

**Game Development Framework**

**Submitted By:**

**Asad Mehmood , 2020-CS-10**

**Submitted To:**

**Sir. Laeeq Khan Niazi**

**For fulfillment of**

**CS 162 Object Oriented Programming**

**Department of Computer Science**

**University of Engineering and Technology, Lahore**

**Problem Statement:**

* Add falling functionality for different objects (picturebox) such as enemies and other players.

**Solution (In procedural programming):**

If we use procedural programming to solve this problem statement, we have used Gotoxy functionality to control the movement of different objects such as enemies and other players. In this we have used conditional statements in such a way that objects keep on coming down until it not strike the ground.

**Solution (In Object oriented programming):**

**1ST Approach**

In OOP, when we first design that we simply added all the functionalities in MainTickGame() function and also used a enum class .

**Demerits of the solution**

In this approach, we get the expected output but the way was not much dynamic. The objects were not protected. We have to add all the falling conditions for the object to fall under gravity. That was not that much dynamic and was also making code lengthy. From this approach, if we add pictureBox in form then we have to add all the added conditions again for that added pictureBox.

**2nd Approach**

**Code:**

**Form1 Code:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Gravity\_Framework

{

public partial class Gravity : Form

{

Game game = new Game(3);

public Gravity()

{

InitializeComponent();

}

private void timer1\_Tick(object sender, EventArgs e)

{

game.update(ground);

}

private void Gravity\_Load(object sender, EventArgs e)

{

GameObject playerObj = new GameObject(pbPlayer);

GameObject enemyObj1 = new GameObject(enemy1);

GameObject enemyObj2 = new GameObject(enemy2);

GameObject enemyObj3 = new GameObject(enemy3);

game.addGameobjects(playerObj);

game.addGameobjects(enemyObj1);

game.addGameobjects(enemyObj2);

game.addGameobjects(enemyObj3);

}

private void ground\_Click(object sender, EventArgs e)

{

}

}

}

**Game Code:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Collections;

namespace Gravity\_Framework

{

class Game

{

public int gravity;

List<GameObject> addobjects = new List<GameObject>();

public Game(int g)

{

this.gravity = g;

}

public void addGameobjects(GameObject gameobj)

{

addobjects.Add(gameobj);

}

public void update(PictureBox ground)

{

foreach (object get\_object in addobjects)

{

GameObject obj = (GameObject)get\_object;

obj.updateObjects(gravity,ground);

}

}

}

}

**GameObject Code:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Gravity\_Framework

{

class GameObject

{

private PictureBox real\_pictureBox;

public GameObject(PictureBox pB)

{

this.real\_pictureBox = pB;

}

public void updateObjects(int gravity,PictureBox ground)

{

if(this.real\_pictureBox.Bounds.IntersectsWith(ground.Bounds))

{

this.real\_pictureBox.Left += gravity;

}

else

{

this.real\_pictureBox.Top += gravity;

}

}

}

}

**Merits of solution:**

In this approach, we used different classes which are associated with each other and data is Encapsulated. The major benefit of this approach is that we not need to add all the conditions when a new pictureBox (object) is added. Simply, we have to create its object and pass it as parameter to Game class then it will fall under gravity without having additional conditions.

**Demerits of solution:**

The solution is still limited to limited number of objects that are allowed to be created. The only objects are allowed to fall under gravity. The objects are also created manually using the form and the movement of object is not generic.

**UML Diagram:**

Form1

-Gravity : Form

+Gravity()

-Gravity\_Load(object sender , EventArgs e)

-timer1\_Tick(object sender , EventArgs e)

+Game

+GameObject

real\_pictureBox : PictureBox

+gravity : int

-addobjects : List<>

+GameObject(PictureBox pB)

+updateObjects(int gravity , PictureBox ground)

+Game(int g)

+addGameObjects(GameObjects gameobj)

+update(PictureBox ground)