Nagykanizsai SZC
Zsigmondy Vilmos Technikum

ZÁRÓDOLGOZAT

Cowboy

Sajni Bercel
Szoftverfejlesztő szak
2024

1 Bevezetés:

1.1 Köszönet Nyilvánítás

Szeretném megköszönni Pető Zsolt, Trencséni Eliot, Smolcz Frigyes tanár úrnak és Deák Istvánné tanárnőnek. Az ők fáradhatatlan munkájuk nélkül és segítségük nélkül nem jöhetett volna létre ez a program.

1.2 A Program Témája

Egy kicsit retro hangulatot idéző 2 dimenziós lövöldözős játékot szerettem volna csinálni, retro kinézettel és hangokkal. A játékot lehet játszani egyedül vagy egy baráttal.

1.3 Indoklás

Azért egy ilyen stílusú program elkészítése mellett döntöttem, mivel ezeknek a régi stílusú játékoknak van egy egyedi nem könnyen felejthető aurája (mint például a doom-nak) és a modern fejlesztői eszközökkel és az interneten felhalmozott és rendelkezésre álló ismeretekkel egy megvalósítható feladatnak tűnt amellett remélem hogy úgy sikerült megvalósítanom hogy közben mindenki számára be tudom mutatni a jelenlegi képességeim egy szeletét.

1.4 Játék Célja

A cél hogy a 2 játékos lelője egymást. Miközben próbálják kikerülni az ellenség töltényeit, aki lehet valós játékos vagy a számító gép által irányított karakter. Aki életben marad az nyer.

2 Felhasználói Dokumentáció

2.1 Rendszerkövetelmények:

Operációs rendszer: Windows 7 (vagy újabb)

Minimum RAM: 128MB

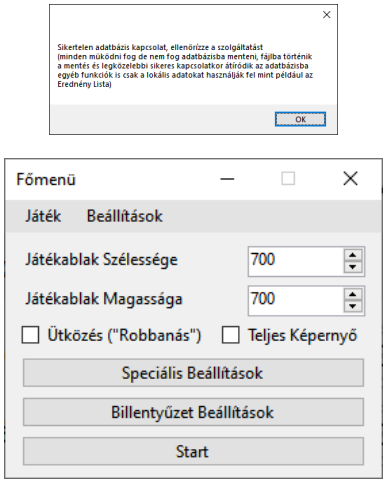
2.2 A program elindítása

„cowboy.exe”-t kell futtatni (tartalmaz minden futáshoz szükséges programot/kellét, tehát nincs szükség másra az exe fájlon kívül). Érdemes lehet rendszergazdakén vagy rendszergazai hozzáféréssel futtatni (mivel fájlba írást is végez, ezért ez nélkül léphetnének fel hibák).

2.3 Főmenü

Először a főmenü nyílik meg. A megnyitás közben ellenőrzi az adatbázis kapcsolatot, ezért az ablak indítása előtt lehetséges hogy megjelenik egy hibaüzenet ha nem jött létre az adatbázis

kapcsolat. Itt lehet elvégezni a játékra vonatkozó beállításokat. Illetve tartalmaz még néhány egyéb apró funkciót.



2.3.1 Főmenü Menüsor

A menüsor az ablak tetején foglal helyet. Tartama:

- Játék
  - Konfigurációs Mappa Megnyitása
  - Eredmény Lista
  - Info
- Beállítások
  - Speciális Beállítások
  - Billentyűzet Beállítások

2.3.2 Játék Menügomb

A „Konfigurációs Mappa Megnyitása” gomb megnyitja azt a mappát ami tárolja a mentett beállításokat.

Az „Erdemény Lista” gomb megnyit egy új ablakot az előző Játékok eredményeit (ha nincs adatbázis kapcsolat akkor csak a helyileg mentett játékok adatait mutatja).

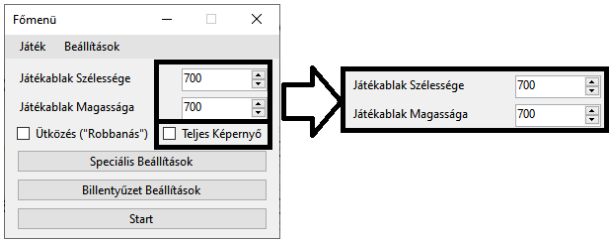
Az „Info” gomb megnyit egy új ablakot ami egy kevés információt tartalmaz a programról.

2.3.2.1 Beállítások Menügomb

A „Speciális Beállítások” gomb megnyitja a Speciális Beállítások ablakot

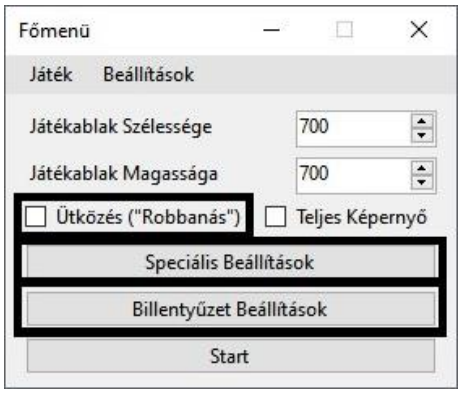
A „Billentyűzet Beállítások” gomb megnyitja a Billentyűzet Beállítások ablakot

2.3.3 Játék Ablak Méretének Beállítása



ezzel a két menüponttal beállítható a játékablak szélessége és magassága hogy mindenki az ízlésének megfelelő méretű játékteret generálhasson magának. Minimum 300x300, maximum 2000x2000 (itt már jelentősen romlik a játék élmény). Illetve a „Teljes Képernyő” jelölő négyszettel lehet be/ki kapcsolni hogy a játék teljes képernyős legyen (ilyenkor természetesen nem lehet kézzel állítani az ablak méretét).

2.3.4 Játékmenet Testre Szabása



Az „Ütközés” jelölő négyzettel be lehet kapcsolni hogy a játék közben a töltények ütközzenek/robbanjanak amikor egymásnak mennek.

A „Speciális Beállítások” gomb megnyitja a Speciális Beállítások ablakot.

A „Billentyűzet Beállítások” gomb megnyitja a Billentyűzet Beállítások ablakot.

2.4 Beállítás ablakok

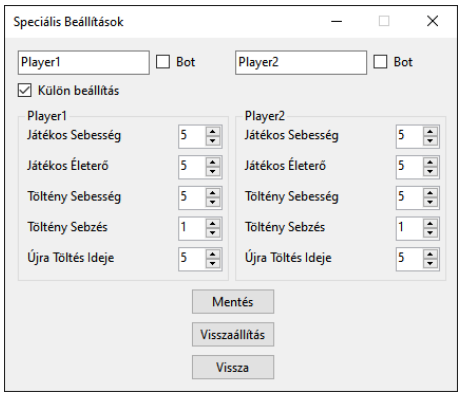
- Speciális Beállítások
- Billentyűzet Beállítások

2.4.1 Speciális Beállítások

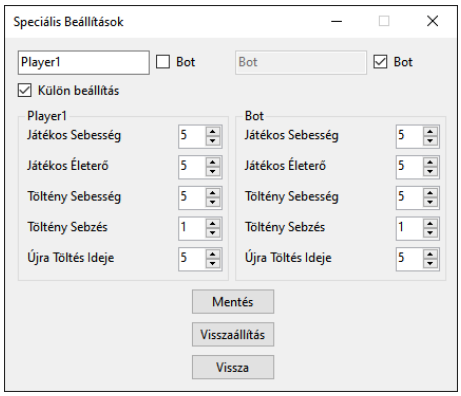
A játékos(ok) tulajdonságait lehet személyre szabni.

2.4.1.1 Beállítható Tulajdonságok

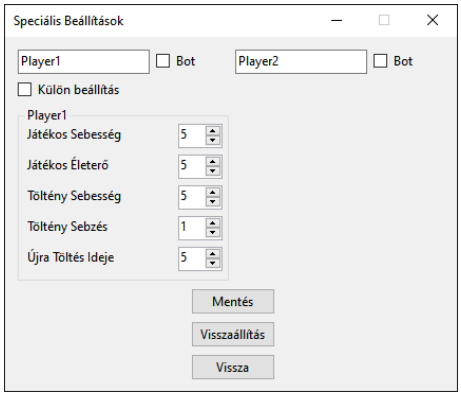
- Játékosok Sebesség
- Játékos Életerő
- Töltény sebesség
- Töltény Sebzés
- Újra töltés Ideje



Az ablak tetején két szöveges mezőben be lehet állítani a játékosok nevét: Mellette hogy ne egy felhasználó által irányított játékossal, hanem a játék által irányított (bot) ellenséggel induljon a játék. Ilyenkor a név helyett a „Bot” felirat jelenik meg a szöveges mezőben és nem lehet módosítani a tartalmát.



Alatta egy „külön beállítás” jelölő négyzettel lehet be/ki kapcsolni hogy külön lehessen beállítani a két játékos tulajdonságait.



Ilyenkor az első játékosra beállított tulajdonságokat kapja meg a másik játékos is.

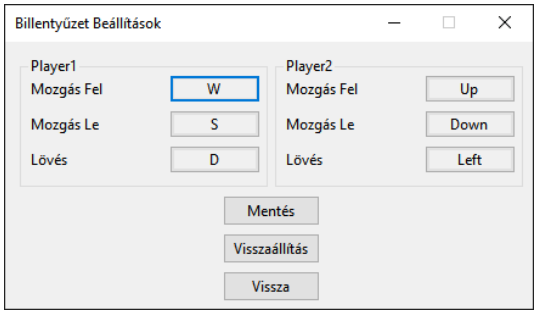
Az ablak alján három gomb van. A mentés gomb menti a beállítások, különben nem ment a program. A Visszaállítás gomb visszaállítja a tulajdonságokat egy előre meghatározott alap értékekre. A Vissza gomb visszalép a Főmenübe.

2.4.2 Billentyűzet Beállítások

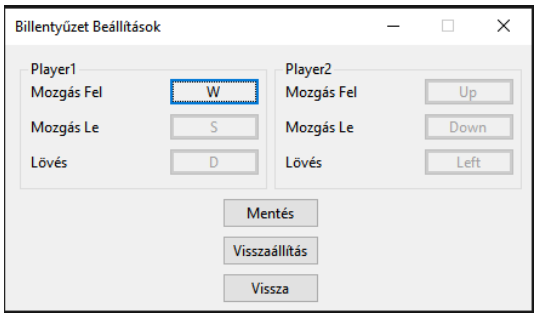
A billentyűzet beállításait lehet személyre szabni.

2.4.2.1 Beállítható Billentyűk

- Mozgás Fel
- Mozgás Le
- Lövés

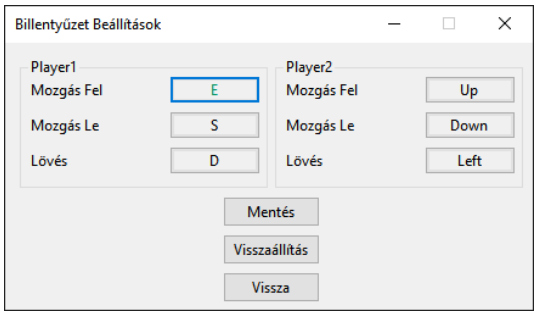


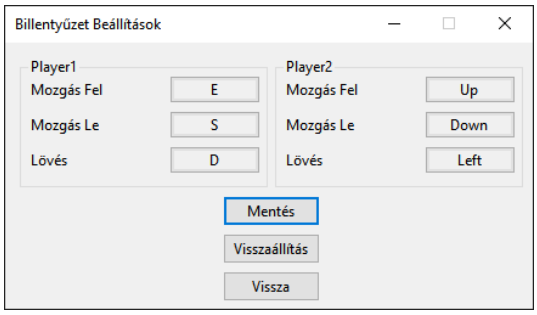
Ha kiválaszt egy gombot a felhasználó hogy átállítsa akkor a többi gomb nem kiválasztható lesz.



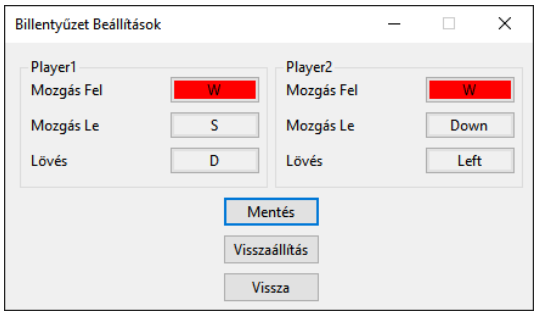
Ilyenkor a következő billentyűzet gomb lenyomásával beállítja azt mint új Billentyű beállítás. Kivéve ha „Esc”-et nyom a felhasználó mivel ezzel meg tudja szakítani folyamatot ha meggondolja magát a felhasználó. Ilyenkor az ablak visszaáll alaphelyzetbe.

Ha megváltoztat egy beállítást akkor zöld színű lesz a felirat mint addig ameddig nem menti el a felhasználó a mentés gombbal.





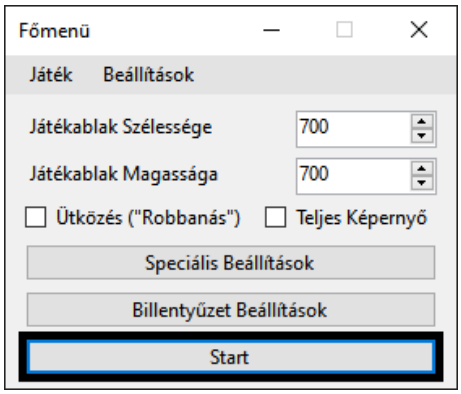
Ha kettő vagy több helyre is ugyan azt a billentyűt szeretné beállítani a felhasználó akkor piros hátteret kapnak azok a gombok ahol ismétlés van. A beállítás ettől még menthető és használható.



Az ablak alján három gomb van. A Mentés gombbal el lehet menti a beállításokat, egyébként magától nem ment a program. A Visszaállítás gombbal egy előre meghatározott alapbeállításokat tölt be a program. A Vissza gombbal vissza lép a Főmenübe.

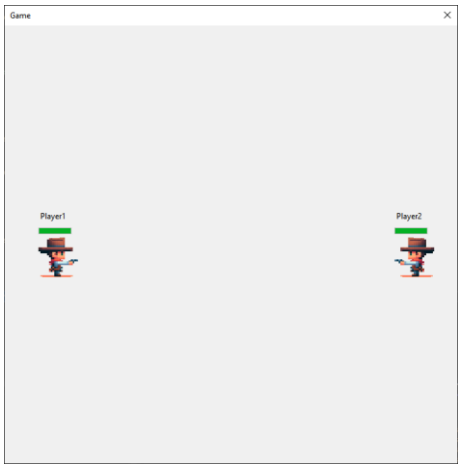
2.5 Játék indítása

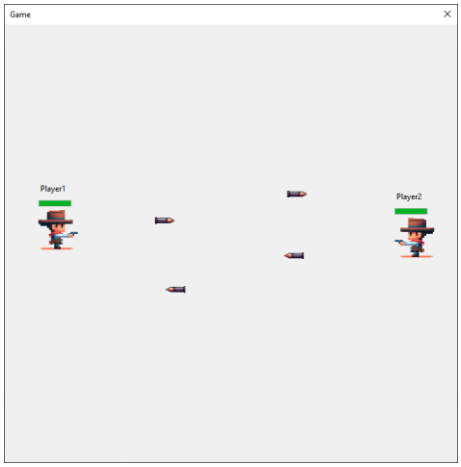
Főmenüben a Start gombbal lehet elindítani a játékot ha már megvagyunk győződve róla hogy mindent beállítottunk amit szerettünk volna.



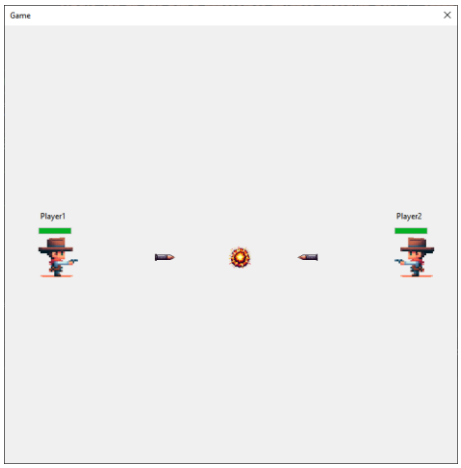
2.6 Játék Ablak

A kezdőállapot hogy a két játékos az ablak két oldalán helyezkedik el egymással szemben állva. A Főmenünk belül a Billentyűzet Beállítások ablakban beállított billentyűkkel tudják a karaktereiket irányítani.





A „Mozgás Fel” gombbal felfelé, a „Mozgás Le” gombbal lefelé tudják mozgatni a karaktereiket a már beállított Sebességgel, hogy kikerüljék az ellenség töltényeit, vagy hogy célba vegyék az ellenséget. Lőni a „Lövés” billentyűvel lehet, ilyekor a karakter ki lő egy töltényt ami már beállított Sebességgel halad és ha eltalálja az ellenfelet akkor a neki beállított Sebzés mennyiséggel megsebzi úgy hogy ez a mennyiség levonódik a játékos Életerejéből. A Játékos Életereje és neve a karaktere felett van. Illetve a lövés után meg kell várni az újra töltés idejét mielőtt megint tüzelhetnénk. Ha be van kapcsolva az „Ütközés” akkor egy kis robbanás effekt jelenik meg a 2 töltény ütközésének a helyénél.



A „Space” billentyűvel megállítani és elindítani lehet a játékot. Míg az „Esc” billentyűvel vissza lehet lépni a „Főmenü”-be. Ha másik ablakot vagy programot, nyit meg vagy indít el a felhasználó vegy tálcára teszi a játékot vagy nem a játék ablak az elsődlegesen kiválasztott ablak akkor is meg lesz állítva a játék.

3 Fejlesztői Dokumentáció

3.1 Fejlesztő környezet

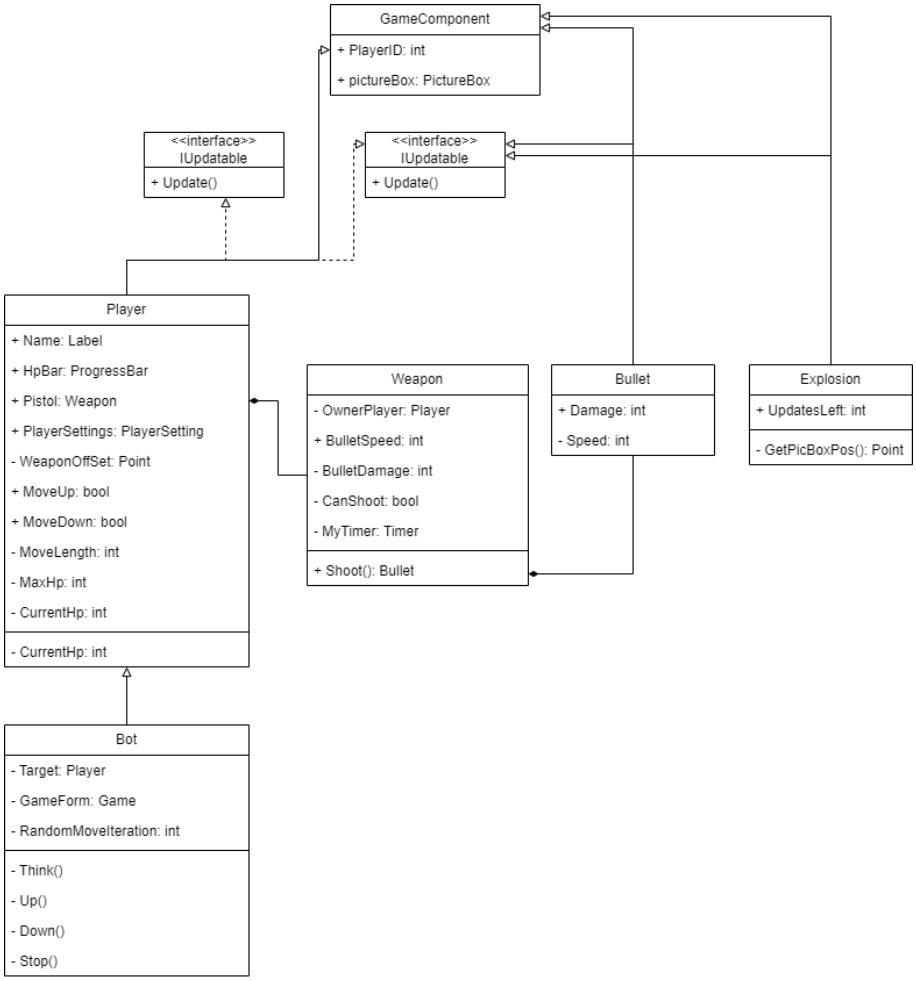
A programot c# programozási nyelven .net core 8.0 környezetben készítettem Visual Studio 2022 winform form projcet-ként.

3.2 Mappa/Fájl Rendszer

- Classes
  - Bot.cs
  - Bullet.cs
  - Explosion.cs
  - GameComponent.cs
  - Player.cs
  - Weapon.cs
- Forms
  - AdvancedSettings.cs
  - Game.cs
  - InputSettings.cs
  - MainMenu.cs
  - ScoreBoard.cs
- Interfaces
  - IHitable.cs
  - IUpdatble.cs
- Settings
  - GameSettings.cs
  - InputSetting.cs
  - PlayerSetting.cs
- Utilities
  - Create.cs
  - DbManager.cs
  - FileManager.cs

3.3 Classes

A „Classes” mappán belül lévő osztályok mind a „játék” részét képezik, kakukk tojás a „Weapon” mivel ő nem nincs leszármaztatva a „GameComponent” osztályból, ellentétben a többi osztállyal akik mind kiterjesztik a „GameComponent” osztályt. Az osztálydiagram (az interface-ekkel kiegészítve) bemutatja hogy hogyan van felépítve az osztályok öröklési rendszere.



3.3.1 GameComponent

Tartalmaz egy „pictureBox”-ot ami a vizuális megjelenése lesz a komponenseknek. Illetve a fizikai valójuk is egyeben mivel ennek a „pictureBox”-nak a méreteit használja fel az ütközés figyelő rész. Ezek mellett tartalmaz egy „PlayerID”-t ami ahhoz kell hogy meg lehessen állapítani bármelyik komponensről hogy melyik „Player”-hez tartozik (pl.: hogy nem tudja saját magát meglőni egyik ”Player” se csak másik játékost tudjon eltalálni).

3.3.2 Player

A Player osztály. Tud fel/le menni, és lőni, ezekre a „parancsokat” a „Game” Form-ból kapja (mivel ez a Form kezeli le a billentyű lenyomásokat). A fel/le mozgás úgy történik hogy a 2 publikus logikai értéket tároló változó „MoveUp” és a „MoveDown” a „Game” Form igazra vagy hamisra állítja a billentyűzetről beolvasott adatok alapján. Majd az „Update” metódusban

ezeknek a logikai változóknak megfelelően mozgatja a PictureBox-ot fel vagy le, és vele együtt a „Player”-hez tartozó egyék dolgokat, az életerő csíkját (HpBar) és a felhasználó álltal beállított név kiírást (Name).

```
3 references
public virtual void Update()
{
    Move();
}

1 reference
protected void Move()
{
    if (MoveUp)
        pictureBox.Top -= MoveLength;
    if (MoveDown)
        pictureBox.Top += MoveLength;

    MoveHUD();
}
```

3.3.3 Bot

Fő feladata hogy irányítsa a neki kijelölt játékost (hogy azt az érzést keltse a felhasználóban hogy a számítógép ellen játszik). Örököl a „Player”-ből és kiegészít/módosít a működésén annyival hogy a mozgáshoz nincs szüksége billentyűzet gomb lenyomásokra. Hanem belül dönti el magától hogy hogyan mozogjon a „Think()” függvény irányítja ezt a mozgást és mindig lő amikor vele szemben van az ellenség. Majd meghívja az eredeti („Player”-hez tartozó) „Update”-et ami végzi a mozgatást.

```
/// <summary>
/// eredeti Player Update-et használja,
/// ahelyett hogy külső input alapján cselekedne ő "gondolja végig" hogy mit csináljon
/// </summary>
4 references
public override void Update()
{
    Think();
    base.Update();
}

/// <summary>
/// Bot-nak az "agya" itt dől el hogy mit csináljon
/// </summary>
1 reference
private void Think()
{

    if (Target.pictureBox.Location.Y + Target.pictureBox.Height < this.pictureBox.Location.Y)
        Up();
    else if (Target.pictureBox.Location.Y - Target.pictureBox.Height > this.pictureBox.Location.Y)
        Down();

    else if (Target.pictureBox.Location.Y - Target.pictureBox.Height/2 < this.pictureBox.Location.Y &&
        Target.pictureBox.Location.Y + Target.pictureBox.Height/2 > this.pictureBox.Location.Y)
    {
        GameForm.Shoot(this);
        Random rnd = new Random();
        int randomMovement = rnd.Next(3);
        RandomMoveIteration++;
        if (RandomMoveIteration == 40)
        {
            switch (randomMovement)
            {
                case 0: Up(); break;
                case 1: Down(); break;
                case 2: Stop(); break;
            }
            RandomMoveIteration = 0;
        }
    }
}
```

3.3.4 Weapon

Célja hogy ez végezzel el a lövést (a „Bullet” létrehozását) a „Shoot()” függvénnyel és visszatér egy új „Bullet” objektummal és lejátszik egy hang effektet, kivéve ha a úgy lett megívva a lövés hogy a fegyver még újratölt, ilyenkor „null”-t ad vissza és nem csinál semmit.

```
/// <summary>
/// Elsüti a fegyvert, tud "null"-t visszaadni amikor a fegyver éppen nem tud lőni
/// </summary>
/// <returns>Bullet</returns>
1 reference
public Bullet? Shoot()
{
    if (CanShoot)
    {
        SoundPlayer soundPlayer = new SoundPlayer(Resources.shootSound);
        soundPlayer.Play();
        soundPlayer.Dispose();

        Bullet bullet = new Bullet(OwnerPlayer.PlayerID,
            Create.pictureBox("Bullet", new Size(30, 30), new Point(0, 0),
            Properties.Resources.bullet), BulletSpeed, BulletDamage);
        Point offset = OwnerPlayer.GetWeaponOffSetPoint();
        offset.Offset(0,-bullet.pictureBox.Height/2);
        bullet.pictureBox.Location = offset;
        CanShoot = false;
        MyTimer.Start();
        return bullet;
    }
    return null;
}

1 reference
private void SetShootTrue(object? sender, EventArgs args)
{
    CanShoot = true;
    MyTimer.Stop();
}
```

3.3.5 Bullet

Az Inicializálása után a neki megadott sebességgel halad „előre”. Fontos feladatot lát el a „IsInScreen(int width)” publikus, logikai értékkel visszatérő függvény, megadja hogy a töltény a monitoron van-e ehhez szüksége van az ablak szélességére amit mint bemeneti paraméter kap meg, ez majd abban fog segíteni hogy törölhessük ha tudjuk hogy nincs már rá szükség.

```
/// <summary>
/// A mozgatást végzi
/// </summary>
2 references
public void Update()
{
    pictureBox.Left += Speed;
}

/// <summary>
/// a törlésnél lesz szerepe
/// </summary>
/// <param name="width"></param>
/// <returns>bool, a képernyőn van-e</returns>
1 reference
public bool IsInScreen(int width)
{
    if (pictureBox.Left < -pictureBox.Size.Width || pictureBox.Left > width)
        return false;
    return true;
}
```

3.3.6 Explosion

Akkor Inicializálódik amikor be van kapcsolat az ütközés a töltények között és két töltény össze ütközik (a „Game” Form inicializálja). A konstruktorban átveszi a két össze ütközött „GameComponent”-et és pontosan a 2 közé helyezi el a „robbanást” és lejátszik egy hang effektet.

```
/// <summary>
/// Középre (a 2 komponens közé) helyezi az effektet
/// </summary>
/// <param name="BulletA">komponens 1</param>
/// <param name="BulletB">komponens 2</param>
/// <returns>Point, középpont</returns>
1 reference
private Point GetPicBoxPos(GameComponent BulletA, GameComponent BulletB)
{
    int x1 = BulletA.pictureBox.Location.X;
    int y1 = BulletA.pictureBox.Location.Y + BulletA.pictureBox.Height / 2;

    int x2 = BulletB.pictureBox.Location.X;
    int y2 = BulletB.pictureBox.Location.Y + BulletB.pictureBox.Height / 2;

    int x = (x1 + x2) / 2;
    int y = (y1 + y2) / 2;

    y -= this.pictureBox.Height / 2;

    return new Point(x,y);
}
```

3.4 Forms

A „Forms” mappán belül a program által használt összes Form megtalálható található. Természetesen minden az alap .net-es „Form” osztályból örököl.

Forms ablakok:

- AdvancedSettings.cs
- Game.cs
- InputSettings.cs
- MainMenu.cs
- ScoreBoard.cs

A Program indításkor a „MainMenu”-t nyitja meg. Ez a „Program.cs”-ben van meghatározva.

```
static void Main()
{
    // To customize application configuration such as set high DPI settings or default font,
    // see https://aka.ms/applicationconfiguration.
    ApplicationConfiguration.Initialize();
    Application.Run(new MainMenu());
}
```

3.4.1 MainMenu Form

Amikor be „load”-ol a MainMenu akkor meghívja a „MainMenu_Load(object sender, EventArgs e)” függvényt ami először szinkronizálja az lokálisan tárolt adatokat az adatbázissal (ha sikeres az adatbázis kapcsolat). Majd beolvassa a beállításokat, ha nincsenek mentett beállítások akkor betölti az előre beállított alap tulajdonságokat. Valamint ő tárol minden féle beállítást is amit majd átad más Form-oknak szerkesztésre. Valamit rendelkezek a beállításokhoz tartozó „setter”-ekkel hogy „rá lehessen menteni” a különböző beállításokra a beállítás Form-okból.

```
1 reference
private void MainMenu_Load(object sender, EventArgs e)
{
    DbManager.Instance.SyncDb();

    playerSettings = FileManager.Instance.ReadPlayerSettingsFromFile();
    inputSettings = FileManager.Instance.ReadInputSettingsFromFile();

    // player setting alapbeállítások
    if (playerSettings == null)
    {
        playerSettings = new PlayerSetting[2];
        playerSettings[0] = new PlayerSetting().SetDefaultValues().SetPlayerName("Player1");
        playerSettings[1] = new PlayerSetting().SetDefaultValues().SetPlayerName("Player2");
    }

    // input setting alapbeállítások
    if (inputSettings == null)
    {
        inputSettings = new InputSetting[2];
        inputSettings[0] = new InputSetting(Keys.W, Keys.S, Keys.D);
        inputSettings[1] = new InputSetting(Keys.Up, Keys.Down, Keys.Left);
    }

    playerClone = true;
}
```

A „btn_Start_Click(object sender, EventArgs e)” függvény inicializálja és megnyitja azt létrehozott „Game” Form-ot. a „Game” From konstruktorában meghívja a „GetGameSettings()” függvényt ami alapján fel tudja építeni magát illetve megkapja a „MainMenu”-t ez a vissza lépéshez kell majd. Mert a „MainMenu” csak „láthatatlan” lesz a „Game” Form elindításakor ezzel az a cél hogy a ram-ban benne maradjon és visszalépésnél csak láthatóvá kelljen tenni hogy gyorsan legyen a visszalépés.

```
private void btn_Start_Click(object sender, EventArgs e)
{
    // open game \\
    Game Game = new Game(GetGameSettings());

    Game.Visible = true;
    Game.SetMainMenu(this);

    this.Hide();
    Game.Setup();
}
```

A „GetGameSettings()” függvény egy GameSettings objektummal tér vissza ami tartalmazza a játékra vonatkozó beállításokat. Ha valamelyik beállítás üres vagy nincs beállítva akkor egy előre meghatározott alap beállítást kap.

```
public GameSettings GetGameSettings()
{
    // ha nem valami hiba folytán null lenne valami akkor kap alap értékeket
    if (GetPlayerSettings() == null)
    {
        this.playerSettings = new PlayerSetting[]
        {
            new PlayerSetting().SetDefaultValues().SetPlayerName("Player1"),
            new PlayerSetting().SetDefaultValues().SetPlayerName("Player2")
        };
    }

    if (GetInputSettings() == null)
    {
        Debug.WriteLine("hiba lépett fel a bemeneti billentyűk beálllításánal (null)");
        this.inputSettings = new InputSetting[]
        {
            new InputSetting(Keys.W, Keys.S, Keys.D),
            new InputSetting(Keys.Up, Keys.Down, Keys.Left)
        };
    }

    if (!chb_FullScreen.Checked)
    {
        return new GameSettings(playerSettings, inputSettings, new Size((int)numericUpDown1.Value,
            (int)numericUpDown2.Value), chb_BulletCollision.Checked);
    }
    return new GameSettings(playerSettings, inputSettings, chb_BulletCollision.Checked);
}
```

A „btn_Advanced_Click(object sender, EventArgs e)” függvény inicializálja és megnyitja az „AdvancedSettings” Form-ot és átadja neki a „PlayerSetting” tömböt ami a Játékos beállításokat tárolja, ezt a Form majd felhasználja.

A „btn_InputSettings_Click(object sender, EventArgs e)” függvény inicializálja és megnyitja az „InputSettings” Form-ot és átadja neki a „InputSetting” tömböt ami a Játékos beállításokat tárolja, ezt a Form majd felhasználja.

3.4.2 AdvancedSettings Form

Feladata hogy itt lehessen beállítani a Játékos beállításokat. A konstruktorában kap egy hivatkozást a MainMenu-re (ez majd a mentéshez fog kelleni). Ha a „LoadData(PlayerSetting[]? playerS)” függvény betölti a paraméterben megkapott beállításokat, ha „null”-t kap akkor egy előre beállított alap beállításokat tölt be.

A „btn_AdvsSave_Click(object sender, EventArgs e)” függvény „menti” a beállításokat. Összeszedi a komponensekből az adatokat és inicializál egy „PlayerSetting” objektumot az adatokból, majd „setter”-el átadja a „MainMenu”-nek ezt az objektumot és menti azt fájlba.

A „btn_advsReset_Click” függvény meghívja a „LoadData(PlayerSetting[]? playerS)” függvényt „null” értékkel.

3.4.3 InputSettings Form

Feladata hogy itt lehessen beállítani a Billentyűzet beállításokat. A konstruktorában kap egy hivatkozást a MainMenu-re (ez majd a mentéshez fog kelleni). Ha a „LoadData(InputSetting[]? _inputsettings)” függvény betölti a peraméterben megkapott beállításokat, ha „null”-t kap akkor egy előre beállított alap beállításokat tölt be. Ha a felhasználó rákattint az egyik gombra hogy átállítsa valamelyik billentyű beállítást akkor először a „Set_Input(object sender, EventArgs e)” függvény fut le. Ez nem kiválaszthatóvá teszi a többi gombot (kivéve azt amire rákattintott a felhasználó) és egy „WaitForInput mód”-ba teszi az ablakot, ez azt jelenti hogy a „WaitForInput” privát logikai változó értékét hamisra állítja. Ilyenkor a következő billentyű lenyomás el lesz tárolva kivéve ha „Esc” vagy „Space” Billentyűt nyomja le a játékos (ez nem felel meg a mentésnek ez csak a ram-ba egyenlőre csak ram-ban van tárolva). Ezután meghívódik a „ActivateAllButtons()” ez megint kiválaszthatóvá teszi az összes gombot és meghívódik a „Coloring()” ami színezi a gombokat.

3.4.4 ScoreBoard

Megjeleníti az adatokat az előző játékokról. Ha van adatbázis kapcsolat akkor az adatbázisban lévő adatokat jeleníti meg (megjelenítés előtt feltölti a lokális adatokat). Ha nincs adatbázis kapcsolat akkor csak a lokálisan tárolt adatok jelennek meg. A „ScoreBoard” Form csak egy darab „ListBox”-ot tartalmaz, az oszlopok hatását kóddal éri el. Ez úgy működik hogy megkeresi a leghosszabb elemet egy oszlopban és a többi (rövidebb) elem után a hosszuk különbsége plusz egy szóközt tesz.

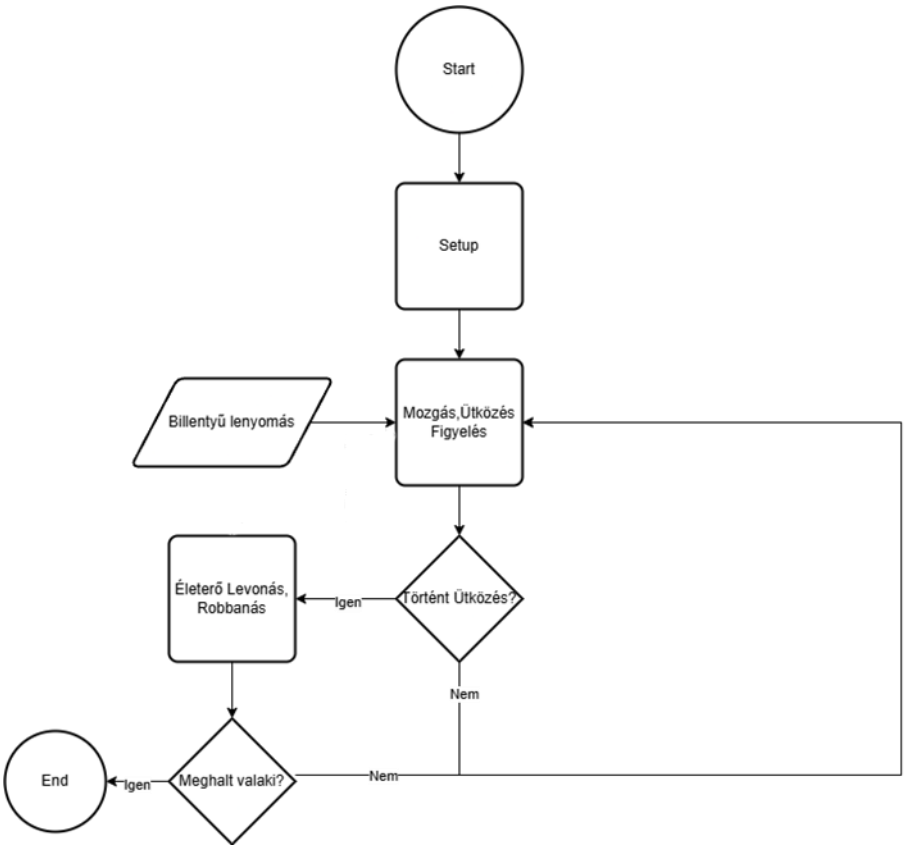
```
for (int i = 0; i < rows.Length; i++)
{
    string[] cols = rows[i].Split(";");
    string rowOutput = string.Empty;
    for (int j = 0; j < cols.Length; j++)
    {
        rowOutput += cols[j];
        int reqSpaces = spaces[j] - cols[j].Length;

        for (int k = 0; k <= reqSpaces + 1; k++)
        {
            rowOutput += " ";
        }
    }
    listBox1.Items.Add(rowOutput);
}
```

3.4.5 Game

Ez a Form amiben a tényleges játék történik, ő végzi el a tipikus „Game Engine”-ekre jellemző feladatokat amikben sokat segítenek az alap Form-os eszközök.

Folyamat Ábra:



A konstruktor paraméterként megkap egy „GameSettings” objektumot ami tartalmaz minden beállítást és információt amire szüksége van a játéknak ha üres a „GameSettings” paraméter akkor egy előre meghatározott alap beállítás töltődik be. Komponensként van az ablakhoz hozzáadva kettő darab Timer, az egyik a játék frissítéséért felel („MainGameTimer”), a másik az eltelt időt méri(„StopWatch”). A Form be „load”-olása után meghívódik a „Setup()” ami felépíti az indításhoz szükséges komponenst és elvégzi a Form-ra vonatkozó beállításokat. A „Setup()” feltölti a „GameComponents” listát ami „GameComponent”-ekből áll, mivel minden osztály ami fizikailag „részt vesz” a játékban az örököl a „GameComponent” osztályból. Így ebben az egy listában könnyen ellehet majd érni az összes komponenst. a komponensek felépítésé után elindítja a két „Timer”-t, ezzel elindul magad a játék.

```
// GameComponents List feltöltése (tárolja az összes GameComponent-et) \\
// index 0: Players
GameComponents.Add(Players);
// index 1: Bullets
GameComponents.Add(new List<GameComponent>());
// index 2: Explosions
GameComponents.Add(new List<GameComponent>());

MainGameTimer.Enabled = true;
StopWatch.Enabled = true;
```

A „MainGameTimer” folyamatosan meghívja egymás után a „MainGame_Update(object sender, EventArgs e)” függvényt ami először megállítja a játékot ha másik ablak van

kiválasztva. A következő ciklus megakadályozza hogy a „Player”-ek kimenjenek a képernyőn kívülre. A következő ciklus meghívja összes komponensnek az „Update()” függvényét (ha kiterjesztik az „IUpdatable” interface-t).

```
/// <summary>
/// Itt történik a frissítés
/// </summary>
1 reference
private void MainGame_Update(object sender, EventArgs e)
{
    // ha másik ablakot nyit meg a felhasználó akkor álljon meg a game
    if (Form.ActiveForm != this && !IsPaused)
    {
        Pause();
    }

    // --- PLAYER --- \\
    for (int i = 0; i < GameComponents[0].Count; i++)
    {
        if (GameComponents[0][i].pictureBox.Location.Y < 0)
            ((Player)GameComponents[0][i]).MoveUp = false;

        if ((GameComponents[0][i].pictureBox.Location.Y +
            GameComponents[0][i].pictureBox.Height*2) > this.Height)
            ((Player)GameComponents[0][i]).MoveDown = false; ;
    }

    // --- MAIN UPDATE --- \\
    for (int i = 0; i < GameComponents.Count; i++)
    {
        for (int j = 0; j < GameComponents[i].Count; j++)
        {
            if (GameComponents[i][j] is IUpdatable)
            {
                ((IUpdatable)GameComponents[i][j]).Update();
            }
        }
    }
```

A következő ciklusban történik mindenféle kezelése a töltényeknek, ezért egy kicsit elméretezett nagy lett a ciklus. A cikluson belül az első dolog az hogy törlődnek a képernyőn kívüli „Bullet”-ek. Ezután egy ciklus végig megy a „Player”-eken és ellenőrzi hogy történt-e találat. Ha történt akkor ellenőrzi hogy meghalt-e valaki a „WinCheck()” függvénnyel.

```
// --- BULLET --- \\
for (int i = 0; i < GameComponents[1].Count; i++)
{
    Bullet bullet = ((Bullet)GameComponents[1][i]);

    // Destruct
    if (!bullet.IsInScreen(Width))
    {
        ((Bullet)GameComponents[1][i]).pictureBox.Dispose();
        GameComponents[1].RemoveAt(i);
        GC_count++;
    }

    // Player hit
    for (int j = 0; j < GameComponents[0].Count; j++)
    {
        if (GameComponents[0][j].pictureBox.Bounds.IntersectsWith(bullet.pictureBox.Bounds) &&
            bullet.PlayerID != GameComponents[0][j].PlayerID)
        {
            ((Player)GameComponents[0][j]).Hit(bullet);

            GameComponents[1][i].pictureBox.Dispose();
            GameComponents[1].RemoveAt(i);
            GC_count++;

            WinCheck();
        }
    }
```

A következő kódrészek csak akkor futnak le ha be van kapcsolva az ütközés a töltények között. A következő kódrész kezeli le azt az esetet hogy ha be van kapcsolva az ütközés a töltények között és ütközik két töltény. Inicializál neki „Explosion”-t és törli a két töltényt ami ütközött

```
// Collision
if (gameSettings.BulletCollision)
{
    for (int j = 0; j < GameComponents[1].Count; j++)
    {
        GameComponent tempBullet = GameComponents[1][j];
        if (bullet.pictureBox.Bounds.IntersectsWith(tempBullet.pictureBox.Bounds) &&
            bullet.PlayerID != tempBullet.PlayerID)
        {
            Explosion explo = new Explosion(-1,
                Create.pictureBox("explo", new Size(40, 40), new Point(0, 0), Properties.Resources.explo),
                bullet, tempBullet, 10);

            GameComponents[2].Add(explo);
            Controls.Add(explo.pictureBox);

            bullet.pictureBox.Dispose();
            tempBullet.pictureBox.Dispose();

            GameComponents[1].Remove(bullet);
            GameComponents[1].Remove(tempBullet);

            GC_count++;
        }
    }
}
```

A következő ciklus törli robbanásokat („Explosion”) miután lejártak.

```
// --- EXPLOSION --- \\
for (int i = 0; i < GameComponents[2].Count; i++)
{
    Explosion explo = (Explosion)GameComponents[2][i];
    if (explo.UpdatesLeft == 0)
    {
        explo.pictureBox.Dispose();
        GameComponents[2].RemoveAt(i);
        GC_count++;
    }
}
```

eddig ebben a függvényben már többször is lehetett látni hogy egy „GC_count” nevű változó értéke inkrementálódik egy törlés után. Ezt arra lesz felhasználva hogy meg legyen hívva a „Garbage Collector” ha ez elhagyható lenne csak akkor kicsit ritkán fut le magától és a program méreteihez képest zavaróan magasra menne a ram használata.

```
// memoria használat csökkentés
if (GC_count > 10)
{
    GC.Collect();
    GC_count = 0;
}
```

A Form ha kap egy billentyű lenyomást akkor meghívja a „Game_KeyDown(object sender KeyEventArgs)” itt van megadva például a játékosoknak hogy merre menjen vagy hogy lőjön. Illetve a kilépés gyorsgombját („Esc”) és a megállítást gyorsgombját („Space”) is itt „veszi be” és kezeli le.

```
/// <summary>
/// Valós időben szedi be az Input-okat a user-től (Down)
/// </summary>
1 reference
private void Game_KeyDown(object sender, KeyEventArgs e)
{
    // back to main menu
    if (e.KeyCode == Keys.Escape)
    {
        Back();
    }
    if (e.KeyCode == Keys.Space)
    {
        Pause();
    }

    if (!LockInputs)
    {
        // left Player (player 1) input (down) management
        if (e.KeyCode == gameSettings.InputSettings[0].UpKey)
            ((Player)GameComponents[0][0]).MoveUp = true;
        else if (e.KeyCode == gameSettings.InputSettings[0].DownKey)
            ((Player)GameComponents[0][0]).MoveDown = true;
        //shoot
        if (e.KeyCode == gameSettings.InputSettings[0].ShootKey)
        {
            Shoot((Player)GameComponents[0][0]);
        }

        // right Player (player 2) input (down) management
        if (e.KeyCode == gameSettings.InputSettings[1].UpKey)
            ((Player)GameComponents[0][1]).MoveUp = true;
        else if (e.KeyCode == gameSettings.InputSettings[1].DownKey)
            ((Player)GameComponents[0][1]).MoveDown = true;
        //shoot
        if (e.KeyCode == gameSettings.InputSettings[1].ShootKey)
        {
            Shoot((Player)GameComponents[0][1]);
        }
    }
}
```

De figyelni kell hogy mikor engedte fel a felhasználó a billentyűt, amikor felenged egy billentyűt akkor meghívódik a „Game_KeyUp(object sender KeyEventArgs)” függvény ami lekezeli hogy ne mozogjon tovább a játékos

```
/// <summary>
/// Valós időben szedi be az "Input"-okat a user-től (Up)
/// </summary>
1 reference
private void Game_KeyUp(object sender, KeyEventArgs e)
{
    // left player (player 1) input (up) management
    if (e.KeyCode == gameSettings.InputSettings[0].UpKey)
        ((Player)GameComponents[0][0]).MoveUp = false;
    else if (e.KeyCode == gameSettings.InputSettings[0].DownKey)
        ((Player)GameComponents[0][0]).MoveDown = false;

    // rigth player (player 2) input (up) management
    if (e.KeyCode == gameSettings.InputSettings[1].UpKey)
        ((Player)GameComponents[0][1]).MoveUp = false;
    else if (e.KeyCode == gameSettings.InputSettings[1].DownKey)
        ((Player)GameComponents[0][1]).MoveDown = false;
}
```

Ennek a rendszernek előnye az hogy a billentyűzettel kapcsolatos dolgok valós időben vannak feldolgozva és frissítés egyenlő időközönként fut le. Azaz a mozgások mindig ugyan olyan gyorsak lesznek nem lesz olyan hogy gyorsabb számítógépen gyorsabb lesz, viszont az a hiba se áll elő hogy valaki két frissítés között nyom le egy billentyűt és ezért nem veszi be a program.

3.5 Interfaces

3.5.1 IUpdatable.cs

Egy „Update()” függvényt tartalmaz ami arra lesz felhasználva hogy könnyű legyen kiválogatni a frissíthető objektumokat.

3.5.2 IHitable

Egy „Hit()” függvényt tartalmaz ami arra lesz felhasználva hogy könnyű legyen kiválogatni az eltalálható objektumokat.

3.6 Settings

3.6.1 GameSettings

Feladata hogy a teljes játékra vonatkozó beállításokat tároljon. Illetve tartalmaz egy publikus „SetDefaultSettings()” függvényt ami egy előre meghatározott alap beállításokat állít be az objektumnak és visszaadja az objektumot.
Tárolja:

- „PlayerSettings” (játékosok beállításai: PlayerSetting[2])
- „InputSettings” (Billentyűzet beállítások: InputSetting[2])
- „WindowSize” (ablak mérete: Size)
- „BulletCollision” (töltény ütközés: bool)

3.6.2 InputSetting

Feladata hogy tárolja egy játékoshoz tartozó billentyűzet beállításokat.

Tárolja:

- „UpKey” (felfelé mozgás: Keys)
- „DownKey” (lefelé mozgás: Keys)
- „ShootKey” (Lövés: Keys)

3.6.3 PlayerSetting

feladata hogy tárolja egy „Player” beállításait. illetve tartalmaz egy publikus „FileFormat()” függvényt ami a fájlba mentéshez megfelelő formátumra formázva adja vissza az adatokat egy string változó formájában, és egy „SetDefaultValues()” függvényt ami beállít egy előre meghatározott alapbeállításokat és visszaadja saját magát.

Tárolja:

- „PlayerName” (játékos neve: string)
- „PlayerSpeed” (játékos sebessége: int)
- „PlayerHP” (játékos életereje: int)
- „BulletSpeed” (töltény sebessége: int)
- „BulletDamage” (töltény sebzése: int)
- „ReloadSpeed” (töltény sebessége: int)
- „Bot” („bot”: bool)

3.7 Utilities

3.7.1 Create

Fő feladata hogy vizuális „Windows Form” komponenseket hoz létere:

- PicutreBox
- ProgressBar
- Label

Ezek publikus statikus függvények hogy a program bármely részén elérhetőek legyenek (a program egy darab szálon fut (nincs multi thread) és ezek a megoldások is 1 szálon futó programoknál használhatóak csak biztonságosan).

Konstruktoraik:

```
/// <summary>
/// Létrehoz egy PictureBox-ot
/// </summary>
/// <param name="name">Neve</param>
/// <param name="size">Méret</param>
/// <param name="location">Hely</param>
/// <param name="image">Kép</param>
4 references
public static PictureBox pictureBox(string name, Size size, Point location, Image image)
```

```
/// <summary>
/// Létrehoz egy ProgressBar-ot
/// </summary>
/// <param name="name">Neve</param>
/// <param name="size">Méret</param>
/// <param name="location">Hely</param>
/// <param name="max">Hány egységre oszlik</param>
1 reference
public static ProgressBar progressBar(string name, Size size, Point location, int max)
```

```
/// <summary>
/// Feliratot/(label)-t hoz létre
/// </summary>
/// <param name="name">Neve</param>
/// <param name="text">Felirat</param>
/// <param name="location">Hely</param>
1 reference
public static Label label(string name, string text, Point location)
```

3.7.2 DbManager

Feladata hogy kezelje az adatbázis-t és az ahhoz tartozó feladatokat (Feltöltés/Letöltés). Ez egy „Singleton Pattern” programozási mintának megfelelő osztály. Először mindenképpen megpróbál adatbázis kapcsolatot (az „Open()” privát függvénnyel) létesít adatbázis kapcsolatot létesíteni. Ha sikeres az adatbázis kapcsolat akkor igaz-ra állítja a „CanConnect” privát logikai változót és bontja a kapcsolatot. Ha az miatt sikertelen a kapcsolat mert nincs ilyen adatbázis akkor létrehozza az adatbázist és a táblát is (!FIGYLEM ha törölni akarjuk az adatbázist akkor ne csak a táblát töröljük mert nincs felkészítve a program arra az esetre ha csak adatbázis van de tábla nincs. Azaz mindig a teljes adatbázist töröljük) Miután létrehozott mindent igazra állítja a „CanConnect” változót. Ha ezek után se sikerült adatbázis kapcsolatot létrehozni akkor hamis lesz a „CanConnect”. Ezt a változót később a többi függvény fogja felhasználni. A többi függvény úgy működik hogy ha a „CanConnect” változó igaz akkor megnyitja az adatbázis kapcsolatot elvégzi a feladatot és lezárja a kapcsolatot. Ha hamis akkor egy alternatív feladatot hajtanak végre a függvények.

Publikus Függvények:

- „Save(string win, string lose, float time)” függvény végzi a mentést. Ha a „CanConnect” változó hamis akkor fájlba ment ahonnan majd a legközelebbi sikeres adatbázis kapcsolatnál feltölti az adatbázisba az adatokat és kiüríti a fájl tartalmát. Viszont ha a kapcsolat sikeres akkor közvetlenül az adatbázisba tölti fel az adatokat. A fájl kezelést a „FileManager” osztály segíti.
- „GetScoreBoard()” függvény egy string tömböt ad vissza az adatokkal és elem egy sort reprezentál „;”-vel elválasztva az adatokat. Ha hamis akkor az ideiglenesen létrehozott fájlból olvassa be az adatokat (a „FileManager” osztály segítségével).
- „SyncDb()” függvény (a „FileManager” osztály segítségével) ha a „CanConnect” változó igaz akkor, beolvassa annak a fájlnak a tartalmát amibe ideiglenesen ment a program ameddig nincs adatbázis kapcsolat. A beolvasott adatokat feltölti majd törli a fájl tartalmát. Ha hamis a „CanConnect” akkor nem csinál semmit.

3.7.3 FileManager

Feladata hogy kezelje a fájlok amik a programhoz kellenek, ezek a fájlok az AppData mappán belül a Roaming mappában vannak, itt hoz létre magának egy mappát a program „Cowboy” néven ahová elhelyezi a fájlokat. Ez egy „Singleton Pattern” programozási mintának megfelelő osztály. Először ellenőri hogy léteznek-e a fájlok, ha nem akkor létrehozza őket három fájlt kell kezelnie ezeknek van három privát string változó ami az elérési útjukat tárolja:

- „inputSettingsPath” (Az Billentyűzet beállításokat tárolja)
- „playerSettingsPath” (A játékosra vonatkozó beállításokat tárolja)
- „gameLogsPath” (ide menti ideiglenesen azokat az adatokat amiket később majd feltölt az adatbázisba a „DbManager”)

Publikus Függvények:

- „SaveToFile(InputSetting[] inputSettings)” függvény végzi az Billentyűzet beállítások mentését
- „SaveToFile(PlayerSetting[] playerSettings)” függvény végzi a Játékos beállítások mentését
- „SaveToFile(string game)” függvény végzi azoknak az adatoknak a mentését amit később majd feltölt a program az adatbázisba sikeres adatbázis kapcsolat esetén. A „string game” bemeneti paraméterben a string formátuma a következő kell legyen „{győzetes neve};{vesztes neve};{játék időtartama}”.
- „ReadGameLogs()” függvény végzi azoknak az adatoknak a beolvasását fájlból amiket az adatbázisba kell feltölteni
- „ReadInputSettingsFromFile()” függvény végzi a Billentyűzet beállítások beolvasását, egy „InputSetting” tömböt ad vissza, a tömbnek egy eleme egy játékoshoz tartozó beállítást tárol.
- „ReadPlayerSettingsFromFile()” függvény végzi a Játékos beállítások beolvasását, egy „PlayerSetting” tömböt ad vissza, a tömbnek egy eleme egy játékoshoz tartozó beállítást tárol.
- „RmGameLogs()” függvény végzi annak a fájlnak a törlését amit az adatbázis adatokkal tölt fel a program ha nincs adatbázis kapcsolat

3.8 Tesztelés

A programot több informatikában nem jártas ember tesztelte, egyelőre hibamentesen. Nem találtak semmilyen hibát. Illetve tesztelve lett több felhasználó által is akik az átlagnál kicsit jobban értenek az informatikához és a játékokhoz, itt se merült fel semmilyen hiba.

3.9 Fejlesztési lehetőségek

A játék vége idő szűkében kicsit összecsapott lett. Lehetne neki csinálni valaki szebb „EndScreen”-t vagy átgondoltabban menteni hogy ki mit ért el és egy szebb eredmény táblát csinálni mondjuk olyanokkal hogy ki hányszor találta el a másikat, vagy hogy a lövések hány százaléka nem talált.

A speciális beállítások ablakon belül lehetne egy olyat hogy csak egy bizonyos mennyiségű „Skill Pont”-ot lehetne „elkölteni” a különböző tulajdonságokra. Valamint érdemes lenne c# „multi thread” rendszerét használni és szétosztani néhány feladatot több szálon hogy jobban fusson a játék. Viszont ebben az esetben kicsit át kell dolgozni a „Singleton” osztályokat mert azok nem „multi thread save”-ek.

Lehetne csinálni hátteret a játékablaknak hogy jobban azt lehessen érezni hogy a vadnyugaton vagyunk. Ez grafikai készségek hiányában elmaradt pedig sokat tudna dobni a játék hangulatán.

3.10 Irodalomjegyzék

A projecthez hozzá lett adva „NuGet Package Manager”-el a „MySqlConnector” külső könyvtár.

- MySqlConnector: „NuGet Package Manager”-el lett hozzáadva a projecthez
- Mysql dokumentáció: https://dev.mysql.com/doc/connector-net/en/
- c# dokumentáció: https://learn.microsoft.com/en-us/dotnet/csharp/
- c# Windows Forms dokumentáció: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-8.0
- Karakter kép: DALL-E (chat gpt)
- Töltény kép: DALL-E (chat gpt)
- icon: Révész Olivér (jóbarát)