { get => gravity; set => gravity = value; }

        public PhysicsComponent(Control objectToAttach, float gravity = 0)
        {
            this.objectToAttach = objectToAttach;
            this.gravity = gravity;
        }

        public void setSpeed(float velocityX, float velocityY)
        {
            this.velocityX = velocityX;
            this.velocityY = velocityY;
        }
        public void setAcceleration(float accelerationX, float accelerationY)
        {
            this.accelerationX = accelerationX;
            this.accelerationY = accelerationY;
        }
        public void update()
        {
            objectToAttach.Top += (int)velocityY;
            objectToAttach.Left += (int)velocityX;
            velocityX += accelerationX;
            velocityY += accelerationY + gravity;
        }
    }
```

GameObject:

```
abstract class GameObject : PictureBox
    {
        protected PhysicsComponent physics;
        protected Movement movement = new Movement();
        //For adjusting gravity from the properties panel
        public float Gravity { get => physics.Gravity; set => physics.Gravity = value; }
        public GameObject()
        {
            //for creating object from toolbox
            physics = new PhysicsComponent(this);
        }
        public GameObject(Image objectImage, float objectGravity)
        {
            //for creating object programatically
            this.Image = objectImage;
            physics = new PhysicsComponent(this, objectGravity);
        }
        public void setMovement(Movement movement)
        {
            this.movement = movement;
        }
        public virtual void update()
        {
            movement.update(physics);
            physics.update();
            //Refresh();
        }

    }
```

Enemy:

```
class Enemy :GameObject
    {
        public Enemy() : base() { }
        public Enemy(Image enemyImage, float enemyGravity) : base(enemyImage, 
enemyGravity) { }
    }
```

Player:

```
class Player: GameObject
    {
        public Player() : base() { }
        public Player(Image playerImage, float playerGravity) : base(playerImage, 
playerGravity) { }

    }
```

Movement:

```
class Movement
    {
        protected float movementSpeed = 10;
        public void setMovementSpeed(float movementSpeed)
        {
            this.movementSpeed = movementSpeed;
        }
        public virtual void update(PhysicsComponent physics)
        {
            //base behaviour to move rightwards
            physics.setSpeed(movementSpeed, 0);
        }
    }
```

MovementUpDown:

```
class MovementUpAndDown : Movement
    {
        public override void update(PhysicsComponent physics)
        {
            if (physics.VelocityY > 20) physics.Gravity = -1;
            else if (physics.VelocityY < -20) physics.Gravity = 1;

        }
    }
```

MovementAtAngle:

```
class MovementAtAngle :Movement
    {
        bool startFlag = true;
        public override void update(PhysicsComponent physics)
        {
            physics.Gravity = 0;
            if (startFlag)
            {
                physics.setAcceleration(1, 1);
                startFlag = false;
            }
            if (physics.VelocityY > 20) physics.setAcceleration(-1, -1);
            else if (physics.VelocityY < -20) physics.setAcceleration(1, 1);

        }
    }
```