

Snake alkalmazás

Konzulens tanár:
Boros Sándor

Készítette:
Fényi Tamás
5/13.1

1 Bevezetés

Az alkalmazásom a régi „Snake” játékot fogja úgymond reprodukálni csak kissé másként. A Snake egy olyan játék, ahol egy Snake-t kell irányítani és el kell kerülni az úgynevezett „ütközőket” más néven akadályokat, miközben minél több objektumot szedsz fel a több pontszámért és nehezebb szintekért, mindeközben a Snake folyamatosan nő az objektumoktól. Oda kell figyelni, hogy ne menj neki a pálya szélének, az akadályoknak vagy magának a Snake-nek, mert akkor játék vége. Mikor a játéknak vége van a játékos kap egy lehetőséget, hogy lementse a végeredményét, amit majd a játék egy úgynevezett „Leaderboard”-on kifog majd mutatni. Miután lementette lehetősége van újra játszani vagy kilépni a játékból.

Azért választottam, mert iskolai tudásomat tesztelni szerettem volna és mindig is akartam egy alkalmazást készíteni. Más hasonló alkalmazások megnézése és játszása után. Kódjaik alapba vételével készítettem el saját programomat.

A Snake játék fejleszti a játékosok reflexét és teszteli a tudásukat. A pontszámtól függő pálya szint változásával és annak nehezedésével és a Snake gyorsaságának és annak hosszának növekedésével, teszteli a játék a játékosokat. A játékot bármely korosztály tudja úgymond játszani, gyerek, felnőtt, bárki. Bármely eszközön lehet a Snake-vel játszani, telefon, gép, a legtöbb eszközön elérhető és játszható. Keveset foglal és kevésbé igényel nagy grafikát vagy hasonlót.

2 Fejlesztői dokumentáció

A Snake alkalmazással összemérhetik a játékosok a tudásukat és fejlesztheti a reflexüket. A program összesíti többféle más hasonló program funkcióját. A Snake attól függően gyorsabb lesz és az alapján, hogy mennyi objektumot vett fel annál hosszabb lesz. Mint ez alatt, szint alapján a pálya egyre nehezebb lesz, mivel több akadály úgynevezett „ütköző” fog megjelenni. Az objektumok randomizálva fognak megjelenni a pályán, egy-két perces intervallummal egy előző objektum lejövetele után. A Snake nem mehet neki a pálya szélének vagy az úgynevezett „ütközőknek” vagy pedig saját magának, mert különben a játék véget ér. A végső pontszám és szint ki lesz majd mutatva egy ablakon és majd a játékosnak lehetősége lesz lementeni azt és majd hozzá kerül a neve is. Majd a játékosnak ahhoz is lehetősége lesz, hogy egy úgynevezett „Leaderboard”-on megtekintse mások eredményeit a „Top 10”-ben. A kezdőképernyőn lesznek majd a lehetőségek, amikre kattintva odamegy az adott Form-hoz.

Azért választottam a c#(C Sharp) programozási nyelvet, a nyelvnek amiben írom, mivel azt tudom leginkább és így jobban erősítem abból a tudásomat, hogy egy asztali alkalmazást írok benne.

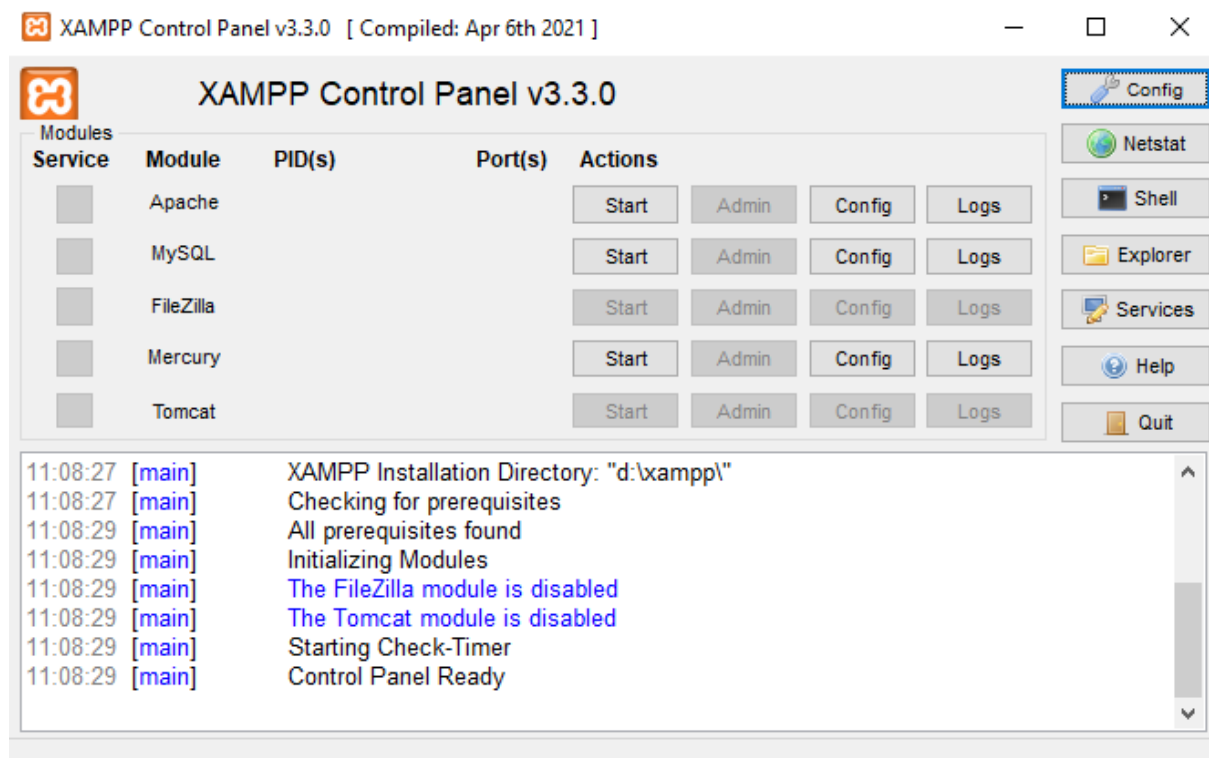
2.1 Használt hardver

A gépnek minimum 4.7.2 verziójú .net framework-ot kell futtatnia.

2.2 Használt szoftver

A számítógép a Windows operációs rendszert használja.

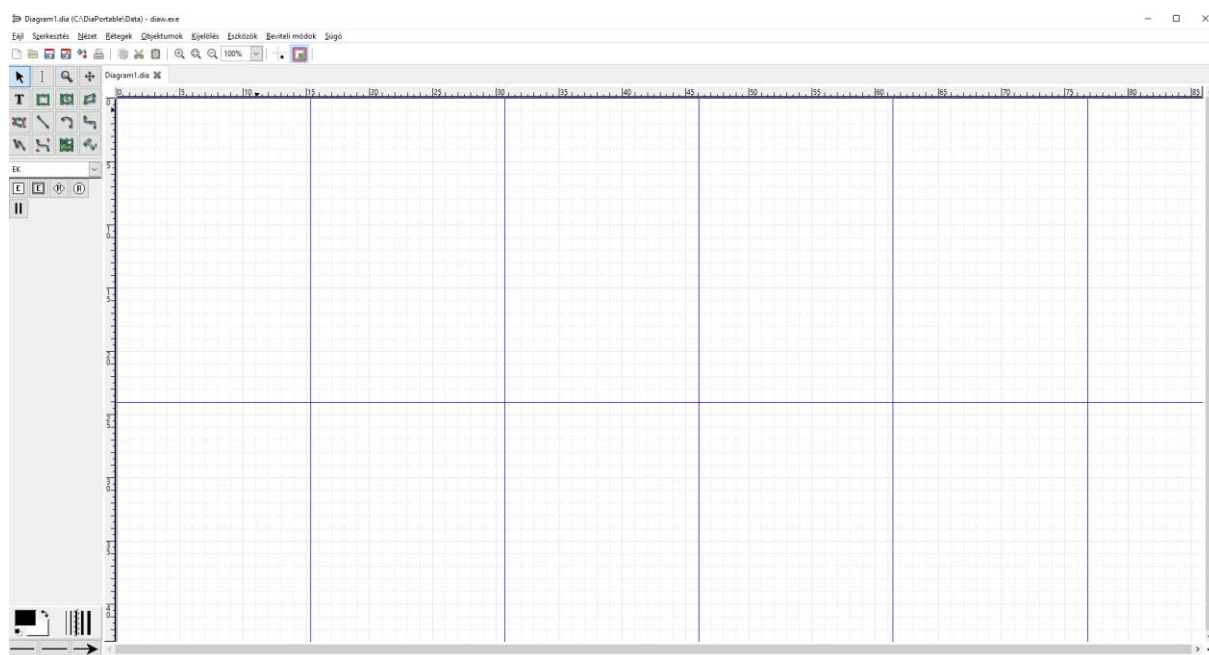
Xampp



ábra 1 Xampp szoftver

A Xampp szoftvert használtam adatbázisomhoz való kapcsolódáshoz és adatbázisom elkészítéséhez. Ezen belül a 'MySQL'-nél az „Admin”-ra kattintva elvisz minket a PhpMyAdmin-ba és ott lehet adatbázist kezelni és ahhoz relációs ábrát készíteni.

DiaPortable



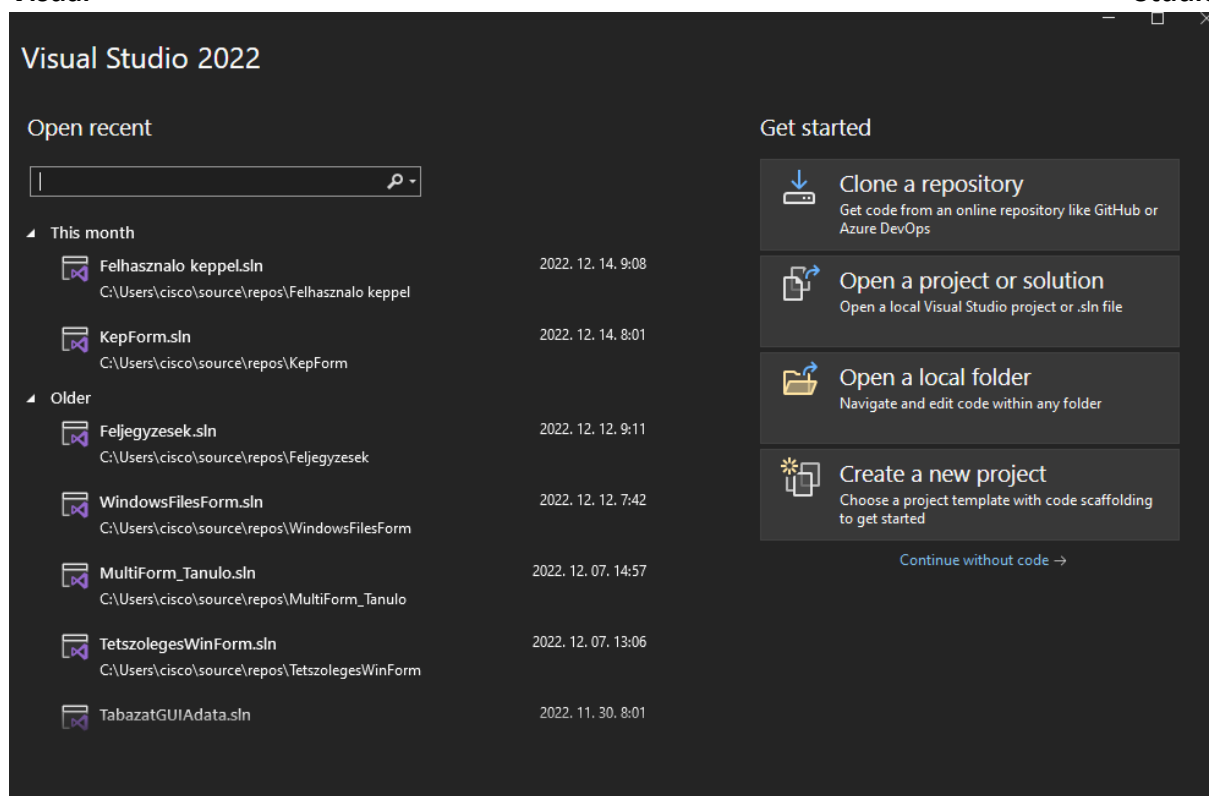
ábra 2 DiaPortable szoftver

A DiaPortable nevezetű szoftver segítségével készítettem el az adatbázisomhoz kellő ábrát.

Különbféle ábrák készítésére megfelelő szoftver és könnyen használható. Elég könnyen meg lehet tanulni a használatát.

Visual

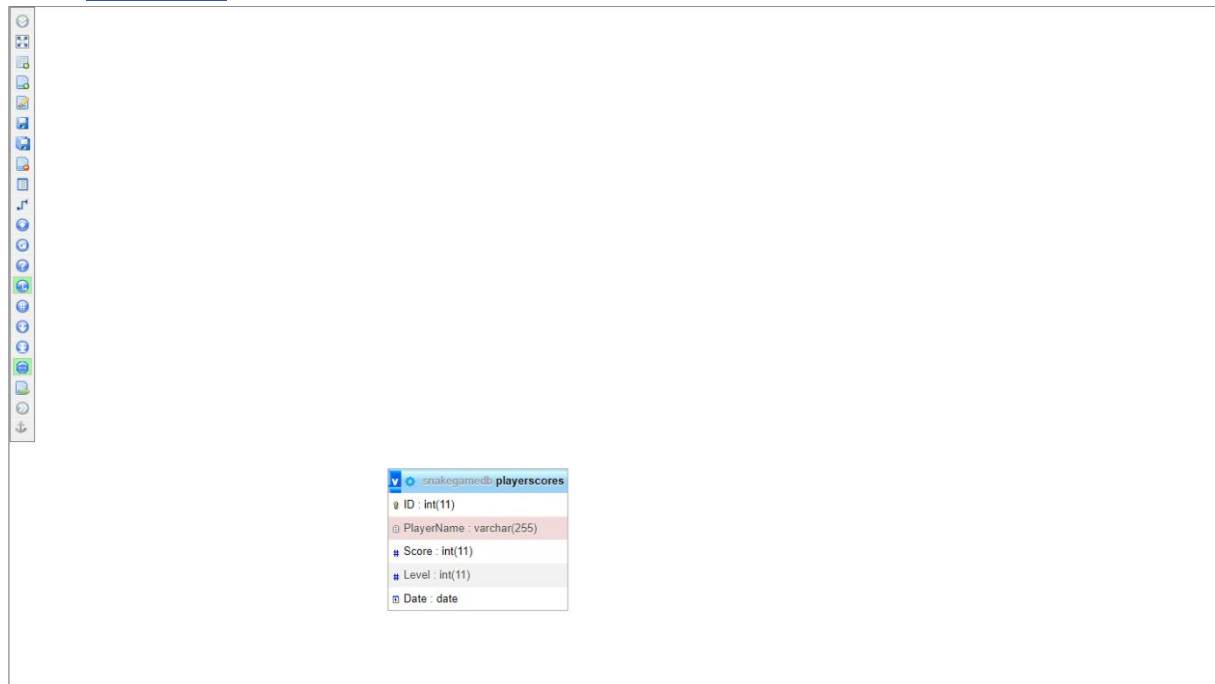
Studio



ábra 3 Visual Studio szoftver

A Visual Studio szoftver segítségével írtam meg az alkalmazásomat, ez az egyik legjobb szoftver az asztali alkalmazások készítésére, tökéletes volt számomra.

2.3 Adatbázis

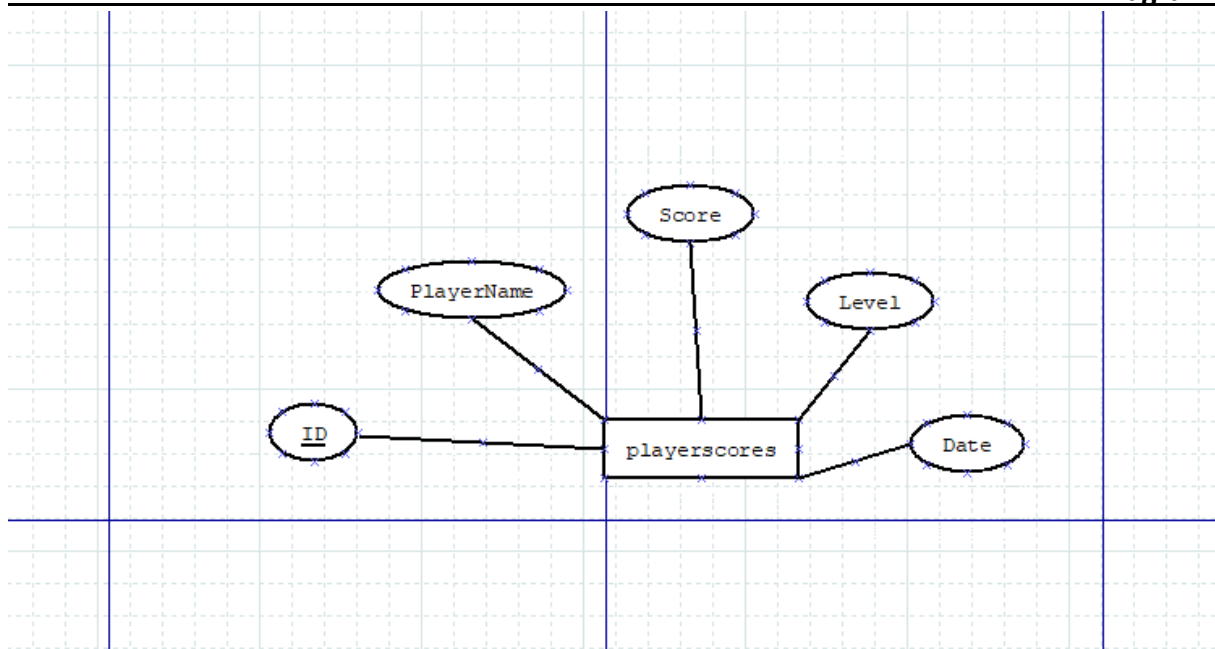


snakegamedb: playerscores	
ID	int(11)
PlayerName	varchar(255)
Score	int(11)
Level	int(11)
Date	date

ábra 4 Adatbázis ábra

ER

Diagram:



ábra 5 ER diagram

Az alábbi ábrák bemutatják az adatbázist, amelyet az alkalmazás használ az adatok tárolására és funkcióinak használatához.

2.3.1 Funkciói

Az adatbázis nevezetesen „snakegamedb”, a játékos végső eredményét fogja letárolni az alapján hogy úgy döntött lementi-e vagy sem, egy „playerscores” táblába és ott letárolja későbbi használatra. A „playerscores” táblának van 5 sora, nevezetesen „ID”, „PlayerName”,

„Score”, „Level”, „Date”. Az „ID” egy speciális azonosító amit arra fogunk használni, hogy beazonosítsuk melyik játékosé az a „PlayerName”, „Score” és „Level”. A „Date” lefogja tárolni mikor lett lementve az adat, így tudni fogjuk napra pontosan a dátumot, ezzel megkönnyebbítve az adat megtalálását. A „PlayerName” fogja tárolni a játékos nevét, amit majd megad a felhasználó/játékos. A „Score” fogja tárolni a vég pontszámot, amit elért a játékos a játék végén, ez majd az úgynevezett „GameOverScreen”-en látható is lesz, amikor vége lesz a játéknak. A „Level” fogja tárolni amelyik szintig eljutott a játékos és ezt is látni fogja majd a „GameOverScreen”-en, amikor a játéknak vége lesz.

Ezt az adatbázis még arra is fogjuk használni, hogy a „LeaderboardScreen”-en, kimutassa a „Top 10” játékost a hozzá tartozó pontszámmal és szinttel.

Kód részletek

Az alábbi kód részlet maga a Database 'class' mármint osztályból lesz, amit az SQL adatbázis csatlakozásához és kezeléséhez használunk.

Amint látni lehet deklarálunk három privát változót, amit használni fogunk az Database class/osztályban.

```
private MySqlConnection sqlconnection;  
private MySqlCommand sqlcommand;  
private string errorMessage = null;
```

Az alábbi metódussal csatlakozunk az SQL adatbázishoz.

```
public Database(string host, string user, string password, string database)  
{  
  
    MySqlConnectionStringBuilder stringbuilder = new MySqlConnectionStringBuilder();  
    stringbuilder.Server = host;  
    stringbuilder.UserID = user;  
    stringbuilder.Password = password;  
    stringbuilder.Database = database;  
    sqlconnection = new MySqlConnection(stringbuilder.ConnectionString);  
    sqlcommand = sqlconnection.CreateCommand();  
  
}
```

//Ez után írunk 2 metódust hogy nyissa és zárja az adatbázishoz való csatlakozást és ha hiba van akkor jelezze.

```
private bool databaseOpen()  
{  
  
    try  
    {  
  
        if (sqlconnection.State != System.Data.ConnectionState.Open)  
        {  
  
            sqlconnection.Open();  

```

```

    }

}
catch (MySqlException ex)
{

    errorMessage = ex.Message;
    return false;

}

return true;
}
private bool databaseClose()
{

    try
    {

        if (sqlconnection.State != System.Data.ConnectionState.Closed)
        {

            sqlconnection.Close();

        }

    }
    catch (MySqlException ex)
    {

        errorMessage = ex.Message;
        return false;

    }

    return true;
}

```

A Program.cs-ben levő kód, amit használunk az applikációban való közlekedésre a változóival.

```

static public StartScreen startscreen = null;
static public GameScreen gamescreen = null;
static public GameOverScreen gameoverscreen = null;
static public LeaderboardScreen leaderboardscreen = null;
static public PauseScreen pausescreen = null;
static public Database database = null;

static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    database = new Database("localhost", "root", "", "snakegamedb");
    pausescreen = new PauseScreen();
    gameoverscreen = new GameOverScreen();
}

```



```

gamescreen = new GameScreen();
leaderboardscreen = new LeaderboardScreen();
startscreen = new StartScreen();

startscreen.Show();

Application.Run();
}

```

Az alábbi metódus minden formon használva van, hogy biztosan minden rendben legyen bezárásukkal és felhozza a kérdő ablakot, ha kell.

```

private void StartScreen_FormClosing(object sender, FormClosingEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing && !userConfirmedClosing)
    {
        e.Cancel = true;

        using (var dialog = new CloseScreen())
        {
            this.FormClosing -= StartScreen_FormClosing; // remove event handler
            var result = dialog.ShowDialog();
            if (result == DialogResult.Yes)
            {
                userConfirmedClosing = true;
                Application.Exit();
            }
            else if (result == DialogResult.No)
            {
                userConfirmedClosing = false;
            }
            this.FormClosing += StartScreen_FormClosing; // re-add event handler
        }
    }
}

```

Az alábbi metódus ellenőrzi az irányokat és azok végét.

```

private void GameTimerEvent(object sender, EventArgs e)
{
    // irányok beállítása.
    if (goLeft)
    {
        Settings.directions = "left";
    }
    if (goRight)
    {
        Settings.directions = "right";
    }
    if (goDown)
    {
        Settings.directions = "down";
    }
    if (goUp)
    {
        Settings.directions = "up";
    }
}

```

```

}
// irányok vége.
for (int i = Snake.Count - 1; i >= 0; i--)
{
    if (i == 0)
    {
        switch (Settings.directions)
        {
            case "left":
                Snake[i].X--;
                break;
            case "right":
                Snake[i].X++;
                break;
            case "down":
                Snake[i].Y++;
                break;
            case "up":
                Snake[i].Y--;
                break;
        }
        if (Snake[i].X < 0 || Snake[i].X > maxWidth || Snake[i].Y < 0 || Snake[i].Y > maxHeight)
        {
            GameOver();
        }
        if (Snake[i].X == food.X && Snake[i].Y == food.Y)
        {
            EatFood();
        }
    }

    if (i > 0)
    {
        Snake[i].X = Snake[i - 1].X;
        Snake[i].Y = Snake[i - 1].Y;
    }
}

// Ellenőrzi, hogy a 'Snake' ütközik-e valamelyik akadállyal.
foreach (var obs in obstacles)
{
    if (obs != null && Snake[0].X == obs.X && Snake[0].Y == obs.Y)
    {
        GameOver();
    }
}

if (score % 5 == 0 && score > 0 && score > level * 5)
{
    level++;
    LevelLbl.Text = "Level " + level;

    // Létrehoz új akadályokat, ha a pontszám 10 vagy ha a pontszám 10 többszöröse.
    if (score == 5 || score % 5 == 0)
    {
        CreateObstacle();
    }
}

```

```

    }

    gamezone.Invalidate();
}

// Segítő metódus az akadályok készítésére.
private void CreateObstacle()
{
    // Adjon egy új akadályt az osztályhoz.
    obstacles.Add(new Circle { X = rand.Next(2, maxWidth), Y = rand.Next(2, maxHeight) });
}

public void RestartGame()
{
    maxWidth = gamezone.Width / Settings.Width - 1;
    maxHeight = gamezone.Height / Settings.Height - 1;
    Snake.Clear();
    obstacles.Clear();
    score = 0;
    level = 1;
    Scorelbl.Text = "Score: " + score;
    Levellbl.Text = "Level " + level;

    int middleX = maxWidth / 2;
    int middleY = maxHeight / 2;

    // Hozzáad 4 kört a fej és 3 testrész ábrázolására
    for (int i = 0; i < 4; i++)
    {
        Circle circle = new Circle { X = middleX - i, Y = middleY };
        Snake.Add(circle);
    }

    food = new Circle { X = rand.Next(2, maxWidth), Y = rand.Next(2, maxHeight) };

    GameTimer.Start();
}

```

Az alábbi metódus frissíti a játék felületet, hogy mindig helyesen legyen az.

```

private void UpdateGameGraphics(object sender, PaintEventArgs e)
{
    Graphics canvas = e.Graphics;
    Brush snakeColour;
    for (int i = 0; i < Snake.Count; i++)
    {
        if (i == 0)
        {
            snakeColour = Brushes.Black;
        }
        else
        {
            snakeColour = Brushes.DarkGreen;
        }
        canvas.FillEllipse(snakeColour, new Rectangle
        (
            Snake[i].X * Settings.Width,

```

```

        Snake[i].Y * Settings.Height,
        Settings.Width, Settings.Height
    ));
}
canvas.FillEllipse(Brushes.DarkRed, new Rectangle
(
    food.X * Settings.Width,
    food.Y * Settings.Height,
    Settings.Width, Settings.Height
));
for( int i = 0; i < obstacles.Count; i++)
{
    canvas.FillEllipse(Brushes.Yellow, new Rectangle
    (
        obstacles[i].X * Settings.Width,
        obstacles[i].Y * Settings.Height,
        Settings.Width, Settings.Height
    ));
}
}
}

```

Az alábbi metódus a 'Food' objektum felvételét ellenőrzi és annak megfelelően cselekszik.

```

private void EatFood()
{
    score += 1;
    Scorelbl.Text = "Score: " + score;
    Circle body = new Circle
    {
        X = Snake[Snake.Count - 1].X,
        Y = Snake[Snake.Count - 1].Y
    };
    Snake.Add(body);
    food = new Circle { X = rand.Next(2, maxWidth), Y = rand.Next(2, maxHeight) };
}
private void GameOver()
{
    GameTimer.Stop();
    GameTimer.Dispose();
    highScore = score;
    Finallevel = level;
    Program.gameoverscreen.ShowDialog();
}
}

```

3 Felhasználói dokumentáció

A programot a felhasználó letudja tölteni a SnakeSetup.exe-el és készen is áll a játékra.

A felhasználó vagyis játékos az úgynevezett „Snake” játékkal fogja szembe találni magát, amikor elindítja a programot.

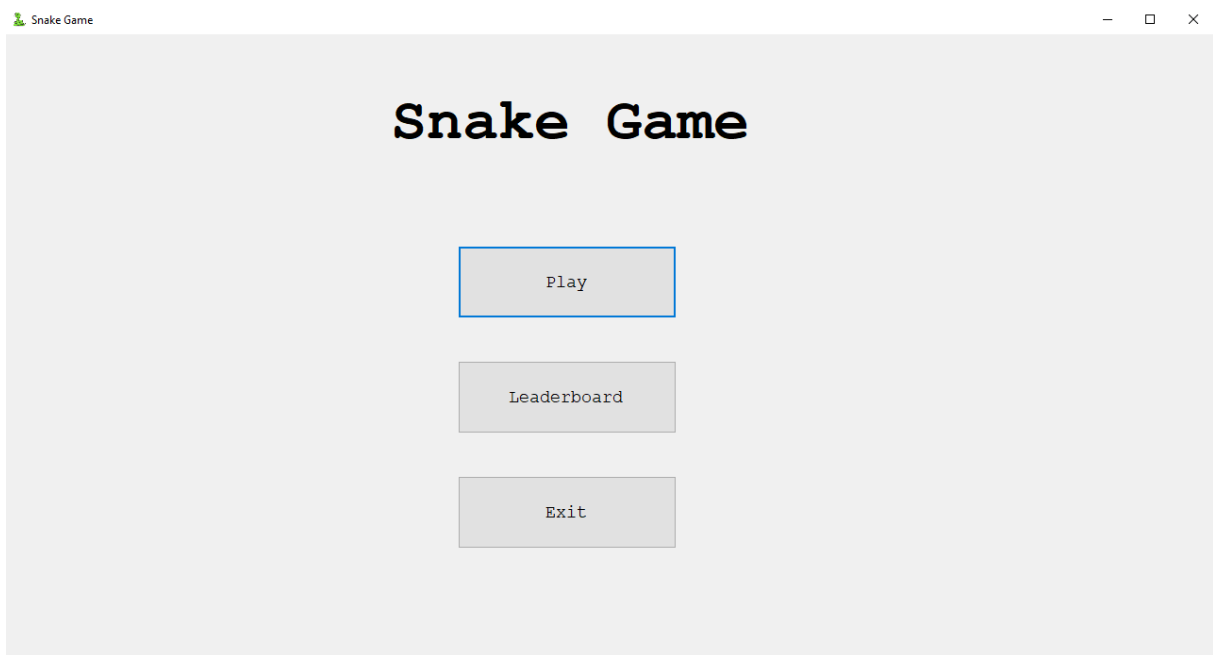
A Snake játék maga arról szól, hogy van egy úgynevezett „Snake”, aminek feladata, hogy minél több objektumot(ételt) vegyen fel a pontszám növeléséért és minél több legyen. Valamint, mint eközben ne menjen neki a különféle „ütközőknek” mármint akadályoknak vagy a pálya szélének, esetleg saját magának. Saját magának, azért mehet neki esetlegesen, mivel a Snake minden objektumtól nő és egyre hosszabb lesz, ezzel nehezítve annak irányítását és mozgását. Mindeközben ahogy pont száma nő, a szint is változni fog a pontszám alapján és több, máshogy elhelyezett akadályok lesznek. Ha mindez nem lenne elég, az objektumok randomizálva fognak a pályán megjelenni és mind addig azon ponton lenni amíg fel nem veszi azt a Snake, azután 1-2 perces intervallummal megjelenni egy új szint úgy randomizálva a pályán. A játéknak akkor van vége, amikor a Snake ütközik magával, a pálya szélével vagy egy akadállyal. Ekkor a játékosnak majd lehetősége lesz lementeni az elért végső eredményét, ami a végső pontszámát valamint a végső szintet jelenti. A Snake-t a játékos a billentyűzetten található le,fel,bal,jobb nyilakkal vagy WASD-vel tudja majd mozgatni a megfelelő irányba, hogy elkerülje a különféle ütközőket(akadályokat), pálya szélét vagy saját magát és persze, hogy feltudja venni a randomizálva megjelennő objektumokat, hogy pontszáma nőjön és nagyobb nehezebb szintre kerüljön.

A programhoz minimum egy számítógép kell és már lehet is használatba venni a programot. A programhoz kell .net framework csomag és egy SQL studió program, hogy a SQL adatbázis működjön a programmal.

A programot még nem lehet letölteni, de gépen elérhető lesz majd.

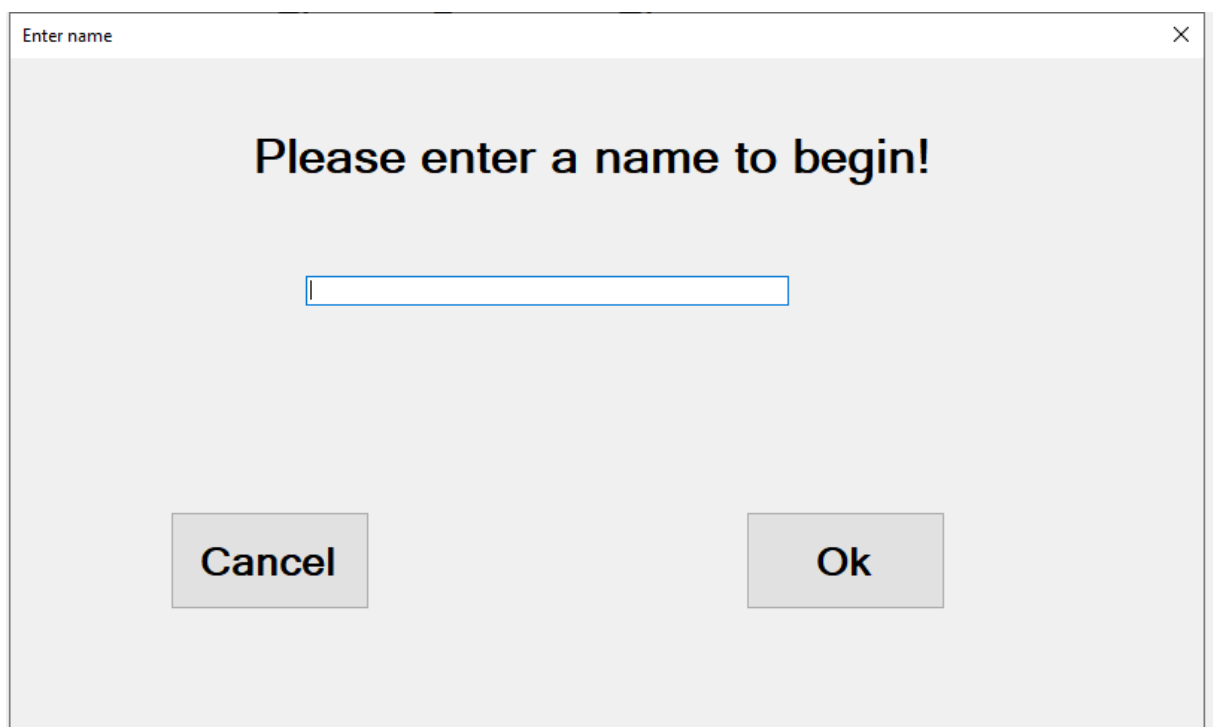
A program legtöbb funkciója elérhető mindenki számára, nem kell semmiféle speciális bejelentkezés vagy bármi, hogy elérjen bizonyos funkciókat, mivel a játék publikus úgymond „mókára” készült, így nincsenek funkció korlátozva többségbe.

Amikor a játékos elindítja magát az applikációt az úgynevezett „Snake Game”-t, szembe fogja találni magát a játék kezdőképernyőjével, itt lesz 3 lehetősége, amint lehet látni az alábbi képen.



6. ábra Kezdőképernyő

„Play”, „Leaderboard” vagy „Exit”. Ha a játékos rányom a „Play”-re, majd kap egy ablakot, hogy adjon meg egy játékos nevet, mint az alábbi képen látható.



7. ábra Név megadása

A név max csak 25 karakter hosszúságú lehet, kicsi vagy nagy betű nem számít, speciális karaktereket nem fogad el.

Miután a név meg lett adva a játékos bekerül a játék mezőre és indulhat a játék.



ábra 8 Játék képernyő

Az alábbi képen látható maga a játéklemező amin lesz majd a „Snake” és majd fognak megjelenni a randomizált objektumok és akadályok a szint alapján. A „Score: 0” fogja mutatni az éppen aktuális eredményét a játékosnak. A „Level 1” meg az aktuális szintet amin van.

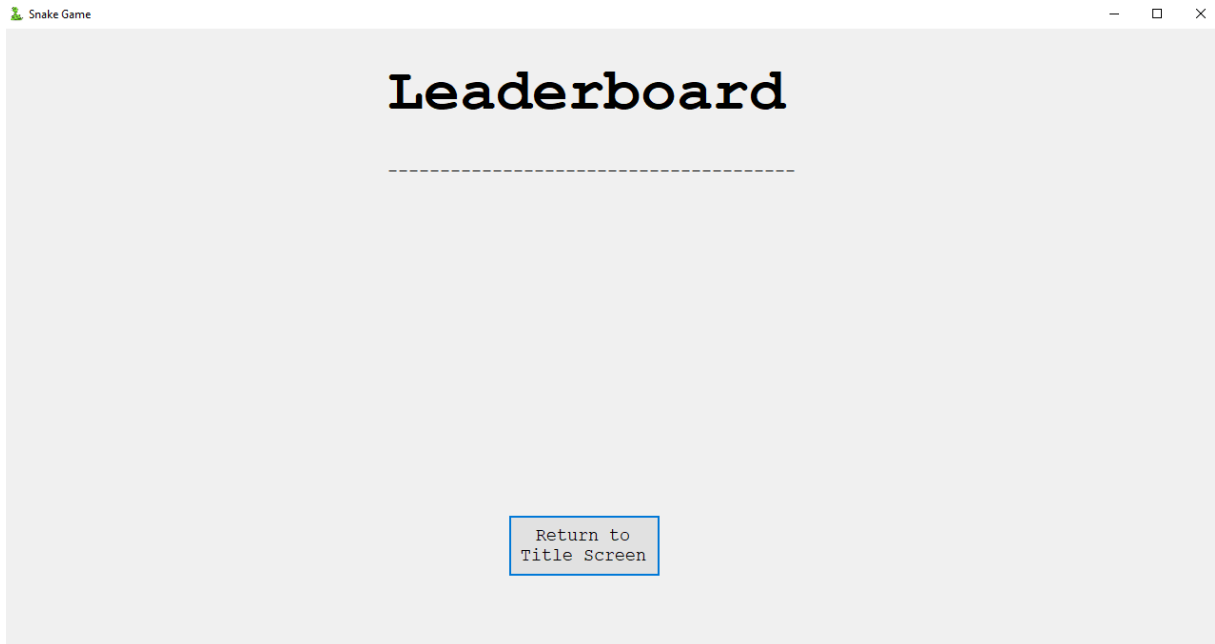


ábra 9 Game Over képernyő

Az alábbi kép mutatja az úgynevezett „Game Over” ablakot, amit majd a játékos fog látni amikor a játéknak vége lett, mert ütközött vagy egy adállyal, a pálya szélével vagy saját magával. A „Final Score: „ és a „Final Level: „ fogja mutatni a játék végén elért végső pontszámot és szintet. Ezután a játékosnak lesz 3 lehetősége: „Save”, „Play Again” vagy „Return to Title Screen”. A „Save” megnyomása után a játék lementi az eredményét nevével

együtt. A „Play Again” megnyomása után a játékos egy újabb játékba kerül ugyan azzal a névvel és próbálkozhat nagyobb pontszámot és szintet elérni, mint előzőleg. A „Return to Title Screen” megnyomása után a játékos visszasikerül a kezdőképernyőre és utána vagy rámeget a „Play”-re megint vagy megnézheti esetlegesen a „Leaderboard”-ot.

Amikor a játékos rányom a „Leaderboard”-ra, az alábbi képen látható ablak fog megjelenni ami majd mutatni fogja a „Top 10” játékost, elért eredményeik alapján a „Leaderboard” felirat alatt. Miután eléggé átnézte lehetősége lesz vissza menni a játék kezdőképernyőjére a „Return to Title Screen” megnyomásával.



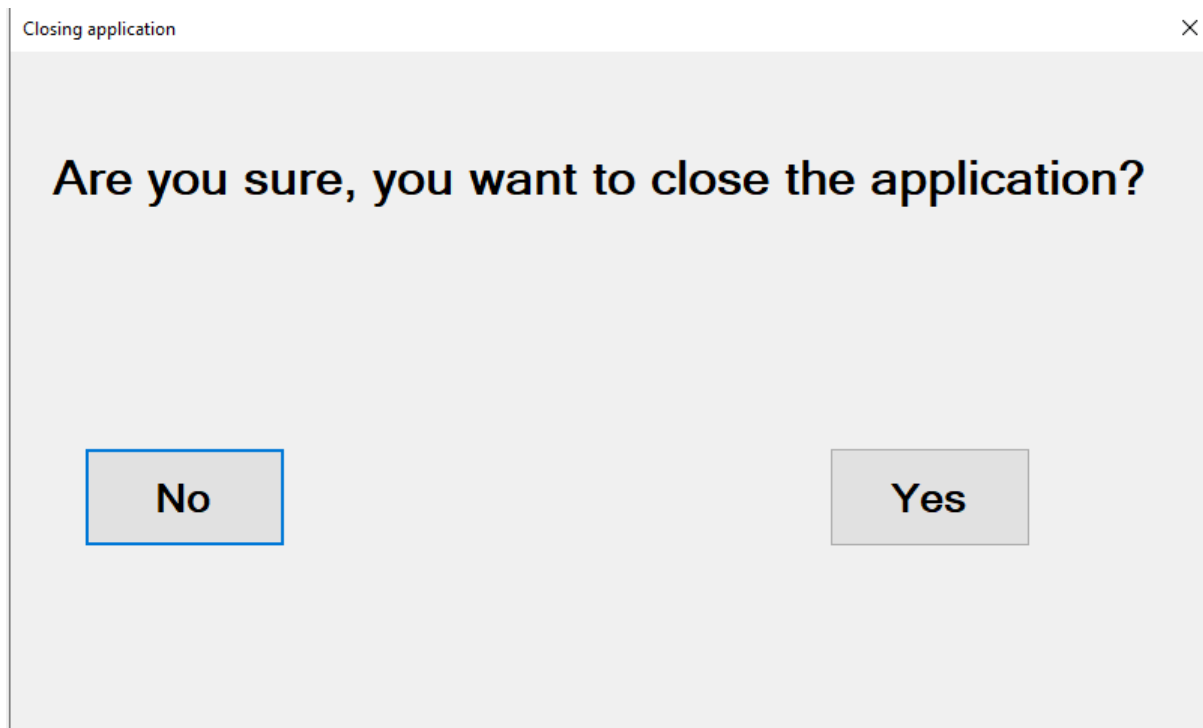
ábra 10 Leaderboard képernyő

Amikor a játékos rányom a „Exit”-re akkor kilép az egész applikációból, mármint bezárja azt.

4 Tesztelői dokumentáció

Az alábbi problémákkal futhat össze a felhasználó és az alábbi módon vannak azok kezelve.

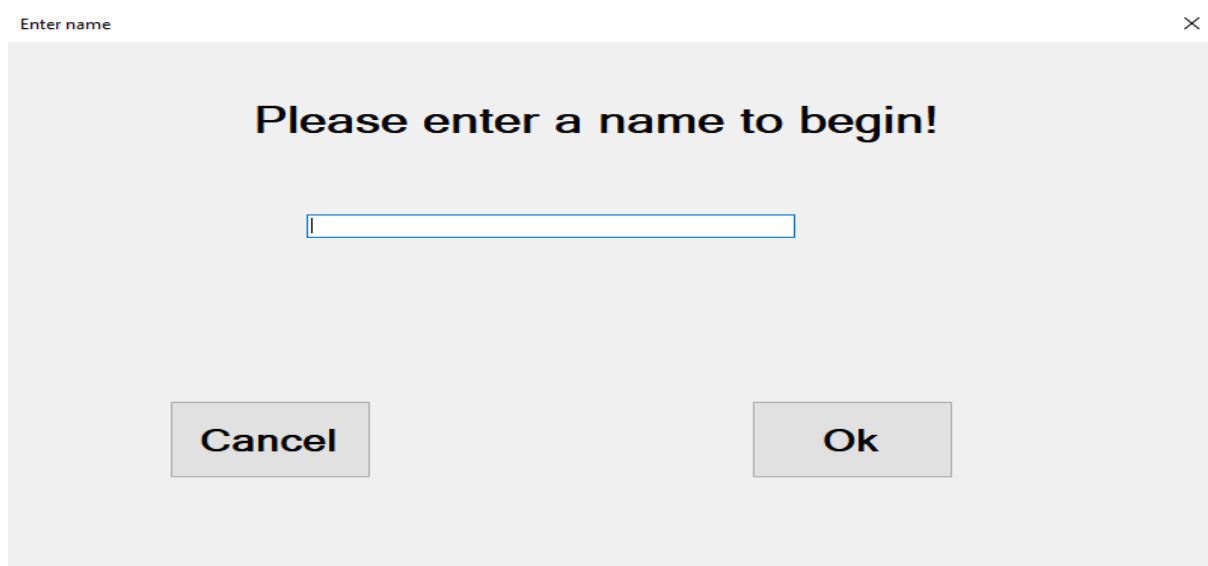
Ha a felhasználó mondjuk megnyomja a „Exit” gombot vagy ki akar lépni az alkalmazásból a X gomb megnyomásával az alábbi ablak fog megjelenni mindig.



ábra 11 Biztos ki akar-e lépni kérdő ablak

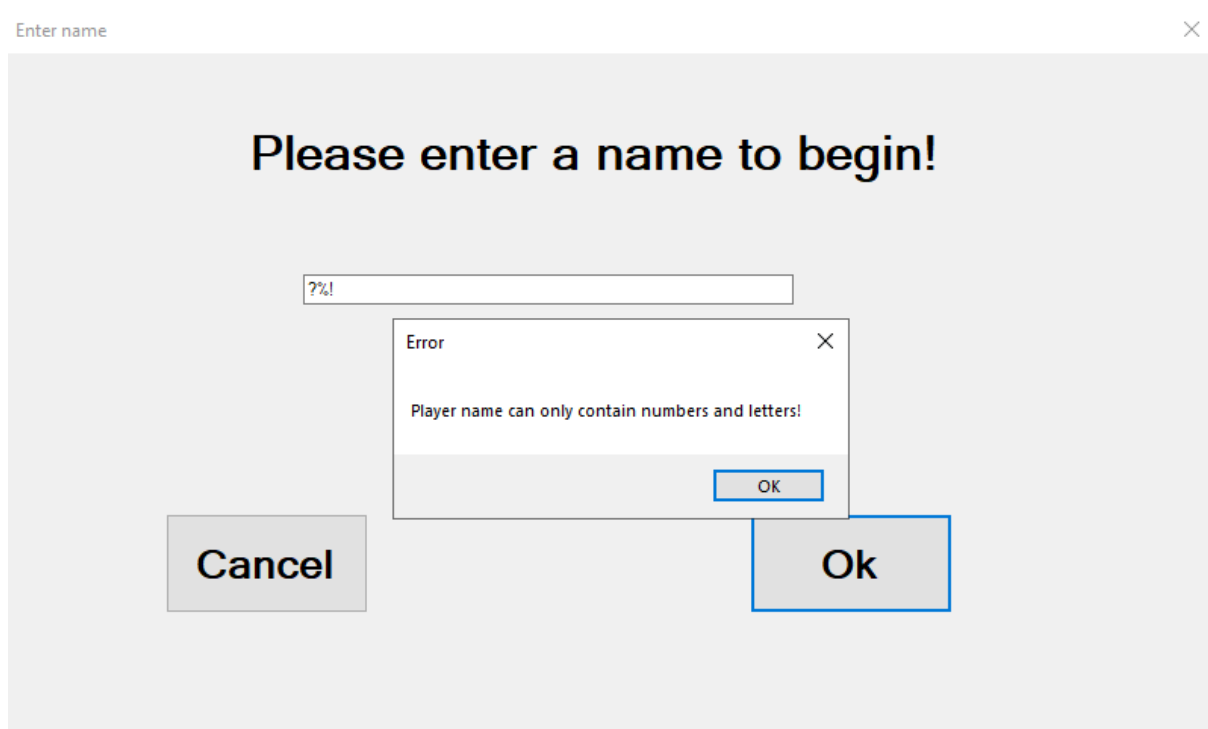
Ez az ablak megkérdi a felhasználót, hogy ténleg ki akar-e lépni, ha igen mármint a „Yes”-re nyom akkor az egész alkalmazásból kilép, ha meg nem, mármint a „No”-ra nyom akkor visszalép ahol volt és minden folytatódik amilyen módon kéne.

Amikor a játék a nevét akarja megadni akkor lehetséges, hogy kis-nagy betűket használ vagy szimbólumokkal akar próbálkozni, estleg nem ír be semmit amikor csak betűket vagy számokat lehet használni, az alábbi ablaknál ami a nevet kéri:



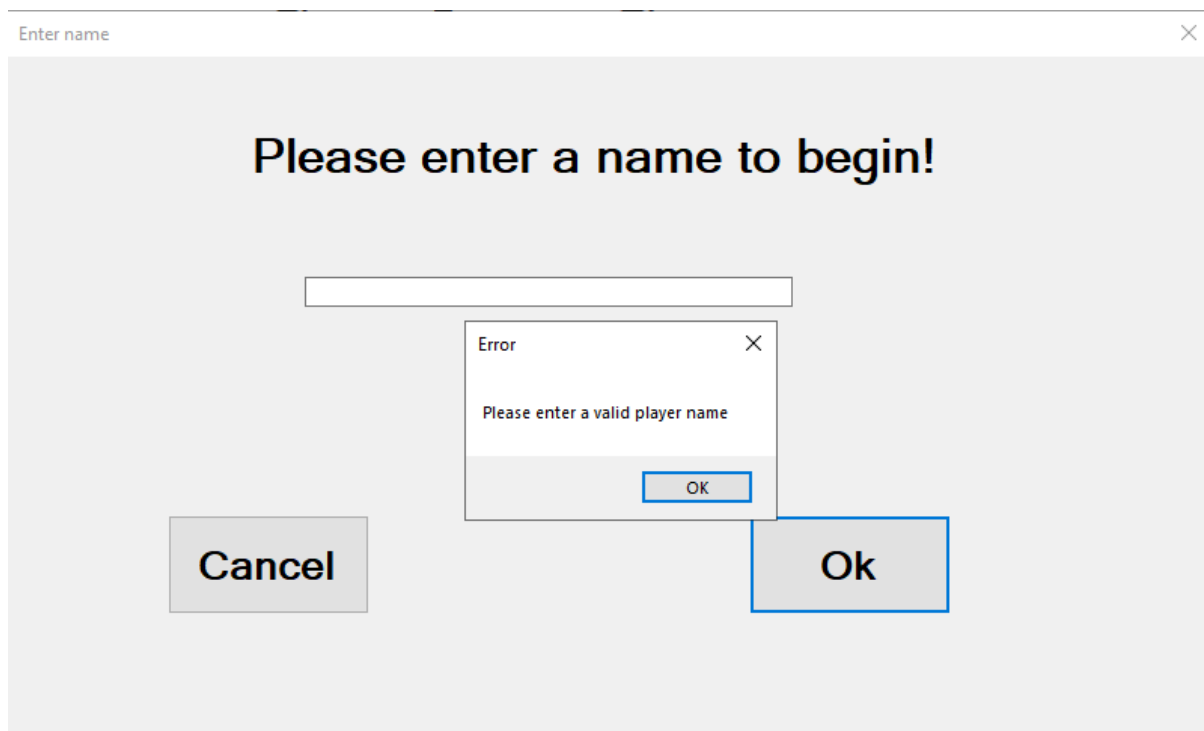
ábra 12 Névadó ablak

A probléma az ablaknál úgy van megoldva ha a felhasználó szimbólumokat próbál használni akkor felhossa az alábbi üzenetet és visszaküldi, hogy adjon meg egy elfogadható nevet.



ábra 13 Névadó ablak, ha szimbólumok hiba

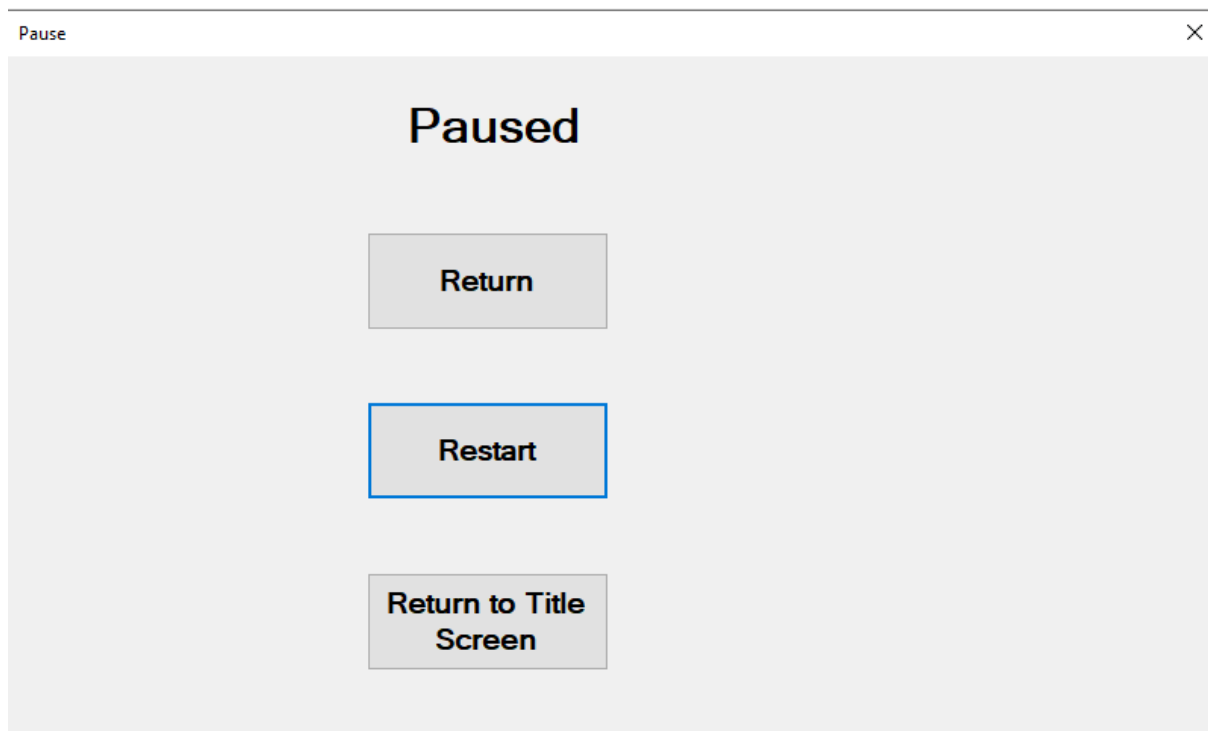
Ha meg üresen hagyja akkor az alábbi és ugyanúgy visszaküldi, hogy adjon meg egy elfogadható nevet.



ábra 14 Névadó ablak, ha üres hiba

Ha esetleg kis-nagy betűket használ az nem probléma mivel elfogadja ha kicsi, ha nagy probléma nélkül a program.

A játékos esetlegesen megakarja állítani a játékot, mert menet közben lehet újrat akar vagy netán kis pihenőt akar tartani vagy visszamenni a kezdőképernyőre. Erre van egy „PauseScreen” elkészítve, amit elő tud hozni a játékos az „Esc” megnyomásával.



ábra 15 Pause ablak

Persze ha itt is rányom a X gombra akkor megkérdi, hogy ki akar-e lépni, mint a többenél. Ha rányom a „Return” gombra a játék újra kezdődik ahol volt. Ha a „Restart” gombra akkor az egész újra kezdődik az elejétől, a „Score” le nullázódik és a „Level” megint 1 lesz és a Snake alap állapotba kerül, mint ahogy volt a játék elején. Ha meg a „Return to Tile Screen” gombra akkor visszaviszi a játékos a kezdőképernyőre.

5 Összefoglalás

Úgy érzem a programom összeségében egy jó kis fejlesztő program az emberek számára. A régi „Snake” játék elemét felhasználva, egy kis változással felpördíti és még izgalmasabbá és fejlesztő típusúvá teszi.

A folyamatosan növekvő „Snake” és a pontok alapján nehezdő szintek és mennyi akadály van a pályán attól függően, nehezebbé teszik az élményt és még fejlesztőbbé. Ugyanis a játékosnak nem csak arra kell oda figyelni, hogy a pálya szélének ne menjen neki, saját magának, hanem még a megjelenő akadályoknak se. Úgy érzem az applikációm segítőkész és szórakoztató, de ugyanakkor fejlesztő is.

Ezzel még a magam programozási tudását is bővítettem és próbára tettem azt.

A program elkészítése közben voltak kisebb – nagyobb problémák, de mindenre találtam megfelelő megoldást. Például a program több formában van megoldva, emiatt a problémás volt a `Close();` metódus, de sikerült a legtöbb Formnál megoldani, kivéve a `GameScreen`-nél megoldani. A név bekérést először sima `ShowDialog();`-gal próbálkoztam megoldani, de később készítettem egy Formot, ami dialógusként szolgál a név bekéréséhez, ugyanaz ez volt a `PauseScreen`-el is, ott persze nem név bekérést, hanem a játék újraindítását, folytatását vagy belőle való kilépést szolgálja, mindeközben a játék áll és a Snake nem mozog. A snake játék felülete átment némi változáson, eleinte nem volt a játék terület látható és a Snake át tudott menni saját magán vagy a pálya széléin mindegyik gond nélkül. Ezt később javítottam és most már látható a terület, a Snake nem tud átmenni magán vagy a játék felület széléin. Az akadályokat a program készítése során változtattam, hogy mennyi jön le mikor és a játék elején lejön egy fix mennyiség, a pálya nagysága miatt. A játék felület résznél gondolkodtam a Snake gyorsaságának növelésében idővel, de egyelőre még nem tettem meg. A programhoz eleinte SQL adatbázist használtam, de később át álltam a CSV fájlra, a program optimalizált működése érdekében. A program készítettem egy `SnakeSetup.exe`-ét, amivel a felhasználó letudja tölteni a játékot és már tud is játszani vele. Eleinte az alkalmazásnak nem volt háttér, csak egy fehér alap háttér volt, ezt később megváltoztattam amit mostanin látni lehet. A `SnakeSetup.exe`-ét megoldottam hogy lehessen választani, hogy a felhasználó akar-e asztali parancsikont vagy start menü parancsikont.

6 Irodalmi jegyzék

Mooict - <https://www.mooict.com/c-tutorial-create-a-classic-snake-game-in-visual-studio/>

Github

Youtube

Google

6.1 Ábrajegyzék

ábra 1 Xampp szoftver.....	4
ábra 2 DiaPortable szoftver.....	4
ábra 3 Visual Studio szoftver	5
ábra 4 Adatbázis ábra.....	6
ábra 5 ER diagram	6
6. ábra Kezdőképernyő.....	14
7. ábra Név megadása	14
ábra 8 Játék képernyő	15
ábra 9 Game Over képernyő	15
ábra 10 Leaderboard képernyő	16
ábra 11 Biztos ki akar-e lépni kérdő ablak.....	17
ábra 12 Névadó ablak.....	18
ábra 13 Névadó ablak, ha szimbólumok hiba	18
ábra 14 Névadó ablak, ha üres hiba	19
ábra 15 Pause ablak.....	20

6.2 Jegyzet

Contents

Snake alkalmazás.....	1
1.Bevezetés	2
2.Fejlesztői dokumentáció.....	3
Használt hardver	3
Használt szoftver.....	3
Adatbázis	6
3.Felhasználói dokumentáció.....	13
4.Tesztelői dokumentáció	17
5.Összefoglalás.....	21
6.Irodalmi jegyzék.....	22
Ábrajegyzék.....	22
Jegyzet.....	22