Universidad Rafael Landívar
Facultad de Ingeniería
Ingeniería en Sistemas
Introducción a Programación, sección 04
Catedrática Inga. María del Carmen Castillo

Proyecto No. 2

Andrés Parada Estrada

Carné No. 1152620

Guatemala, 1 de abril del 2020

Introducción

El segundo proyecto consistió en crear un proyecto que simulara un juego llamado “Sapebox, el cual estuviera programado en C# con Windows Forms. El proyecto tenía como principal objetivo el poder poner en práctica el contenido visto en clase, tanto las funciones de C# como los elementos vistos de Windows Forms. También para poner a prueba el razonamiento del alumno para determinar como resolver los problemas o instrucciones que se presenten.

El programa solicita un nombre y un apellido del cual se obtendrá el nombre del piloto. El programa debe de poder leer un archivo de texto donde se va a definir el mapa del nivel, el programa debe ser capaz de leer este archivo y poder verificar si es válido, debiendo cumplir las debidas condiciones. El juego debía de controlarse con las flechas, además se deberá de proveer instrucciones al usuario de como jugar. Si el usuario pierde podrá reiniciarse o directamente salirse.

Análisis

Entradas:

Archivo del nivel, nombre, apellido.

Salidas:

Tablero con el juego y resultados en caso que gane.

Proceso:

```
Form1
{
	string mapa, celda;
	string[,] matrizLetras = new string[20, 20];
	string[] lineasMapa;
	OpenFileDialog openDialog = new OpenFileDialog();
	Form1()
	{
	    InitializeComponent();
	}

	  button1_Click(object sender, EventArgs e)
	{
	archivo:
	    if (openDialog.ShowDialog() == DialogResult.OK)
	    {
	        var text = new StreamReader(openDialog.FileName);
	        if (text != null)
	        {
	            mapa = text.ReadToEnd();
	            lineasMapa = mapa.Split('\n');
	            for (int i = 0; i < 20; i++)
	            {

	                for (int j = 0; j < 20; j++)
	                {
	                    if(lineasMapa[i].IndexOf('\r') != -1)
	                    {
	                        lineasMapa[i] = lineasMapa[i].Remove(lineasMapa[i].IndexOf('\r'), 1);
	                    }

	                    celda = lineasMapa[i].Substring(j, 1);
	                    if(lineasMapa.Length == 20 && lineasMapa[i].Length == 20)
	                    {
	                        matrizLetras[i, j] = celda;
	                    }
	                    else
	                    {
	                        string mensaje = "Error al crear el nivel, numero de filas y columnas es
inválido.";
	                        var result = MessageBox.Show(mensaje, "Error", MessageBoxButtons.OK);
	                        goto archivo;
	                    }
	                }
	            }
```

```
                var th = new Thread(() => Application.Run(new Form2(matrizLetras)));
                th.SetApartmentState(ApartmentState.STA);
                th.Start();
                this.Close();
            }
            else
            {
                string mensaje = "El arhivo esta vacío, por favor eliga otro.";
                var result = MessageBox.Show(mensaje, "Error", MessageBoxButtons.OK);
                goto archivo;
            }

        }
    }

      Form1_Load(object sender, EventArgs e)
    {

    }

      button2_Click(object sender, EventArgs e)
    {
        var th = new Thread(() => Application.Run(new Instrucciones()));
        th.SetApartmentState(ApartmentState.STA);
        th.Start();
    }
}

Form2
{
string piloto, nombre, apellido;
    string[,] map;
     Form2(string[,] mapa)
    {
        InitializeComponent();
        piloto = "GUA-";
        map = mapa;
    }

      btnAceptar_Click(object sender, EventArgs e)
    {
        if (txtNombre.Text != null && txtNombre.Text != "" && txtApellido.Text != null &&
txtApellido.Text != "")
        {
            nombre = txtNombre.Text;
            apellido = txtApellido.Text;
            piloto = piloto + apellido.Substring(apellido.Length - 1, 1).ToUpper() + "-" +
(nombre.Length + apellido.Length);
            var th = new Thread(() => Application.Run(new Tablero(map, piloto)));
            th.SetApartmentState(ApartmentState.STA);
            th.Start();
            this.Close();

        }
        else
        {
```

```
            MessageBox.Show("Ingrese un nombre y un apellido");
        }
    }
}
```

```
Tablero
{
Operaciones o;
        Image[,] tab;
         Tablero(string[,] mapa, string nombre)
        {
            InitializeComponent();

            o = new Operaciones();
            tab = o.generarMapa(mapa, nombre);
            if(tab != null)
            {
                for (int i = 0; i < 20; i++)
                {
                    dgMapa.Rows.Add();
                    for (int j = 0; j < 20; j++)
                    {
                        dgMapa.Rows[i].Cells[j].Value = tab[i, j];
                    }
                }
            }
            else
            {
                Application.Restart();
            }
            lblCasillas.Text = o.casillas.ToString();
            lblMov.Text = o.movimientos.ToString();
            lblPuntos.Text = o.puntos.ToString();
            lblNombre.Text = nombre;
        }

          actualizarTablero(Image[,] map)
        {
            if(map != null)
            {
                for (int i = 0; i < 20; i++)
                {
                    for (int j = 0; j < 20; j++)
                    {
                        if(dgMapa.Rows[i].Cells[j].Value != map[i, j])
                        {
                            dgMapa.Rows[i].Cells[j].Value = map[i, j];
                        }
                    }
                }
            }
            else
            {
                if(o.gano)
                {
                    DialogResult th = MessageBox.Show("¡Has llegado a la tierra! \n" + "Puntos: " +
o.puntos + "\n Movimientos: " + o.movimientos + "\n Casillas" + o.casillas + "\n¿Desea iniciar otro
nuevo nivel?", "Mision Cumplida", MessageBoxButtons.YesNo);
                    if (th == DialogResult.Yes)
                    {
                        Application.Restart();
```

```
                }
                else if(th == DialogResult.No)
                {
                    Application.Exit();
                }
            }
            else if(o.perdio)
            {
                DialogResult th = MessageBox.Show("¡Te perdiste en el espacio!", "Mision Fallida",
MessageBoxButtons.RetryCancel);
                if (th == DialogResult.Retry)
                {
                    reiniciar();
                }
                else if(th == DialogResult.Cancel)
                {
                    Application.Restart();
                }
            }
        }
    }

      dgMapa_KeyDown(object sender, KeyEventArgs e)
    {
        switch(e.KeyCode)
        {
            case Keys.Down:
                actualizarTablero(o.bajar());
                lblCasillas.Text = o.casillas.ToString();
                lblMov.Text = o.movimientos.ToString();
                lblPuntos.Text = o.puntos.ToString();
                break;

            case Keys.Up:
                actualizarTablero(o.subir());
                lblCasillas.Text = o.casillas.ToString();
                lblMov.Text = o.movimientos.ToString();
                lblPuntos.Text = o.puntos.ToString();
                break;

            case Keys.Left:
                actualizarTablero(o.izquierda());
                lblCasillas.Text = o.casillas.ToString();
                lblMov.Text = o.movimientos.ToString();
                lblPuntos.Text = o.puntos.ToString();
                break;

            case Keys.Right:
                actualizarTablero(o.derecha());
                lblCasillas.Text = o.casillas.ToString();
                lblMov.Text = o.movimientos.ToString();
                lblPuntos.Text = o.puntos.ToString();
                break;
        }
    }
```

```
          reiniciar()
        {
            actualizarTablero(o.reiniciar());
            o.casillas = 0;
            o.movimientos = 0;
            o.puntos = 0;
            lblMov.Text = "0";
            lblCasillas.Text = "0";
            lblPuntos.Text = "0";
        }

          btnReiniciar_Click(object sender, EventArgs e)
        {
            reiniciar();
        }
}

Instrucciones
{
 Instrucciones()
        {
            InitializeComponent();
        }
  button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
}

Operaciones
{
string[,] matrizLetras = new string[20, 20];
        string[,] inicialLetras = new string[20, 20];

        Image[,] matrizMapa = new Image[20, 20];

        Image[,] inicialMapa = new Image[20, 20];

        static string path = @".\Images\";
        string name;
        Bitmap imagenNaveAr = new Bitmap(path + "nave.jpg");
        Bitmap imagenNaveAb = new Bitmap(path + "nave_abajo.jpg");
        Bitmap imagenNaveD = new Bitmap(path + "nave_derecha.jpg");
        Bitmap imagenNaveI = new Bitmap(path + "nave_izquierda.jpg");
        Bitmap imagenVacio = new Bitmap(path + "vacio.jpg");
        Bitmap imagenSombreado = new Bitmap(path + "vaciosombreado.jpg");
        Bitmap imagenGemaRoja = new Bitmap(path + "gema_roja.jpg");
        Bitmap imagenGemaAzul = new Bitmap(path + "gema_azul.jpg");
        Bitmap imagenGemaAmarilla = new Bitmap(path + "gema_amarilla.jpg");
        Bitmap imagenAsteroide = new Bitmap(path + "asteroide.jpg");
        Bitmap imagenTierra = new Bitmap(path + "tierra.jpg");
        int cantTierra = 0;
        int cantGemaR = 0;
        int cantGemaA = 0;
        int cantGemaAma = 0;
        int cantNave = 0;
```

```
int cantAsteoride = 0;
int posNaveX = 0;
int posNaveY = 0;
int inicioX = 0;
int inicioY = 0;

 int puntos { set; get; }
 int movimientos { set; get; }
 int casillas { set; get; }
string celda;

 bool gano { set; get; }
 bool perdio { set; get; }
bool parar = false;


 Image[,] generarMapa(string[,] mL, string nombre)
{
    name = nombre;
    bool valido = true;
    gano = false;
    perdio = false;
    puntos = 0;
    movimientos = 0;
    casillas = 0;
    matrizLetras = mL;
    for (int i = 0; i < 20; i++ )
    {
        for (int j = 0; j < 20; j++ )
        {
            switch (mL[i,j])
            {
                case "A":
                    matrizMapa[i, j] = imagenVacio;
                    break;
                case "B":
                    matrizMapa[i, j] = imagenNaveAr;
                    cantNave++;

                    posNaveY = i;
                    inicioY = i;
                    posNaveX = j;
                    inicioX = j;
                    break;
                case "C":
                    matrizMapa[i, j] = imagenAsteroide;
                    cantAsteoride++;
                    break;
                case "D":
                    matrizMapa[i, j] = imagenTierra;
                    cantTierra++;
                    break;
                case "E":
                    matrizMapa[i, j] = imagenGemaAzul;
                    cantGemaA++;
                    break;
```

```
                case "F":
                    matrizMapa[i, j] = imagenGemaRoja;
                    cantGemaR++;
                    break;
                case "G":
                    matrizMapa[i, j] = imagenGemaAmarilla;
                    cantGemaAma++;
                    break;
                default:
                    valido = false;
                    break;
            }
        }
    }


    if(cantAsteoride > 0 && cantNave == 1 && cantTierra == 1 && cantGemaA <= 5 && valido)
    {
        for (int i = 0; i < 20; i++ )
        {
            for(int j = 0; j < 20; j++)
            {
                inicialMapa[i, j] = matrizMapa[i, j];
            }
        }
        return matrizMapa;
    }
    else
    {
        string mensaje = "Error al crear el nivel, el archivo es inválido.";
        var result = MessageBox.Show(mensaje, "Error", MessageBoxButtons.OK);
        return null;
    }
}

 Image[,] reiniciar()
{
    posNaveX = inicioX;
    posNaveY = inicioY;
    matrizLetras = inicialLetras;
    matrizMapa = inicialMapa;
    cantTierra = 0;
    cantGemaR = 0;
    cantGemaA = 0;
    cantGemaAma = 0;
    cantNave = 0;
    cantAsteoride = 0;
    gano = false;
    perdio = false;
    parar = false;
    return matrizMapa;
}

 Image[,] bajar()
{
     parar = false;
```

```
movimientos++;

while(posNaveY + 1 <= 20 && !gano && !perdio && !parar)
{

   if((posNaveY) < 19)
   {
      celda = matrizLetras[posNaveY + 1, posNaveX];
      switch(celda)
      {
         case "A":

            matrizMapa[posNaveY, posNaveX] = imagenSombreado;
            matrizLetras[posNaveY, posNaveX] = "A";

            posNaveY++;


            matrizMapa[posNaveY, posNaveX] = imagenNaveAb;
            matrizLetras[posNaveY, posNaveX] = "B";
            casillas++;
            break;
         case "C":
            parar = true;
            break;
         case "D":

            matrizMapa[posNaveY, posNaveX] = imagenSombreado;
            matrizLetras[posNaveY, posNaveX] = "A";

            posNaveY++;

            matrizMapa[posNaveY, posNaveX] = imagenNaveAb;
            matrizLetras[posNaveY, posNaveX] = "B";
            casillas++;
            gano = true;
            break;
         case "E":

            matrizMapa[posNaveY, posNaveX] = imagenSombreado;
            matrizLetras[posNaveY, posNaveX] = "A";
            posNaveY++;
            matrizMapa[posNaveY, posNaveX] = imagenNaveAb;
            matrizLetras[posNaveY, posNaveX] = "B";
            casillas++;
            puntos = puntos + 100;
            break;
         case "F":
            matrizMapa[posNaveY, posNaveX] = imagenSombreado;
            matrizLetras[posNaveY, posNaveX] = "A";
            posNaveY++;
            matrizMapa[posNaveY, posNaveX] = imagenNaveAb;
            matrizLetras[posNaveY, posNaveX] = "B";
            casillas++;
            puntos = puntos + 200;
            break;
```

```
                case "G":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveY++;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveAb;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 50;
                    break;
                }
            }
            else
            {
                perdio = true;
            }
        }
        if (perdio || gano)
        {

            return null;
        }
        else
        {
            return matrizMapa;
        }

    }

     Image[,] subir()
    {
        parar = false;
        movimientos++;
        while (posNaveY-1 >= -1 && !gano && !perdio && !parar)
        {
            if ((posNaveY) > 0)
            {
                celda = matrizLetras[posNaveY - 1, posNaveX];
                switch (celda)
                {
                    case "A":

                        if(posNaveY == 0)
                        {
                            perdio = true;
                            break;
                        }
                        else
                        {
                            matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                            matrizLetras[posNaveY, posNaveX] = "A";
                            posNaveY--;
                            matrizMapa[posNaveY, posNaveX] = imagenNaveAr;
                            matrizLetras[posNaveY, posNaveX] = "B";
                            casillas++;
                            break;
                        }
```

```
                case "C":
                    parar = true;
                    break;
                case "D":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveY--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveAr;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    gano = true;
                    break;
                case "E":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveY--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveAr;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 100;
                    break;
                case "F":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveY--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveAr;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 200;
                    break;
                case "G":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveY--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveAr;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 50;
                    break;
            }
        }
        else
        {
            perdio = true;
        }
    }
    if (perdio || gano)
    {

        return null;
    }
    else
    {
        return matrizMapa;
    }
}
```

```
 Image[,] derecha()
{
    parar = false;
    movimientos++;
    while (posNaveX + 1 <= 20 && !gano && !perdio && !parar)
    {
        if ((posNaveX) < 19)
        {
            celda = matrizLetras[posNaveY, posNaveX + 1];
            switch (celda)
            {
                case "A":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX++;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveD;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    break;
                case "C":
                    parar = true;
                    break;
                case "D":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX++;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveD;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    gano = true;
                    break;
                case "E":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX++;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveD;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 100;
                    break;
                case "F":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX++;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveD;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 200;
                    break;
                case "G":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX++;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveD;
                    matrizLetras[posNaveY, posNaveX] = "B";
```

```
                casillas++;
                puntos = puntos + 50;
                break;
            }
        }
        else
        {
            perdio = true;
        }
    }
    if (perdio || gano)
    {
        return null;
    }
    else
    {
        return matrizMapa;
    }
}

 Image[,] izquierda()
{
    parar = false;
    movimientos++;
    while (posNaveX - 1 >= -1 && !gano && !perdio && !parar)
    {
        if ((posNaveX) > 0)
        {
            celda = matrizLetras[posNaveY, posNaveX - 1];
            switch (celda)
            {
                case "A":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveI;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    break;
                case "C":
                    parar = true;
                    break;
                case "D":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveI;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    gano = true;
                    break;
                case "E":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX--;
```

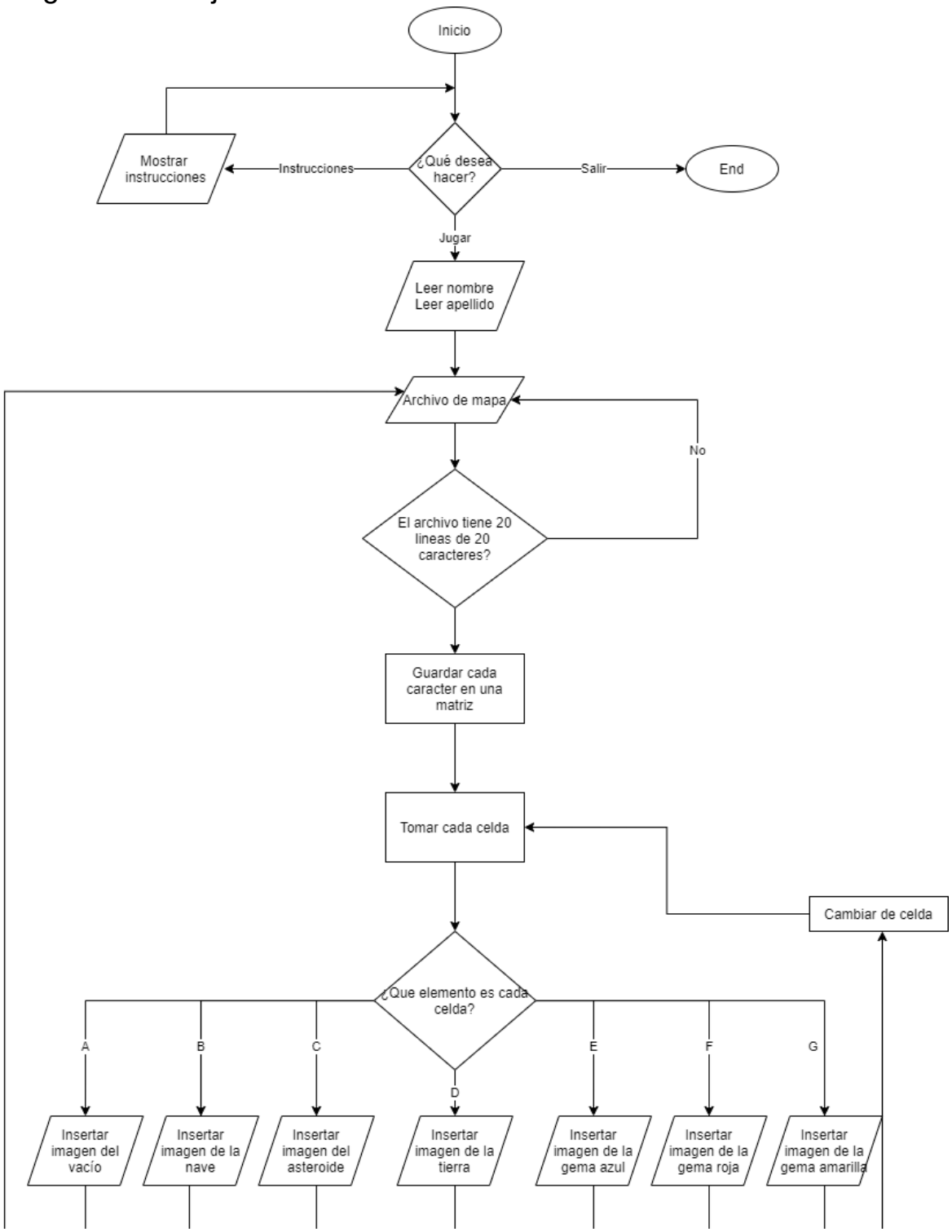
```
                    matrizMapa[posNaveY, posNaveX] = imagenNaveI;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 100;
                    break;
                case "F":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveI;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 200;
                    break;
                case "G":
                    matrizMapa[posNaveY, posNaveX] = imagenSombreado;
                    matrizLetras[posNaveY, posNaveX] = "A";
                    posNaveX--;
                    matrizMapa[posNaveY, posNaveX] = imagenNaveI;
                    matrizLetras[posNaveY, posNaveX] = "B";
                    casillas++;
                    puntos = puntos + 50;
                    break;
            }
        }
        else
        {
            perdio = true;

        }
    }
    if (perdio || gano)
    {

        return null;
    }
    else
    {
        return matrizMapa;
    }
  }
}
```

Diseño

Diagrama de flujo



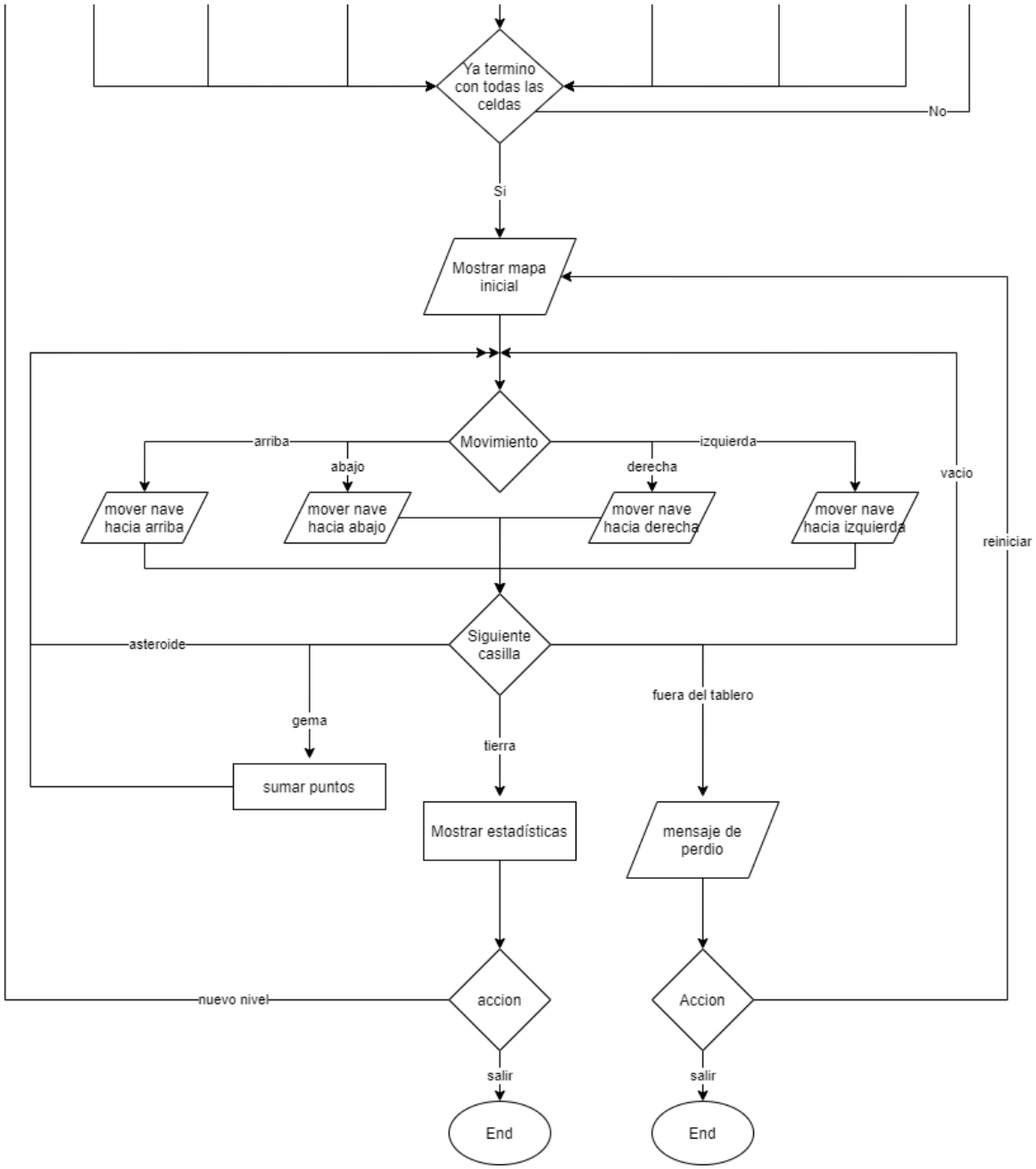

Ya termino con todas las celdas
No
Si
Mostrar mapa inicial
Movimiento
arriba
abajo
derecha
izquierda
mover nave hacia arriba
mover nave hacia abajo
mover nave hacia derecha
mover nave hacia izquierda
vacio
reiniciar
Siguiente casilla
asteroide
gema
tierra
fuera del tablero
sumar puntos
Mostrar estadísticas
mensaje de perdio
nuevo nivel
accion
Accion
salir
salir
End
End

Diagrama de clases

Form1	
Atributos	string mapa, celda string[,] matrizLetras = new string[20, 20] string[] lineasMapa OpenFileDialog openDialog = new OpenFileDialog()
Métodos	button1_Click()

Constructor	Form1()
	{
	InitializeComponent();
	}

Form2	
Atributos	string piloto, nombre, apellido; string[,] map;
Métodos	btnAceptar_Click()

Constructor	Form2(string[,] mapa)
	{
	InitializeComponent();
	piloto = "GUA-";
	map = mapa;
	}

Instrucciones	
Atributos	N/A
Métodos	button1_Click()

Constructor	Instrucciones()
	{
	InitializeComponent();
	}

Operaciones	

Atributos	string[,] matrizLetras = new string[20, 20];
	string[,] inicialLetras = new string[20, 20];
	
	Image[,] matrizMapa = new Image[20, 20];
	
	Image[,] inicialMapa = new Image[20, 20];
	
	//Declaración de strings
	static string path = @".\Images\";
	string name;
	
	//Declaración de las imagenes
	Bitmap imagenNaveAr = new Bitmap(path + "nave.jpg");
	Bitmap imagenNaveAb = new Bitmap(path + "nave_abajo.jpg");
	Bitmap imagenNaveD = new Bitmap(path + "nave_derecha.jpg");
	Bitmap imagenNaveI = new Bitmap(path + "nave_izquierda.jpg");
	Bitmap imagenVacio = new Bitmap(path + "vacio.jpg");
	Bitmap imagenSombreado = new Bitmap(path + "vaciosombreado.jpg");
	Bitmap imagenGemaRoja = new Bitmap(path + "gema_roja.jpg");
	Bitmap imagenGemaAzul = new Bitmap(path + "gema_azul.jpg");
	Bitmap imagenGemaAmarilla = new Bitmap(path + "gema_amarilla.jpg");
	Bitmap imagenAsteroide = new Bitmap(path + "asteroide.jpg");
	Bitmap imagenTierra = new Bitmap(path + "tierra.jpg");
	
	//Declaración de enteros
	int cantTierra = 0;
	int cantGemaR = 0;
	int cantGemaA = 0;
	int cantGemaAma = 0;
	int cantNave = 0;
	int cantAsteoride = 0;
	int posNaveX = 0;
	int posNaveY = 0;
	int inicioX = 0;
	int inicioY = 0;
	
	int puntos { set; get; }
	int movimientos { set; get; }
	int casillas { set; get; }
	
	//Declaración de strings
	string celda;
	
	//Declaración de booleanos
	bool gano { set; get; }
	bool perdio { set; get; }
	bool parar = false;

Métodos	Image[,] generarMapa(string[,] mL, string nombre) Image[,] reiniciar() Image[,] bajar() Image[,] subir() Image[,] derecha() Image[,] izquierda()
Constructor	N/A

Tablero	
Atributos	Operaciones o; Image[,] tab;
Métodos	actualizarTablero(Image[,] map) dgMapa_KeyDown() btnReiniciar_Click()

Constructor	Tablero(string[,] mapa, string nombre)
	{
	InitializeComponent();
	
	//Inicialización del mapa y creación del datagrid
	o = new Operaciones();
	tab = o.generarMapa(mapa, nombre);
	
	//Valida que el mapa sea valido, si lo es lo mandara al datagrid
	if(tab != null)
	{
	for (int i = 0; i < 20; i++)
	{
	dgMapa.Rows.Add();
	for (int j = 0; j < 20; j++)
	{
	dgMapa.Rows[i].Cells[j].Value = tab[i, j];
	}
	}
	}
	else
	{
	Application.Restart();
	}
	//Inicializa los label
	lblCasillas.Text = o.casillas.ToString();
	lblMov.Text = o.movimientos.ToString();
	lblPuntos.Text = o.puntos.ToString();
	lblNombre.Text = nombre;
	}

Conclusiones

1. Las matrices y vectores dan lugar a muchas funcionalidades. Con estos elementos se pueden formular una gran cantidad de opciones e ideas para un programa.

2. Windows Forms es un buen diseñador de pantallas muy fácil de usar y que tienen una gran variedad de elementos para hacer del programa más estético.

3. La mejor forma de realizar proyectos que requieren de muchos proyectos es estructurar el código, ordenándolos y colocando comentarios para poder clasificar muy bien el código.

Recomendaciones

1. Reforzar algunos temas que sean esenciales para el proyecto, dando ejemplos que se asemejen a la forma de resolver ciertas trozos del proyecto.
2. Evitar dejar temas sin dar que sean esenciales para e proyecto, o que forma parte de como hacer el proyecto. Por ejemplo las imágenes que nunca se enseñaron como cargarlos por medio de código para que sea dinámico.

Bibliografía

- Microsoft (s.f.) Fuente de código de StreamReader().https://docs.microsoft.com/en-us/dotnet/api/system.io.streamreader?view=netcore-3.1
- Microsoft (s.f.) Fuentes de código de Thread. https://docs.microsoft.com/en-us/dotnet/api/system.threading.thread?view=netcore-3.1
- Microsoft (s.f.) Fuente de código de MessageBox. https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.messagebox?view=netcore-3.1

Anexos

Manual de usuario

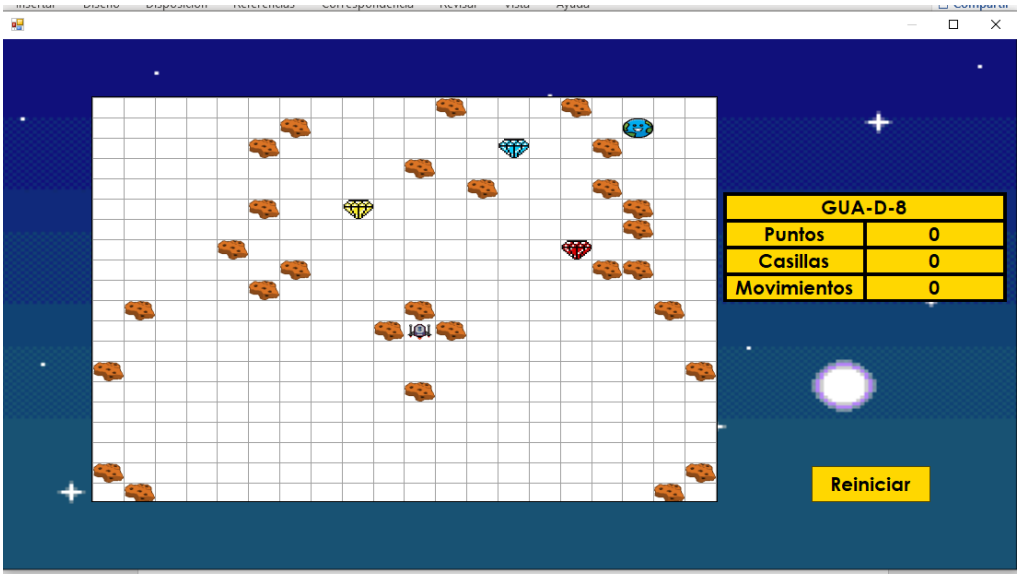
El botón Jugar abre un cuadro de dialogo para poder elegir el archivo de texto con la información de nivel. Si este es inválido muestra un message box y le vuelve a solicitar el archivo.

El botón instrucciones abre la ventana con las instrucciones para jugar.

Muestra las instrucciones.

El botón de ok regresa al menú principal

Solicita un nombre y un apellido para crear el nombre del piloto. No acepta campos vacíos.



Esta ventana muestra el tablero del juego. El jugador se mueve con las flechas, se le muestran las estadísticas de los puntos, las casillas y los movimientos que lleva. El botón reiniciar, vuelve a comenzar el nivel. Gano el usuario gana o pierde se le muestra un message box. Si ganó, se le mostrarán las estadísticas y la opción de regresar al menú principal, si perdió se le dará la oportunidad de reiniciar el mapa o salir de la simulación.