Lab Project

Title:

Human Computer Interaction

Presented By:

Urooj Fatima (2022-BSE-074)

Zainab (2022-BSE-076)

Iqra Shahzad (2022-BSE-078)

Section:

VI-B

Submitted To:

Sir Rehan

Table of Contents

🐸 Frog Game Project

Abstract

A simple 2D game developed using Windows Forms in C#, where the player controls a frog and attempts to navigate across obstacles to reach a goal. The game features levels, a game menu, and interactive controls.

Introduction

The Frog Game is inspired by classic arcade games like "Frogger." The main objective of frog is to get the flying birds while avoiding obstacles like spider or enemy. The game has been built using Windows Forms to demonstrate object movement, collision detection, user input handling, and level-based gameplay.

Objective

The primary goal of the Frog Game is to navigate the frog safely across the screen to reach the other side, avoiding any collisions with obstacles such as spiders or enemies. The player controls the frog using specific key combinations to move forward, backward, or jump. Throughout the game, various hazards are programmed to move or fall dynamically, creating a challenging environment. The player must carefully time their movements and jumps to avoid these obstacles. If the frog collides with an enemy or falls, the game ends. The objective emphasizes quick reflexes, accuracy, and strategic decision-making to successfully reach the destination.

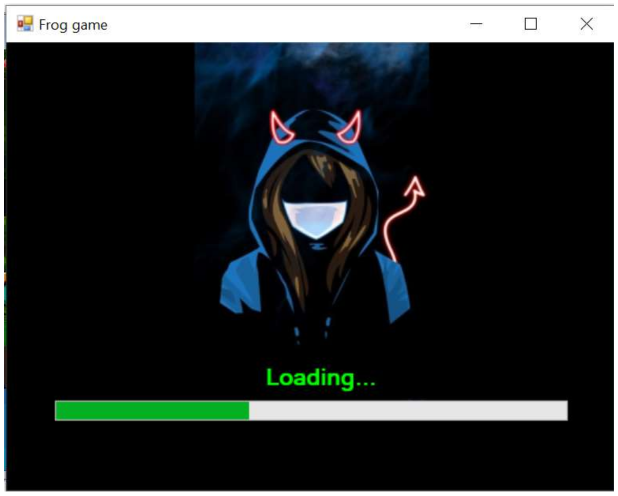
Tools and Technologies Used

- **Programming Language:** C#
- **Framework**: .NET Framework
- **IDE**: Visual Studio
- **Platform**: Windows Forms Application
- **Resources**: Images (sprites), labels, timers for animation

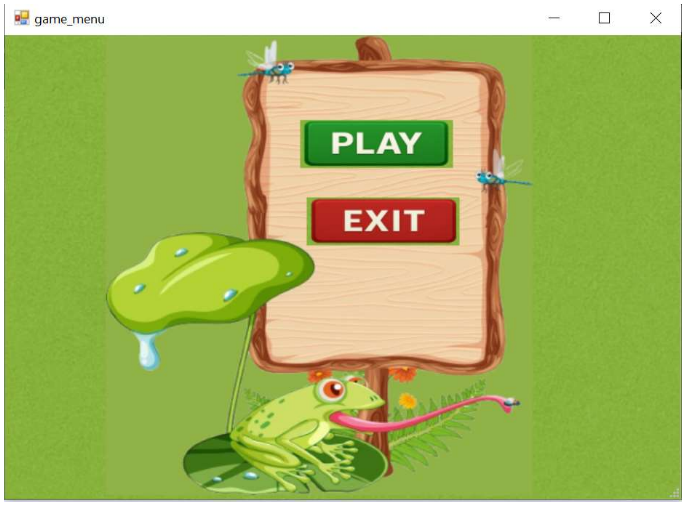
Game Interface Design

Forms:

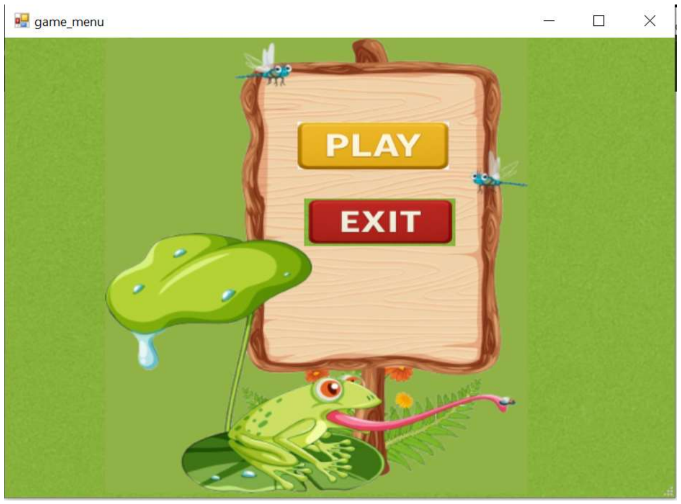
- Form1.cs[Design]



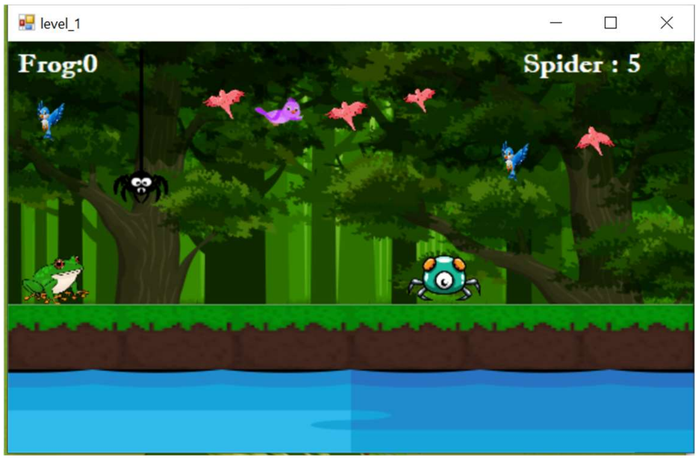
- game_menu.cs[Design]

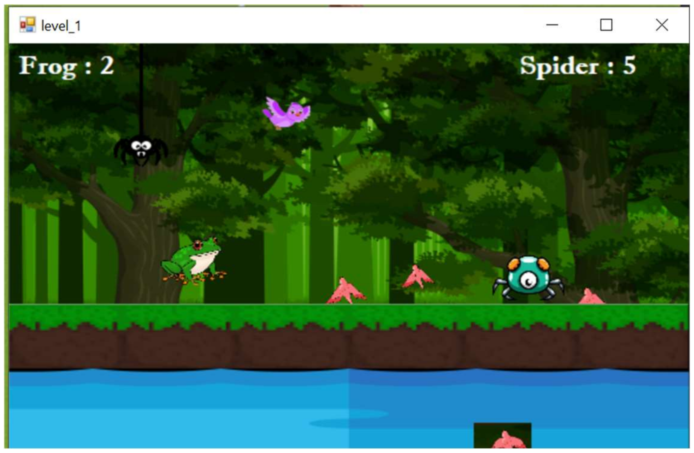


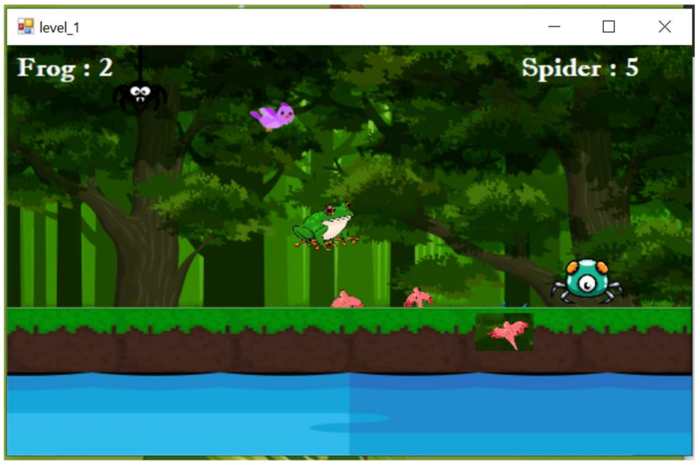
- When User hover on buttons their color changes

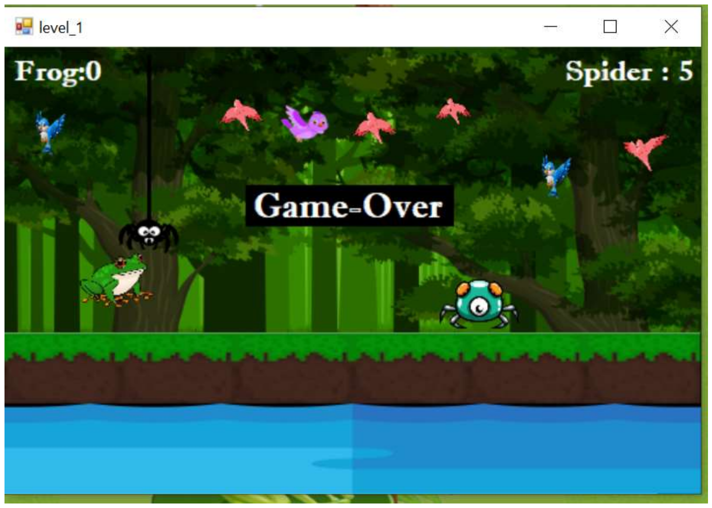


- level1.cs[Design]


level_1
Frog:0
Spider : 5


level_1
Frog : 2
Spider : 5


level_1
Frog : 2
Spider : 5



UI Components:

Labels for score/time, PictureBoxes for obstacles, player, and backgrounds.

Design Style:

Simple and clean interface using default WinForms controls with images.

Game Controls & User Interaction

- **Forward key (Right Arrow)** → Move frog right
- **Backward key (Left Arrow)** → Move frog left
- **Space + Forward (Space + Right Arrow)** → Jump frog forward
- **Shift + Backward (Shift + Left Arrow)** → Move frog backward

➢ **Mouse Clicks** may be used to start or reset the game via buttons on the menu

Code Overview

Form1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
```

```
            this.Opacity = 0;
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (Opacity < 100)
            {
                Opacity += 0.05;
                progressBar1.Increment(2);
            }
            if (progressBar1.Value == 100)
            {
                timer1.Stop();
                game_menu option = new game_menu();
                option.ShowDialog();
            }
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }
    }
}
```

game_menu.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class game_menu : Form
    {
        public game_menu()
        {
            InitializeComponent();
            Application.OpenForms["Form1"].Hide();
        }
```

```
private void pictureBox2_Click(object sender, EventArgs e)
{

}

private void pictureBox3_Click(object sender, EventArgs e)
{

}

private void pictureBox1_Click(object sender, EventArgs e)
{

}

private void btn_play_MouseHover(object sender, EventArgs e)
{
   btn_play.Image = Properties.Resources.btn_play_hover;
}

private void btn_play_MouseLeave(object sender, EventArgs e)
{
   btn_play.Image = Properties.Resources.play_removebg_preview;
}

private void btn_play_Click(object sender, EventArgs e)
{
   level_1 l1 = new level_1();
   l1.ShowDialog();

}

private void btn_exit_MouseHover(object sender, EventArgs e)
{
   btn_exit.Image = Properties.Resources.btn_exit_hover;
}

private void btn_exit_MouseLeave(object sender, EventArgs e)
{
   btn_exit.Image = Properties.Resources.exit_removebg_preview2;
}

private void btn_exit_Click(object sender, EventArgs e)
{
   Application.Exit();
}

private void game_menu_Load(object sender, EventArgs e)
{

}
```

```
    }
}
```

level_1.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class level_1 : Form
    {
        public level_1()
        {
            InitializeComponent();
            lbl_over.Visible = false;
        }
        bool right, left, jump, hold;
        int jump_speed = 1;
        int enemy_speed = 1;
        Random rnd = new Random();
        int spide_speed = 1, spd;

        private void level_1_Load(object sender, EventArgs e)
        {
            timer1.Start();
        }
        int frog_score, spider_score, p;
        void Game_logic()
        {
            //foreach (Control x in this.Controls)
            //{
            //    if (x is PictureBox && string.Equals(x.Tag as string, "fly"))
            //    {
            //        x.Top -= 1;
            //        if (x.Top < 200)
            //        {
            //            x.Top = 200;
            //        }
            //    }
            //}
            foreach (Control x in this.Controls)
            {
                if (x is PictureBox && x.Tag != null && x.Tag.ToString().Contains("fly"))
                {
```

```
                if (x.Tag.ToString().EndsWith("up"))
                {
                   x.Top -= 2;
                   if (x.Top <= 200)
                      x.Tag = "fly_down";
                }
                else if (x.Tag.ToString().EndsWith("down"))
                {
                   x.Top += 2;
                   if (x.Top >= 200)
                      x.Tag = "fly_up";
                }
                if (player.Bounds.IntersectsWith(x.Bounds))
                {
                   frog_score += 1;
                   lbl_frog_score.Text = "Frog : " + frog_score;
                   p = rnd.Next(100, 1200);
                   x.Location = new Point(p, 500);
                }
                if (spider.Bounds.IntersectsWith(x.Bounds))
                {
                   spider_score += 5;
                   lbl_spider_score.Text = "Spider : " + spider_score;
                   p = rnd.Next(100, 1200);
                   x.Location = new Point(p, 500);
                }
                if (player.Bounds.IntersectsWith(enemy.Bounds) ||
player.Bounds.IntersectsWith(spider.Bounds))
                {

                   timer1.Stop();
                   lbl_over.Visible = true;
                   lbl_over.BringToFront();
                }
             }
          }
       }
       void Enemies_movement ()
       {
       if(enemy.Left>500)
       {
       enemy_speed -= 1;
       }
       if(enemy.Left<50)
       {
       enemy_speed= 1;
       }
       enemy. Left += enemy_speed;
          if(spider.Top>10)
          {
          spide_speed -= 2;
```

```
    }
    if (spider.Top < -200)
    {
        spd = rnd.Next(0, 450);
        spider.Location = new Point(spd, -200);
        spide_speed = 2;
    }
    spider. Top+= spide_speed;
}

void player_movement()
{

    if (right == true)
    {
        player.Left += 2;
        player.Image = Properties.Resources.ideal_right_img;
    }
    if (left == true)
    {
        player.Left -= 2;
        player.Image = Properties.Resources.ideal_left_img;
    }
    if (jump == true && right)
    {
        player.Top -= 4;
        player.Left += 3;
        player.Image = Properties.Resources.ideal_right_img;
        jump_speed = 3;
    }
    if (jump == true && left)
    {
        player.Top -= 4;
        player.Left -= 3;
        player.Image = Properties.Resources.ideal_left_img;
        jump_speed = 3;
    }
    if(jump==false)
        {
        player.Top += jump_speed;
        if(player.Bounds. IntersectsWith(ground. Bounds))
        {
            player. Top= ground. Top - player.Height;
            jump_speed= 0;
        }
    }
}
private void pictureBox1_Click(object sender, EventArgs e)
{

}
```

```
private void level_1_KeyDown(object sender, KeyEventArgs e)
{
   if (e.KeyCode==Keys. Right)
   {
   right = true;
   }

   if (e.KeyCode==Keys.Left)
   {
   left = true;
   }
   if (e.KeyCode==Keys.Space&hold)
   {
   jump = true;
   hold = false;
   }
}

private void level_1_KeyPress(object sender, KeyPressEventArgs e)
{

}

private void level_1_KeyUp(object sender, KeyEventArgs e)
{
   if (e.KeyCode == Keys.Right)
   {
      right = false;
   }

   if (e.KeyCode == Keys.Left)
   {
      left = false;
   }
   if (e.KeyCode == Keys.Space & ! hold)
   {
      jump = false;
      hold = true;
   }
}

private void timer1_Tick(object sender, EventArgs e)
{
   player_movement();
   Enemies_movement();
   Game_logic();
}

private void pictureBox2_Click(object sender, EventArgs e)
{
        }
    }
}
```

Challenges Faced & Solutions

- **Handling Smooth Movement**: Using timers and key event handling effectively
- **Collision Detection**: Implemented basic collision logic with .IntersectsWith()
- **Form Navigation**: Switching between menu and game level1 smoothly

Testing and Debugging

- **Manual Testing**: Game was played repeatedly to ensure collision logic, movement, and UI transitions work as expected
- **Error Handling**: Exceptions handled in button events or timer ticks to prevent crashes

Conclusion

This project was a hands-on exercise in game development using Windows Forms. It strengthened understanding of event-driven programming, timers, and graphics handling in C#. Future enhancements may include sound effects, more levels, lives/health system, and improved graphics.

References

- Microsoft Docs (for WinForms)
- Stack Overflow (for logic issues)
- YouTube tutorials on Windows Forms Games