

Budapesti Műszaki Szakképzési Centrum
Verebély László Szakgimnáziuma és Szakközépiskolája
54 213 05 Szoftverfejlesztő szakképesítés

TBSGame

Szabó Martin

Tartalomjegyzék

Bevezető.....	1
Elképzelés.....	2
Fejlesztői dokumentáció	4
Alkalmazott fejlesztői eszközök	4
Futtatási környezet	4
Fejlesztői ütemterv	5
Megvalósítás.....	6
Program.cs	6
MainMenu.cs	8
OptionsMenu.cs	10
MapSelector.cs	11
Map.cs	12
UnitInfo.cs	12
TileInfo.cs.....	13
TownInfo.cs	14
TileMap.cs.....	14
Coordinate.cs	15
Tile.cs.....	15
Town.cs.....	15
Unit.cs.....	16
ActionButton.cs.....	16
Player.cs.....	18
GameWindow.cs	18
Editor.cs	20
Felhasználói dokumentáció.....	25
Rendszerkövetelmények	25
Hardver.....	25
Szoftver.....	25
Telepítési útmutató.....	25
Menüpontok.....	25
Főmenü.....	25
Beállításmenü	26
Pályaválasztó menü.....	26
Játéklablak	27
Pályaszerkesztő.....	28

Játékmenet	30
Továbbfejlesztési lehetőségek.....	31
Játék	31
Pályaszerkesztő.....	31
Összegzés	31
Források	31
Ábrajegyzék	32

NYILATKOZAT A SZAKDOLGOZAT EREDETISÉGÉRŐL

Alulírott *Scali Norik* a BMSZC Verebély László
Szakgimnáziuma és Szakközépiskolájának 54 213 05 OKJ Szoftverfejlesztői képzésében részt vevő
hallgatója büntetőjogi felelősségem tudatában nyilatkozom és aláírásommal igazolom, hogy
a *TBS GAME*

.....
című szakdolgozat saját, önálló munkám, és abban betartottam az iskola által előírt, a szakdolgozat
készítésére vonatkozó szabályokat.

Tudomásul veszem, hogy a szakdolgozatban plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás nélkül,
- tartalmi idézet hivatkozás megjelölése nélkül,
- más publikált gondolatainak saját gondolatként való feltüntetése.

E nyilatkozat aláírásával tudomásul veszem továbbá, hogy plágium esetén szakdolgozatom
visszautasításra kerül.

Budapest, 2020. április 10.

Scali Norik
Hallgató aláírása

Bevezető

A játékipar egy rohamosan növekvő, ezzel együtt viszont egy folyton változó, fejlődő dolog. Egy viszonylag új, de rohamosan növekvő dolog a játékiparban például az Android rendszerre készült játékok fejlesztése.

A szakdolgozatom témája is szintén egy játék, de én nem az Android rendszerre írt játékokkal foglalkoztam, hanem a játékok fejlődésével. Hiszem, hogy a játékon belüli pályaszerkesztők a játékmotorok őseinek tekinthetők. Bizonyára mindenki számára is ismert a játékok MOBA¹ műfaja. A MOBA az RTS² alműfaja, mely úgy jött létre, hogy eleinte a Blizzard által fejlesztett Starcraft, később a Warcraft 3 nevű játékok pályaszerkesztőivel létrehozott minigame³ volt. Mára a MOBA műfaj akkorára nőtte ki magát, hogy az egyik legnépszerűbb játék, a Riot Games által fejlesztett League of Legends is ebbe a műfajba tartozik, mely 2009 óta a mai napig fejlesztés alatt áll, és az esport piacon nagy jelentősége van.⁴

A célom az volt, hogy egy játékot biztosítva a lehető legnagyobb szabadságot nyújtsam a mechanika változtatásában a felhasználónak. Véleményem szerint ezek a minigamek növelik a játékok értékét, hiszen szinte végtelen, a felhasználó által generált tartalmat jelentenek. Erre való precedencia, hogy a korább említett Starcraft vagy Warcraft 3 1998-, illetve 2002-ben jelent meg, és még mindig szoros közösség alkotja a játékosbázisukat, ezért az utóbbi időben felújított kiadásuk is megjelent.

¹ online többjátékos csatamező

² valós idejű stratégia

³ játék a játékban - az alap játéktól eltérő működésel

⁴ Forrás: <https://www.polygon.com/2013/9/2/4672920/moba-dota-arts-a-brief-introduction-to-gamings-biggest-most>

Elképzelés

A szakdolgozatom témája egy egyszerű körökre osztott stratégiai játék egy pályaszerkesztővel. A felhasználó a pályaszerkesztővel állíthatja a pálya szerkezetét, a csempék⁵ illetve az egységek kinézetét, számát és erősségét. Az egységeket falvakban lehet toborozni, mely egységenként a pályaszerkesztőben meghatározott körig tart. A játék 2-12 játékosal játszható local multiplayer⁶ elven. A cél hogy az összes többi játékos falvait leromboljuk.

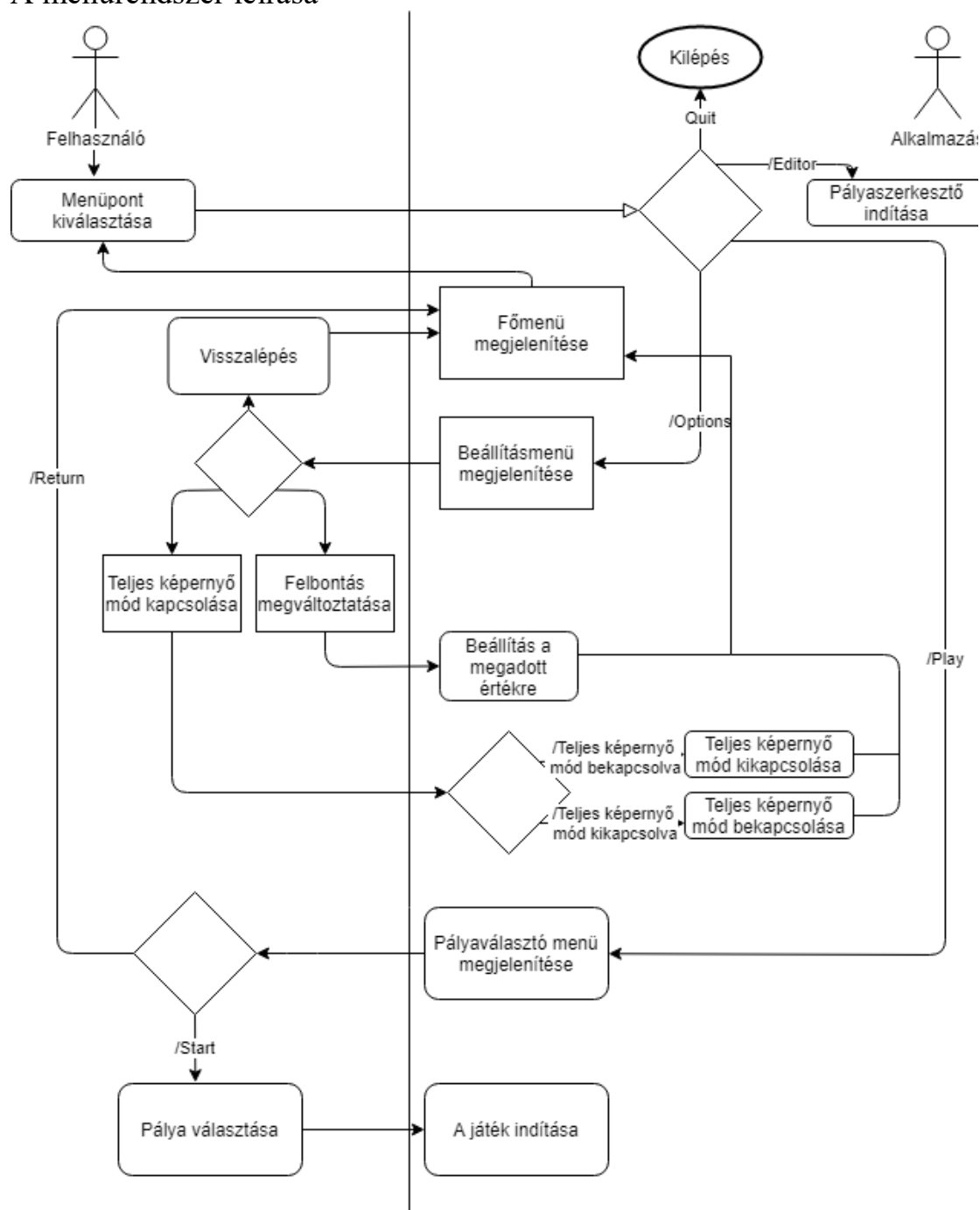
A játékmenet a következőképpen zajlik:

A játékban egy 20x20 csempéből álló pályán minden játékos rendelkezik legalább egy faluval, melyben egységeket toborozhat. A falvaknak a pályaszerkesztőben meghatározott életpontja és körönkénti regenerációja van. Az egységeket, a pályát alkotó csempéket, azok tulajdonságait szintén a pálya készítője határozhat meg. Az egységeknek két fajtája van: harci és gyógyító. Minden egység egy körben csak egyszer támadhat ellenséges egységeket vagy gyógyíthatja a társait. Ezen felül körönként egy egység az akciópontjai szerint meghatározott mértékben mozoghat, melyet a csempe amelyre lép egyedi mértékben csökkent. Az egységre, falvakra, csempékre kattintva a jobb oldali menüben megjelennek azok állandó vagy pontbeli tulajdonságai. A cél, hogy a saját falvainkat védve az összes többi játékos falvát leromboljuk. A győztes játékosnak lehetősége van egyedül folytatni a befejezett játékot, úgynevezett „freeplay” módban.

⁵ A játékban megjelenő mező melyek a pályát alkotják

⁶ egy gépen több játékos

A menürendszer leírása



1. ábra: A menürendszer

Fejlesztői dokumentáció

Alkalmazott fejlesztői eszközök

Használt programok:

- Visual Studio 2017
- Visual Studio 2019
- Notepad++
- Visual Studio Code

Programozási nyelv:

- C#

Külső könyvtárak:

- ini-parser: A .NET keretrendszer nem támogatja az .ini fájl alapú beállításkézelést melyet én preferáltam a .NET saját konfigurációs módszere szerint. Ez a könyvtár beolvassa a beállításokat.

A fejlesztés legelején a legfontosabb lépés a megfelelő technológia és fejlesztői környezet kialakítása. A játékok fejlesztéséhez ajánlott egy játékmotor használata, mely egy játékok fejlesztésére kialakított fejlesztői környezet. Néhány versenyből és kisebb személyes projektekből megtanultam használni az Unity motort, de én úgy gondoltam, hogy a játékmotorok nyújtotta magas szintű utasítások és komponensek használata nélkül nagyobb belátásom lesz egy komplex program működésébe. így többet tanulhatok az objektum orientált programozásról. Ennek viszont több hátránya van, például nincsen multiplatform támogatás, vagy render motor hiányában a program nem néz ki jól.

Fejlesztői környezetként ezért grafikai megjelenítésre a Windows Forms ablakkezelő rendszert használtam, mert ráépül a .NET keretrendszerre és biztosítja azokat a GUI elemeket amelyekre szükségem volt. Azért a C# nyelvet választottam, mert az iskolában is végig azzal programoztunk, így azt a nyelvet sikerült megismernem a legjobban. A .NET keretrendszer legnépszerűbb fejlesztői környezete a Visual Studio, melyet én is használtam, eleinte a 2017-es, majd felfrissítettem a 2019-es verzióra. A térképek szerkezetét illetve egy teszt térképet Notepad ++ használatával alakítottam ki.

Futtatási környezet

A program futtatásához szükséges a .NET keretrendszer 4.5.2 verziója Windows rendszereken. A Windows 10ben beépítve található, de a régebbi Windows verziókra letölthető, egészen a Windows Vista SP2ig.

Telepítésre nincs szükség, elég ha a futtatható fájlt elindítjuk. Fontos viszont, hogy az .exe fájl mellett egy mappában legyen az INIFileParser.dll fájl is. A programnak szüksége van fájlírási

jogosultságra, így vagy rendszergazdai jogosultsággal indítsa el, vagy tegye olyan mappába, ahol nincs szüksége rá. Internetkapcsolatra nincs szükség.

Fejlesztői ütemterv

A projekt fejlesztése két különálló részre bontható. Az első a menürendszer kialakítása, illetve a játék elkészítése. A második a pályaszerkesztő megvalósítása.

Feladat megnevezése	Tervezett határidő	Elkészült
A menürendszer kialakítása		
A menürendszer megtervezése	2019.10.03	2019.10.03
A főmenü kialakítása	2019.10.06	2019.10.06
A beállításokért felelős könyvtár integrációja	2019.10.12	2019.10.13
A beállítások menüjének kialakítása	2019.10.20	2019.10.21
A képernyő felbontás szerinti méretezésének beállításának megvalósítása	2019.10.21	2019.10.21
A játék elkészítése		
A játéklablak megtervezése	2020.02.03	2020.02.04
A játéklablak kialakítása	2020.02.05	2020.02.04
A játéktérületen megjelenő objektumok paneljeinek külön objektumokba szervezése	2020.02.16	2020.02.17
A csempék megjelenítésének megvalósítása	2020.02.18	2020.02.17
A csempék kattintásra az adott csempe adatainak a csempék paneljén történő megvalósítása	2020.02.20	2020.02.19
A falvak kialakítása a csempék gyermekosztályaként, külön esemény átadása	2020.02.24	2020.02.24
Az egységek kialakítása, az egységekre a kattintásra megjelenő panelre az adott egység adatainak kilistázása	2020.02.26	2020.02.27
Az egységek toborzásának illetve a kattintásra megjelenő akciógombok eseményeinek megvalósítása.	2020.03.01	2020.03.01
Az egységek illetve a falvak halálának megvalósítása	2020.03.02	2020.03.01
A körváltás és a játékoskezelés implementálása	2020.03.05	2020.03.05
A győzelem mechanikájának létrehozása, a „freeplay” rendszer kialakítása	2020.03.09	2020.03.11
A pályaszerkesztő megvalósítása		

A pályaszerkesztő megtervezése	2020.03.20	2020.03.18
Az egység réteg kiépítése	2020.03.25	2020.03.26
A csempe réteg kiépítése	2020.03.26	2020.03.27
A térkép fő rétegének kiépítése	2020.03.27	2020.03.28
A térkép mentéséhez kötődő korlátozások implementációja	2020.03.29	2020.03.29
A mentett térkép tesztelése a játékon	2020.03.31	2020.03.31
A tesztelés során felmerülő hibák javítása	2020.04.05	2020.04.05

Megvalósítás

A programkódon belül angol nyelvű kódbeli dokumentáció található mely leírja az alprogramok illetve függvények működését.

Program.cs

Ez a fájl tartalmazza a program belépési pontját, a Main() metódust.

```
static void Main()
{
    Initialize();
    ReadSettings();
    Utils.scale = DetermineSizeScale();

    switch (StartMenu())
    {
        case MainMenuAction.START:
        {
            if (OpenMapSelector())
                PlayGame();
            break;
        }
        case MainMenuAction.EXIT:
        {
            Application.Exit();
            break;
        }
        case MainMenuAction.EDITOR:
        {
            ShowEditor();
            break;
        }
    }
}
```

Az Initialize() alprogram bekapcsolja a formos ablakok vizuális effektjeit (ha azok be vannak kapcsolva operációs rendszeri szinten, illetve kitörli a tartalék fájlokat ha azok a program valamely okból megmaradtak (például a program váratlanul leállt vagy külső hatásra áll le)

```

private static void Initialize()
{
    Application.EnableVisualStyles();

    if (Directory.Exists(Utils.MAP_CACHE)) Directory.Delete(Utils.MAP_CACHE, true);
    Directory.CreateDirectory(Utils.MAP_CACHE);

    if (Directory.Exists(Utils.EDITOR_CACHE)) Directory.Delete(Utils.EDITOR_CACHE,
true);
    Directory.CreateDirectory(Utils.EDITOR_CACHE);

    if (!Directory.Exists(Utils.MAP_FOLDER)) Directory.CreateDirectory(Utils.MAP_FOLDER);
}

```

Szerepel a fenti kódrészletben hivatkozás az Utils osztályra.

Ez az osztály központi szerepet foglal a programban, mert globális konstans változókat tárol, illetve a beállítások olvasása után ebben található a skálázási arány, mely megmutatja mindennek, hogy hányszorosára kell növekedniük hogy alkalmazkodjanak az új felbontáshoz.

```

public const string MAP_CACHE = @"MapCache\",
SETTINGS_FILE = "AppSettings.ini",
MAP_FOLDER = @"Maps\",
EDITOR_CACHE = @"Editor\",
EDITOR_UNIT_IMAGE_POSTFIX = "@.png",
EDITOR_TILE_IMAGE_POSTFIX = "$.png";

public const int BASE_WIDTH = 608,
BASE_HEIGHT = 342,
BASE_TILE_HEIGHT = 15,
BASE_TILE_WIDTH = 15;

public static IniData settings;

public static SizeF scale;

```

A konstans változók:

- MAP_CACHE: A térképeknek kicsomagolása után a fájlokat eltároló mappa relatív útvonala.
- SETTINGS_FILE: A program beállításait tartalmazó fájl neve.
- MAP_FOLDER: A térképek számára fenntartott mappa. A program csak akkor látja őket, ha ebben a mappában találhatóak.
- EDITOR_CACHE: A pályaszerkesztő a térkép készítése során a térkép fájljait ideiglenesen tároló mappa.
- EDITOR_UNIT_IMAGE_POSTFIX / EDITOR_TILE_IMAGE_POSTFIX: Mivel az egységekhez illetve a csempékhez definiált kép neve a térképfájlban a név alapján van meghatározva, ezért hogy lehetőség legyen definiálni ugyan azon a néven egy egységet és egy csempét, az egységek után egy @, a csempék után egy \$ van hozzátéve.
- BASE_WIDTH / BASE_HEIGHT: az ablakok alap mérete.
- BASE_TILE_HEIGHT / BASE_TILE_WIDTH: Egy csempe alap méretei.

A ReadSettings() metódus beolvassa a beállításokat.

A beállításfájl hiánya vagy egy hibás érték esetén létrehozza a fájlt az alap beállításokkal.

A fájl szerkezete a szabványos .ini formátumnak megfelelő: szekciókon belüli kulcs-érték párokból áll.

[UI]

```
fullscreen = false  
res-x = 608  
res-y = 342  
aspectRatioWarning = true
```

A beállítások magyarázata:

- fullscreen: A teljes képernyő mód értéke: igaz/hamis
alapérték: hamis
- res-x / res-y: az ablak méretei: egész szám
alapérték: 608, 304.
- aspectRatioWarning: A program bár támogat 16:9(illetve 683x384)-tól eltérő arányú felbontást, csak az előbbi arány(ok)ra volt tervezve, így a teljes élmény érdekében figyelmezteti a felhasználót ha ettől eltérő van beállítva: igaz/hamis
alapérték: igaz
Mivel a támogatott képaránytól eltérő beállítás csak a beállításfájl kézi átírásával történhet meg (kivéve ha más képaránnyal teljes képernyő módban használjuk a programot) ennek a beállításnak a megváltoztatása is csak így érhető el.

A beállítások után a program a belső képarányszorzót határozza meg, ennek képlete:
új felbontás hossz/alap felbontás hossz;új felbontás magasság/alap felbontás magasság

MainMenu.cs

A StartMenu() függvény visszatérési értéke egy érték a MainMenuAction enumerációból mely a MainMenu osztály példányának egy tulajdonsága

```
public MainMenuAction Action  
{  
    get { return _action; }  
    private set  
    {  
        DialogResult = DialogResult.OK;  
        _action = value;  
        Close();  
    }  
}  
...  
private void Exit(object sender, EventArgs e)  
{  
    Action = MainMenuAction.EXIT;  
}  
private void Options(object sender, EventArgs e)  
{  
    var opts = new OptionsMenu();  
    opts.ShowDialog();  
    opts.Dispose();  
}  
private void Editor(object sender, EventArgs e)  
{  
    Action = MainMenuAction.EDITOR;  
}  
private void Play(object sender, EventArgs e)  
{  
    Action = MainMenuAction.START;  
}
```

A MainMenuAction érték függ a lenyomott gombtól, mely bezárja az ablakot és az érték szerint a következő ablakot nyitja vagy bezárja a programot, kivéve a beállításmenüt. A beállításmenü egy új ablakként jelenik meg kisebb mértékben a képernyő közepén, és amíg aktív a fókuszot nem engedi át a főmenü ablakára.

Az összes ablaknál el kellett érni, hogy az alap formos ablakszél hiányában is lehessen az ablakot mozgatni. Ezért a következő kódrészlet felel:

```
private bool mouseDown;
private Point lastLocation;
private void TopField_MouseDown(object sender, MouseEventArgs e)
{
    mouseDown = true;
    lastLocation = e.Location;
}
private void TopField_MouseMove(object sender, MouseEventArgs e)
{
    if (mouseDown && WindowState != FormWindowState.Maximized)
    {
        Location = new Point(
            (Location.X - lastLocation.X) + e.X,
            (Location.Y - lastLocation.Y) + e.Y);

        Update();
    }
}
private void TopField_MouseUp(object sender, MouseEventArgs e)
{
    mouseDown = false;
}
```

Ahol:

- topField egy panel valahol az ablak tetején
- A metódusok a panel adott eseményére hivatkoznak

ScaleFontSize()

A formok Scale(SizeF) metódusa nem tartalmazza a betűtípusok skálázását, így minden formban ez az alprogram megteszi.

A pályaszerkesztő kivételével ez mindenhol egyesével történik. Azért így, mert a szöveget megjelenítő controlok mind statikusak, ezért nem láttam szükségesnek, viszont a pályaszerkesztőben a controlok száma miatt iterál, kivéve két controlt, mely valamiért nem méreteződik át az iterációban:

```
private void ScaleFontSize(SizeF scale)
{
    lbTitle.Font = new Font(lbTitle.Font.FontFamily, lbTitle.Font.Size * scale.Height);
    unitDescription.Font = new Font(unitDescription.Font.FontFamily,
                                    unitDescription.Font.Size * scale.Height);

    foreach (Control control in Controls)
        control.Font = new Font(control.Font.FontFamily, control.Font.Size * scale.Height);
}
```

A beállításmenüben lehet váltani a teljes képernyő módra vagy egy új felbontásra.

Az állítható felbontások mennyisége, illetve a mérete függ az elsődleges kijelző felbontásától:

```
private void FillCb()
{
    byte scaleDouble = 2;

    while (Fits(scaleDouble))
    {
        cbResolution.Items.Add($"{Utils.BASE_WIDTH * scaleDouble / 2.0f}x" +
                                $"{Utils.BASE_HEIGHT * scaleDouble / 2.0f}");
        scaleDouble++;
    }

    Highlight();
}
```

A választható felbontások az alap felbontás felét adják hozzá magukhoz amíg beleférne a képernyőbe.

A Fits() függvény egy logikai értékkel tér vissza, mely megmutatja, hogy a jelenlegi érték beleférne-e az elsődleges képernyőbe.

```
private bool Fits(byte scaleDouble)
{
    return !(Utils.BASE_HEIGHT * scaleDouble / 2.0f > Screen.PrimaryScreen.Bounds.Height ||
            Utils.BASE_WIDTH * scaleDouble / 2.0f > Screen.PrimaryScreen.Bounds.Width);
}
```

A Highlight() metódus kiválasztja a ComboBoxban a jelenleg használt felbontást.

```
private void Highlight()
{
    bool found = false;
    int resX = int.Parse(Utils.settings["UI"]["res-x"]);
    int resY = int.Parse(Utils.settings["UI"]["res-y"]);
    foreach (var res in cbResolution.Items)
    {
        var tmp = res.ToString().Split('x');
        if (resX.ToString() == tmp[0] &&
            resY.ToString() == tmp[1])
        {
            found = true;
            cbResolution.SelectedItem = res;
        }
    }

    if (!found)
    {
        cbResolution.Items.Add($"Custom ({resX}x{resY})");
        cbResolution.SelectedIndex = cbResolution.Items.Count - 1;
    }
}
```

Amennyiben a jelenleg használt felbontás nincs az előbbi módon hozzáadott felbontások listájában (mert a beállításfájl át lett írva), a program magától hozzáadja Custom néven, mellette jelzi a beállított értéket is zárójelben.

A mentés után a program valójában nem visszatér a főmenübe, hanem újraindul, és az új beállítások szerint töltődik be. Ez minden más főmenübe való visszatéréskor így történik.

```
Application.Restart();
```

A beállításmenünél hozzátettem, hogy az ablakot ne lehessen mozgatni, ezt úgy értem el, hogy a form WndProc alprogramát felülírtam, hogy kiszűrje a mozgásért felelős paramétert, és akkor ne csináljon semmit

```
protected override void WndProc(ref Message m)
{
    const int WM_SYSCOMMAND = 0x0112;
    const int SC_MOVE = 0xF010;

    if (m.Msg == WM_SYSCOMMAND)
    {
        if ((m.WParam.ToInt32() & 0xffff) == SC_MOVE)
            return;
    }
    base.WndProc(ref m);
}
```

MapSelector.cs

```
private static bool OpenMapSelector()
{
    var mapSelector = new MapSelector(useFullScreen);

    bool result = mapSelector.ShowDialog() == DialogResult.OK;

    if (result)
    {
        string mapFile = mapSelector.SelectedMapFileName;
        mapSelector.Dispose();
        ZipFile.ExtractToDirectory(Path.Combine(Utils.MAP_FOLDER, mapFile),
        Utils.MAP_CACHE);
    }

    return result;
}
```

A pályák kiválasztása egy DataGridView sorának kiválasztásával történik.

A pályák egy .zip kiterjesztésű tömörített állományból állnak, melyek tartalmazzák a következőket:

- map.txt: 20x20 karakter mely leírja a pálya szerkezetét.
- units.txt: leírja az összes egység adatait.
- tiles.txt: leírja az összes csempe adatait és egy karaktert mely megegyezik a map.txtben megegyező karakterek egyikével.
- + a csempék és az egységek képei.

A térképek fájlnevének a szerkezete a következő:

DataGridView 1. oszlopa			DataGridView 2. oszlopa			DataGridView 3. oszlopa
[Játékosok száma]	Szóköz	Térkép neve	-	Térkép készítőjének neve
()				
{		}				

Amennyiben a térkép neve nem egyezik meg ennek a formátumnak vagy rosszul jelenik meg vagy az

egész fájlnev a második oszlopban jelenik meg, ez függ a térkép fájlnevétől. Hibát nem dob a program.

A kiválasztott pálya kibontásra kerül a MapCache mappában.

Az OpenMapSelector logikai visszatérési értéke azt határozza meg, hogy történt-e kibontás. Amennyiben úgy zárul be az ablak hogy nem került kibontásra térkép fájl, a program bezárul.

Map.cs

A térképobjektum magában foglal mindent, amit a térkép definiál. Ezen felül nem csak a játéklablakban jelenik meg, hanem a térkép mentésekor is a pályaszerkesztőben.

```
public Map()
{
    GetTileInfos();
    GetUnitInfos();
    GetTileMap();
    GetPlayerCount();
}
public Map(List<UnitInfo> units, List<TileInfo> tiles, char[,] tileMap,
           byte playerCount, string mapName, string mapCreator)
{
    Tiles = tiles;
    Units = units;
    tileChars = tileMap;
    PlayerCount = playerCount;
    this.mapName = mapName;
    this.mapCreator = mapCreator;
}
```

Az objektumnak két konstruktora van. A paraméterek nélküli meghívásnál a térképobjektum az adatait a kicsomagolt fájlokat eltároló mappában (MapCache) lévő fájlokból olvassa. Ez a játék kezdetekor van. A paraméterezett konstruktor átadja az objektumnak a térkép összes adatát, ezt a pályaszerkesztőben mentésnél teszi. A pálya mentését, illetve annak tömörítését a SaveMap() alprogram vezényeli.

```
public void SaveMap()
{
    SavePlayerCount();
    SaveTileInfos();
    SaveUnitInfos();
    SaveTileMap();

    ZipFile.CreateFromDirectory(Utils.EDITOR_CACHE,
                               Utils.MAP_FOLDER + "${PlayerCount}"+
    {mapName}-{mapCreator}.zip");
    Directory.Delete(Utils.EDITOR_CACHE, true);
}
```

UnitInfo.cs

Ez egy struktúra, mely eltárolja az egységek összes adatát. Szintén két konstruktort tartalmaz, az elsőnél csak egy karakterlánc a paraméter, a másodiknál az összes tulajdonság külön paraméterként megtalálható. A térképen belüli objektumoknak is szintén betöltéskor és mentéskor használatos a két külön konstruktor.

Az egységek fájlírása csv⁷ formátumban történik. A következő adatokat hordozza:

- | | |
|---|---------------------|
| • Az egység neve | karakterlánc |
| • Az egység képzéséhez szükséges idő körökben | 32 bites egész szám |
| • Az egység maximális élete | 32 bites egész szám |
| • Az egység minimális sebzése | 32 bites egész szám |
| • Az egység maximális sebzése | 32 bites egész szám |
| • Az egység akciópontjai | 32 bites egész szám |
| • Az egység típusa | 8 bites egész szám |
| • Az egység textúrája | kép |
| • Az egység szöveges leírása | karakterlánc |

A C# véletlenszám-generálás függvénye nem tartalmazza a felső értéket (tehát a második paraméter mínusz 1 a felső határérték). E miatt viszont be kellett vezetni, hogy az egység GenerateDamage() függvényébe ne a minimum és a maximum sebzés között generáljon, hanem a minimum és a maximum + 1 között. Ezért mivel a játék engedi hogy a két érték egyenlő legyen (ha nem akarjuk, hogy a sebzése egy véletlen határon múljon) a pályaszerkesztő nem a 32 bites előjeles egész szám szerinti 2,147,483,647, hanem ennél egyel kevesebb korlátot állít be a két értéknek, ugyanis ha az előbbi értéket adjuk meg a maximális sebzésnek, a sebzés generálásakor hozzáad egyet, és a 32 bites egész szám integer overflow miatt -2,147,483,648-at fog megadni a sebzés felső határértékének és a program az egység támadásakor hibát dob. Ez a hiba kezelve van a pályaszerkesztőben.

Tartalmazza a ToCSV() függvényt, mely mentéskor átalakítja a csv formátumba az egységek adatait:

```
public string ToCSV()
{
    string filePath = Name + Utils.EDITOR_UNIT_IMAGE_POSTFIX;

    return $"{Name};{RecruitTime};{Health};{MinDamage};{MaxDamage};{ActionPoints};" +
           $"{Type};{filePath};{Description}";
}
```

Fontos még, hogy a szöveges fájlban a textúrát nem szöveges formátumra (base64) alakítja át, hanem átmásolja a kiválasztott képet, és annak a relatív útvonalát. Mivel a kép elnevezése az egység illetve a csempe nevétől függ, többnyire e miatt nem enged a pályaszerkesztő többször egy néven egy egységet vagy csempét definiálni.

Bár 256 féle lehetőség van az egység típusára, a játékban jelenleg 2 fajta támogatott:

- 0=harcos
- 1=gyógyító

A pályaszerkesztőben ez logikai értéként van megadva, így ez is kezelt.

[TileInfo.cs](#)

A csempék adatait eltároló osztály. Hasonlóan az egységek adataival, ez is két konstruktorral rendelkezik, tartalmazza a ToCSV() függvényt és eltárolja a csempék összes adatát.

A következő adatokat tárolja:

- | | |
|--------------------------------|----------|
| • A csempéhez tartozó karakter | karakter |
|--------------------------------|----------|

⁷ pontosvesszővel elválasztott értékek egy szöveges fájlban

- A csempére lépéskor levont akciópontok 32 bites egész
- A csempén álláskor kapott páncélbónusz tört szám (double)
- A csempe neve karakterlánc
- A csempéhez tartozó textúra kép

Ennél az osztálynál a legnagyobb probléma az volt, hogy míg a magyar szabványban a tizedestörteket egy vesszővel jelezzük, addig máshol ezt egy ponttal teszik, és a .NET keretrendszer ezt átalakításakor figyelembe veszi, így a készített pálya egy olyan nyelvű Windows környezetben, ahol az adott országban a másik szabvány van használatban hibát dob. Ezt a `CultureInfo.InvariantCulture`, illetve a `NumberStyles.Any` paraméterekkel kezeltem:

```
double.TryParse(tmp[2], NumberStyles.Any, CultureInfo.InvariantCulture, out double tmpArmorBonus)
```

Fontos még megjegyezni, hogy ebben az osztályban a `ToCSV()` függvény virtuális, mert a falvakat leíró osztály ennek az osztálynak a gyermekosztálya, és az felülírja.

`TownInfo.cs`

A csempék adatainak gyermekosztálya. Tartalmaz mindent, amit a csempék plusz a következőket:

- A falvat birtokló játékos indexe 8 bites egész szám
- A falu maximális élete 32 bites egész szám
- A falu körönkénti regenerációja 32 bites egész szám

```
public override string ToCSV()
=> base.ToCSV() + $"{OwnerPlayer};{MaxHealth};{Regeneration}";
```

A `ToCSV()` függvény kiegészíti az ősoosztályét a saját adatainak hozzáadásával.

Úgy dönti el, hogy mely osztály adataként vegye fel a listába az adatot, hogy a csv formátumból tömbbé alakítás során hány adatot tartalmaznak:

A két lehetőség (5 és 8) közül a feltételben az 5 vizsgált.

```
if (dummy.Split(';').Length == 5) Tiles.Add(new TileInfo(dummy));
else Tiles.Add(new TownInfo(dummy));
```

A falvak csempéként való kezelése miatt a térkép fájlban minden játékos faluja külön van definiálva.

`TileMap.cs`

Ez tartalmazza az egész térkép csempe rétegét.

Konstruktorában értéként megkapja a csempék adatainak listáját és a kiválasztott térkép `map.txt` fájljának tartalmát. A `map.txt` állomány tartalma egy 20x20 méretű karakterekből álló mátrix, ahol a karakterek mindegyike megyegyezik a csempék adatainál definiált karakterek valamelyikével. Az osztály tartalma egy ebből példányosított 20x20-as mátrix, ahol a példányosított csempék vagy falvak találhatóak. A falvak példányosított változata ugyan úgy a csempék osztályának gyermekosztálya.

Coordinate.cs

Egy struktúra, mely magában foglal két számot 1-20 közötti nyílt intervallumban. A két szám határozza meg az egyenlő pontot az egységek és a csempék közt. Nem egyenlő a csempék a mátrixban foglalt indexével, hanem mindkét számnál egyel több.

Az egyszerűbb vizsgálatért az Equals() függvény használata helyett az == operátor van használatban, melyet a következőképpen definiáltam:

```
public static bool operator ==(Coordinate a, Coordinate b)
    => (a.X == b.X) && (a.Y == b.Y);
public static bool operator !=(Coordinate a, Coordinate b)
    => !(a == b);
```

Továbbá van egy IsBetween függvény is, mely megmutatja, hogy egy koordináta beleillik-e két koordináta közé:

```
public bool IsBetween(Coordinate cornerA, Coordinate cornerB)
{
    return X >= Math.Min(cornerA.X, cornerB.X) && X <= Math.Max(cornerA.X, cornerB.X) &&
        Y >= Math.Min(cornerA.Y, cornerB.Y) && Y <= Math.Max(cornerA.Y, cornerB.Y);
}
```

Minden játékmezőn megjelenő objektumnak van egy koordinátája:

- Tile
- Town
- Unit
- ActionButton

Tile.cs

A csempe példánya a definiált karakteren kívül minden adatot magában foglal, plusz a csempe koordinátáit. Már a Control osztályból van származtatva, mely megjelenik a formon.

A controlok mérete, illetve helye csak egész számokban határozható meg, így a kerekítésnél előfordulhat, hogy a játékterület méreténél nagyobb lesz a csempék összes hossza. Ezt hogy elkerüljem a csempék mérete mindenképpen lefele van kerekítve, és a játéklak mérete is ahhoz van viszonyítva:

```
gameArea.Size = new Size(map.TileMap.Tiles[0, 0].Width * 20,
                        map.TileMap.Tiles[0, 0].Height * 20);

gameArea.Location = new Point((gameAreaBorder.Height - gameArea.Size.Height) / 2,
                             (gameAreaBorder.Width - gameArea.Size.Width) / 2);
```

Ez a módszer a pályaszerkesztőben is használva van, a térkép réteg csempe területén.

Town.cs

A falvak a formon megjelenő objektumai. Tartalmaz mindent, amit egy falu definiált adata, és még:

- A falu jelenlegi életpontjai: 32 bites egész szám
- A falut birtokló játékos indexe: 8 bites egész szám

- A falu életben van-e: logikai
- A falu jelenleg képez-e egységeket: logikai
- Ha képez, ez mennyi körig tart még: 32 bites egész szám
- Ha képez, milyen egységet: egység
- ElapsedTurn(): bool: A falut regenerálja a körönkénti regenerációs értékkel, és levon egy kört a hátralévő körök számából. Visszatér igazat, ha az egység már kész.
- Recruit(UnitInfo,byte): void: Megkezd egy egység kiképzését.
- +Damage(int): bool: A paraméterben megadott sebzést a falunak adja. Visszatér igazat ha az falu le lett rombolva

Unit.cs

Az egység a formon megjelenő objektuma. A tulajdonságai részhalmaza egyenlő az UnitInfoval és még néhány extra:

- A Coords tulajdonság setterében a mozgásnál az egység a játékterületen belüli helyét is igazítja:

```
public Coordinate Coords
{
    get { return _coords; }
    set
    {
        Location = new Point((int)((Utils.BASE_TILE_WIDTH - BASE_WIDTH) *
(Utils.scale.Width / 2)) +
(int)(Utils.BASE_TILE_WIDTH * Utils.scale.Width) *
(value.X - 1),
        Height * (value.Y - 1));

        coords = value;
    }
}
```

- Az egységet birtokló játékos indexe: 8 bites egész szám
- Az egység támadott-e az adott körben: logikai
- A jelenlegi életpontok száma: 32 bites egész szám
- A jelenlegi akciópontok száma: 32 bites egész szám
- +Damage(int): bool: A paraméterben megadott sebzést az egységnek adja. Visszatér igazat ha az egység meghalt
- +Heal(int): void: A paraméterben megadott számot gyógyít az egységen
- +ElapsedTurn(): void: Az egységek akciópontjait, illetve hogy támadott-e az adott körben az alapértékre állítja. Minden kör végén meghívódik minden egységnek

ActionButton.cs

Az egységek körönkénti támadását, illetve mozgásukat meg kellett oldani.

Ez a control egy egyedi gombként funkcionál, mely kisebb egy egységnél. Az egység kiválasztásakor az egység körül négy irányban megjelenik, majd felvesz egy szint az adott akcióra, attól függően, hogy milyen egység található az adott koordinátán

- zöld: szabad csempe, mozgás.

- citromsárga: barátságos egység, a gyógyítók gyógyíthatják.
- narancssárga: barátságos falu, nem enged visszalépni.
- piros: az ellenség falva, egységei.

Ez a TileStatus enumeráció, melynek egy értéke az osztály egyik tulajdonsága.

Az akció a gombra való kattintáskor hajtódik végre. Fontos megjegyezni, hogy az egységeket előbb támadja, tehát ha egy ellenséges egység található egy faluban, először azt támadja, majd csak utána támadja a falvakat.

A négy akciógomb a játéklapokban egy tömbben van tárolva, a négy iránya az akciógombnak függ a tömbön belüli indexétől:

Az irányt a CoordinateDirection enumeráció adja meg, melyet a konstruktorban kap meg, és egyenlő az objektum tömbbeli indexével:

```
enum CoordinateDirection
{
    UP = 0,
    RIGHT = 1,
    DOWN = 2,
    LEFT = 3
}
```

A kapott érték szerint veszi majd körbe a kiválasztott egységet a CenterAroundUnit() alprogrammal, mely körbeveszi az adott egységet, és a koordinátáit beállítja a négy irányban az egység köré:

```
public void CenterAroundUnit(Coordinate unitLocation)
{
    switch (direction)
    {
        case CoordinateDirection.UP:
        {
            Coords = new Coordinate(unitLocation.X, (byte)(unitLocation.Y - 1));
            break;
        }
        case CoordinateDirection.RIGHT:
        {
            Coords = new Coordinate((byte)(unitLocation.X + 1), unitLocation.Y);
            break;
        }
        case CoordinateDirection.DOWN:
        {
            Coords = new Coordinate(unitLocation.X, (byte)(unitLocation.Y + 1));
            break;
        }
        case CoordinateDirection.LEFT:
        {
            Coords = new Coordinate((byte)(unitLocation.X - 1), unitLocation.Y);
            break;
        }
    }
    Visible = true;
}
```

Player.cs

A játékost kifejező objektum. Tartalmaz egy logikai változót, mely meghatározza, hogy a játékos még játékban van-e, illetve a játékos által birtokolt egységek listáját.

```
public List<Unit> OwnedUnits { get; set; }

public bool InGame { get; set; }
```

GameWindow.cs

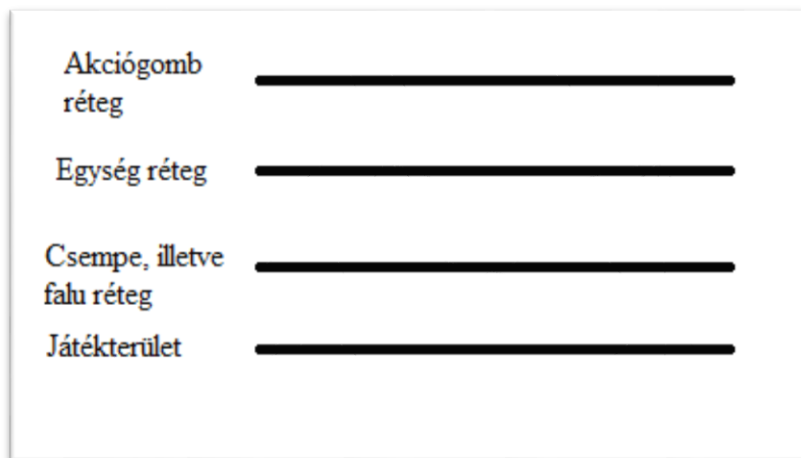
A játék tevékenységét lebonyolító formablak. Tartalmazza:

- Négy akciógombot egy tömbben
- A játék által használt térkép objektumát
- A kattintásra felmerülő menüket
- A játékosokat
- A jelenlegi játékos indexét
- A kiválasztott egységet
- A kiválasztott falvat

Az ablak konstruktorában beolvassa a térképobjektumot a kicsomagolt fájlokból. Amennyiben bármilyen kezelt vagy kezeletlen hiba felmerül a térképfájlban, egy MessageBoxon keresztül jelzi a felhasználónak a hiba üzenetével együtt, majd visszatér a főmenübe.

A konstruktorban kerül sorra a TileInfoPanel, TownPanel, UnitInfoPanel mezők hozzáadására is. Ezek származtatott panelek, melyeket a játéklak nagy mennyiségű objektumainak csökkentésére hoztam létre. A TownPanel magában foglal még egy

TownInfoPanelt és egy RecruitInfoPanelt. Ezeknek saját módszereik vannak, melyek kezelik a kattintott egységek adatainak hozzáadását, illetve egy interfészt nyújtanak a felhasználónak az egységek kiképzésére és az adataik megjelenítésére. A jobb oldali menüben találhatóak.



2. ábra: A játékterület rétegei

Az Unityban használatos rétegelvet nem támogatja a Windows Forms felülete a

lehető legtisztább módon, így hogy egy panelre kerüljen mind a három fajta objektum (falvak/csempék, egységek, akciógombok) a BringToFront() alprogram lefuttatásával előre kell őket kényszeríteni. Szerencsére a rétegezés az objektumok hozzáadásának sorrendje szerint történik, így a csempéknél ez nem akadály. Azonban az egységek eltakarhatják az akciógombokat, ezért az egység kattintásakor előre kell őket küldeni.

```
for (int direction = 0; direction < actionButtons.Length; direction++)
{
    ...
    actionButtons[direction].BringToFront();
    ...
}
```

Az egységek váltása mellett kiemelkedően fontos volt a jobb oldalon lévő menük, illetve az akciógombok kikapcsolása is, mely elrejti őket, ha valamely más dolgot választunk ki. A ClearMenuView metódus feladata ez, majd az adott objektumot kezelő alprogram majd ismét bekapcsolja az adott panelt, illetve az akciógombokat:

```
private void ClearMenuView()
{
    tileInfo.Visible = false;
    unitInfo.Visible = false;
    townArea.Visible = false;

    for (int direction = 0; direction < actionButtons.Length; direction++)
        actionButtons[direction].Visible = false;
}
```

Körökre osztás

A körök váltásánál figyelembe kellett venni már kiesett játékosok kihagyását, illetve azt, hogy kettőnél több játékos szerepel, ezért ciklusonként csak egyszer bonyolítsuk le a körönkénti eseményeket. Továbbá fontos volt, hogy tudja kezelni a körváltást az egy játékosbeli „freeplay” módhoz is.

A körönkénti események kezelésére szolgál a NewTurn() metódus, mely kezeli az egységek kiképzését, a falvak regenerációját, illetve az egységek akciópontjának újratöltését.

```
private void NewTurn()
{
    currentPlayer = 0;

    for (byte line = 0; line < map.TileMap.Tiles.GetLength(0); line++)
        for (byte column = 0; column < map.TileMap.Tiles.GetLength(1); column++)
            if (map.TileMap.Tiles[line, column] is Town &&
                (map.TileMap.Tiles[line, column] as Town).ElapsedTurn())
            {
                var newUnit = (map.TileMap.Tiles[line, column] as Town).recruitingUnit;
                newUnit.Click += new EventHandler(ClickUnit);

                (map.TileMap.Tiles[line, column] as Town).recruitingUnit = null;

                players[(map.TileMap.Tiles[line, column] as Town).ownerPlayer]
                    .OwnedUnits
                    .Add(newUnit);

                gameArea.Controls.Add(newUnit);
            }

    foreach (var p in players)
        foreach (var u in p.OwnedUnits)
            u.ElapsedTurn();
}
```

Ez akkor hívódik meg, ha a „New Turn” gombra kattintva a játékos a következő játékosra vált, és már véget ért az összes játékoson a kör. A következő játékosot kezelő alprogramnál a kiesett játékosok kihagyására rekurziót kellett használnom, mert a kattintást kezelő esemény miatt nem váltható át ciklussá:

A jobb láthatóság érdekében a kör váltásakor mindig beállítjuk a jelenlegi játékos által birtokolt egységek hátterét zöldre, az ellenségeseket pedig pirosra. Ez azért is jó, mert így le van kezelve az,

hogy ha a pályaszerkesztőben egy teljesen átlátszó képet adunk meg textúraként akkor ne egy láthatatlan egységünk legyen.

```
private void NextPlayer(object sender, EventArgs s)
{
    ClearMenuView();
    currentPlayer++;

    if (currentPlayer == players.Length) NewTurn();

    if (!players[currentPlayer].InGame)
    {
        NextPlayer(sender, s);
        return;
    }

    foreach (var p in players)
        foreach (var u in p.OwnedUnits)
        {
            u.BackColor = (u.ownerPlayer == currentPlayer)
                ? Color.Green
                : Color.Red;
        }
    PlayerNameDisplay.Text = "Player " + (currentPlayer + 1);
}
```

Az események rögzítését a képernyő jobb oldalán található RichTextBox végzi. Ezt a Log() alprogram végzi

```
private void Log(string text)
{
    logger.Text += Environment.NewLine + text;
}
```

A TBSGame.Misc névtérben találhatóak származtatott UI elemek, melyeket a kód mennyiségének csökkentésére hoztam létre. Célja, az Unityban megszokott „prefab”ok helyettesítése, továbbá a DisplayLabel tartalmaz egy statikus függvényt, mely segíti a betűtípus méretének skálázását:

```
public static Font ScaledFont(float scale)
=> new Font("Microsoft Sans Serif", 8.25F * scale);
```

Editor.cs

A pályaszerkesztő célja, hogy a felhasználó tudjon pályákat készíteni. Az ablak három, a kód négy logikai részre van osztva:

- Egység
- Térkép
- Csempe
- A kód az ablak egészére kiterjedő része (META)

Az egységek, illetve a csempék eltávolítása egy UnitInfo, illetve egy TileInfo listában történik. A hatalmas mennyiségű statikus control miatt itt a betűméretek skálázása egy iterációval történik. kivéve a címet és az egység rétegen szereplő RichTextBoxot


```

private void ScaleFontSize(SizeF scale)
{
    lbTitle.Font = new Font(lbTitle.Font.FontFamily, lbTitle.Font.Size * scale.Height);
    unitDescription.Font = new Font(unitDescription.Font.FontFamily,
                                    unitDescription.Font.Size * scale.Height);

    foreach (Control control in Controls)
        control.Font = new Font(control.Font.FontFamily, control.Font.Size * scale.Height);
}

```

A rétegek váltása a bal oldalon történő ComboBox SelectedChanged eseményének meghívásával történik. Ilyenkor a SelectLayer() metódus fut le:

```

private void SelectLayer(object sender, EventArgs e)
{
    mapLayer.Visible = false;
    mapLayerMenuPanel.Visible = false;
    unitLayer.Visible = false;
    tileLayer.Visible = false;

    switch (layerSelector.SelectedItem.ToString())
    {
        case "Map":
        {
            mapLayer.Visible = true;
            mapLayerMenuPanel.Visible = true;
            break;
        }
        case "Unit":
        {
            unitLayer.Visible = true;
            break;
        }
        case "Tile":
        {
            tileLayer.Visible = true;
            break;
        }
    }
}

```

Az egységek, illetve a csempek rétegében a bal oldali DataGridView soraiban láthatjuk a definiált objektumokat. Ez a megoldás ahhoz vezet, hogy egy egység illetve egy csempe egy névvel csak egyszer definiálható, pedig a játék motorja ezt megengedné.

Az egységek és a csempek definiálásánál lehetőségünk van az adott egység vagy csempe törlésére illetve szerkesztésére is. A szerkesztés az „Edit Mode” feliratú CheckBox bekapcsolásával történik, mely nincs külön változóba mentve, így külön kezelt a két rétegen. A szerkesztő mód kiválasztásakor a CheckBox CheckedChanged eseményén fut le a két metódus, mely váltja a hozzáadó, illetve a szerkesztő mód váltást is. A név textboxon a bevitel kikapcsolásra történik, mert a szerkesztőben az objektumok neve meghatározza őket, így az új név új objektumot jelentene szerkesztés helyett. A szerkesztő módban kiválasztáskor a DataGridView SelectedChanged eseménye a kiválasztott egység vagy csempe adatait az input mezőkbe teszi az egyszerű szerkeszthetőség miatt.

```

private void ChangeUnitEditMode(object sender, EventArgs e)
{
    btnEditUnit.Enabled = cbUnitEditMode.Checked;
    tbUnitName.Enabled = !cbUnitEditMode.Checked;
    btnAddUnit.Enabled = !cbUnitEditMode.Checked;

    dgvUnitList.ClearSelection();
}

private void ChangeTileEditMode(object sender, EventArgs e)
{
    btnEditTile.Enabled = cbTileEditMode.Checked;
    tbTileName.Enabled = !cbTileEditMode.Checked;
    btnAddTile.Enabled = !cbTileEditMode.Checked;

    dgvTileList.ClearSelection();
}

```

A beviteli mezők típusa megegyező az egységek illetve a csempék adatainak típusával, két kivétellel:

- Mivel csak két egységtípus támogatott, egy CheckBoxon keresztül nyilvánul meg az egység típusa.

```
byte type = (byte)(unitIsHealer.Checked ? 1 : 0);
```

- A csempék karakterét a szerkesztő automatikusan kezeli. Fontos, hogy egy karakter csak egy csempéhez legyen hozzáfűzve, így a szerkesztő a konstruktorában egy Dictionary<char,bool> segítségével nyomon követi a foglalt karaktereket, és aszerint fűz a térképhez.

```

private void AddCharacterMap()
{
    for (byte asciiValue = 33; asciiValue < 126; asciiValue++)
        if (asciiValue != (byte)';')
            characterMapping.Add((char)asciiValue, false);
}

```

A karakter térképezés a 33-as ASCII karaktertől kezdődik, és 125-nél fejeződik be, így a szerkesztő összesen 92 csempe definiálását engedi. Ha elfogyott a szabad karakter, a szerkesztő figyelmeztet.

A törlés a DataGridView oszlopának törlésénél kívül a listabejegyzés törlését is jelenti. Mivel játékosonként a falvak külön vannak definiálva, a törlést egyszerre kell megoldani.

```
tiles.RemoveAll(x => x.Name == tileName);
```

Egyszerre csak egy falut lehet definiálni, és a falunak le van foglalta a „Town” név.

A hozzáadásnál a TextBoxokban, illetve a RichTextBoxban lehetőség adott arra, hogy a felhasználó az adott bevitelnél egy pontosvesszőt tartalmazó karakterláncot ad. Ez súlyos hibát okozhat a csv formátumban, így ez kezelve van, és a felhasználót piros színnel figyelmezteti az adott beviteli mező a TextChanged eseményen keresztül egy közös alprogramot használva.

```
private void CheckForInvalidCharacter(object sender, EventArgs e)
{
    (sender as Control).BackColor = (sender as Control).Text.Contains(';')
        ? Color.Red
        : Color.White;
}
```

A térkép rétegben lehetőség van a térkép csempéinek helybeli definiálására, illetve a térkép egészének a mentésére. A bal oldalt két DataGridView található, melyeknek tartalma megegyezik a csempék rétegének a listájával, azonban a falu külön található. A csempék illetve falvak kirajzolásakor itt szükségtelenek éreztem a játéklapokban definiált csempeobjektumot, hiszen a mentéshez csak a csempét definiáló karakter kell, vizuális visszajelzésnek meg a megadott textúra is, így ezt a két tulajdonságot egy külön EditorTile controlba tömörítettem.

A csempék lerakását rajzprogram szerűen terveztem, azonban a Forms felület nem támogatja globálisan az egérgomb lenyomásának érzékelését, így ezt egy kapcsolón át oldottam meg

```
tileLayout[line, column].MouseEnter += new EventHandler(PlaceTile);
tileLayout[line, column].Click += new EventHandler(delegate (object sender, EventArgs e)
{
    placementMode = !placementMode;
    PlaceTile(sender, e);
}));
```

Ahol:

- placementMode a jelenleg aktív lerakási mód. Ezt a módot egy csempére kattintással lehet kapcsolni.
- PlaceTile() metódus lerakja a jelenleg kiválasztott csempét vagy falut (ha van olyan) ha a lerakási mód aktív.

Nem lehet egységeket visszamozgatni a faluba, ezért egy fal elkerülése miatt a pályaszerkesztő nem engedi hogy két falut egymás mellé tegyünk. Továbbá a falvakat listázó ComboBox kiválasztott eleme a falu tulajdonos játékosa lesz.

Figyeltem arra is, hogy a csempe rétegben a definíció törlésekor az összes olyan berajzolt csempét törölje.

A térkép rétegben lehetőség van a pálya mentésére is, mely újraépíti a térkép objektumot a konstruktorban minden adatot megadva, és annak Save() metódusával menti. Ehhez több feltételnek teljesülnie kell:

- A térképben legalább egy egységnek és falunak definiálva kell hogy legyen.
- Minden csempét be kell rajzolni
- Minden játékosnak kell rendelkeznie legalább egy faluval.

A falvak játékosonkénti vizsgálatát egy logikai visszatérésű függvénybe különítettem `VerifyAllPlayerTowns()` néven.

```
private bool VerifyAllPlayerTowns()
{
    var allTownCharacters = tiles.Where(x => x is TownInfo)
        .Select(x => new {
            x.Character,
            IsDefined = false})
        .ToDictionary(x => x.Character, x => x.IsDefined);

    if (allTownCharacters.Count == 0) return false;

    for (byte line = 0; line < tileLayout.GetLength(0); line++)
        for (byte column = 0; column < tileLayout.GetLength(1); column++)
            if (allTownCharacters.Keys.Contains(tileLayout[line, column].Character))
                allTownCharacters[tileLayout[line, column].Character] = true;

    return !allTownCharacters.Values.Contains(false);
}
```

Mentés után visszatér a játék a főmenübe.

Felhasználói dokumentáció

Rendszerkövetelmények

Hardver

- Processzor: 1 magos 1 GHz-es vagy jobb
- Szabad memória (RAM): 256MB
- Szabad tárhely: 10MB

Szoftver

- Operációs rendszer: Windows Vista SP2 vagy újabb
- .NET keretrendszer 4.5.2 vagy újabb

Telepítési útmutató

1. Ha nincs telepítve a .NET keretrendszer 4.5.2 vagy annál újabb verziója telepítse fel.
2. A TBSSGame mappa tartalmát másolja át egy tetszőleges helyre
3. A TBSSGame.exe állomány futtatásával indítsa a programot. Amennyiben a Program Files könyvtáron belülre másolta, rendszergazdai jogosultsággal futtassa.

Menüpontok

Főmenü



3. ábra: A főmenü

Indítás után ez a felület jelenik meg

- Play A pályaválasztó menüre irányít
- Editor A pályaszerkesztőre irányít
- Options Megjelennek a beállítások

- Exit Kilép a programból

Beállításmenü

The screenshot shows the 'Options' menu with the following elements and annotations:

- 1**: Points to the 'Resolution' dropdown menu showing '1216x684'.
- 2**: Points to the 'fullscreen' checkbox, which is checked.
- 3**: Points to the 'Default' button.
- 4**: Points to the 'Cancel' button.
- 5**: Points to the 'Save' button.

4. ábra: Beállítások

1. A felbontás beállítása
2. A teljes képernyő mód kapcsolása
3. A beállítások alapértékre állítása
4. Kilépés mentés nélkül
5. Módosítások mentése

Pályaválasztó menü

The screenshot shows the 'Map selector' menu. It includes a 'Quit' button in the top right corner. Below the title bar is a table with the following data:

Players	Name	Created By
2	MAP2	Yiselita
2	Summern't	John Smith
2	Za Map	Yiselita

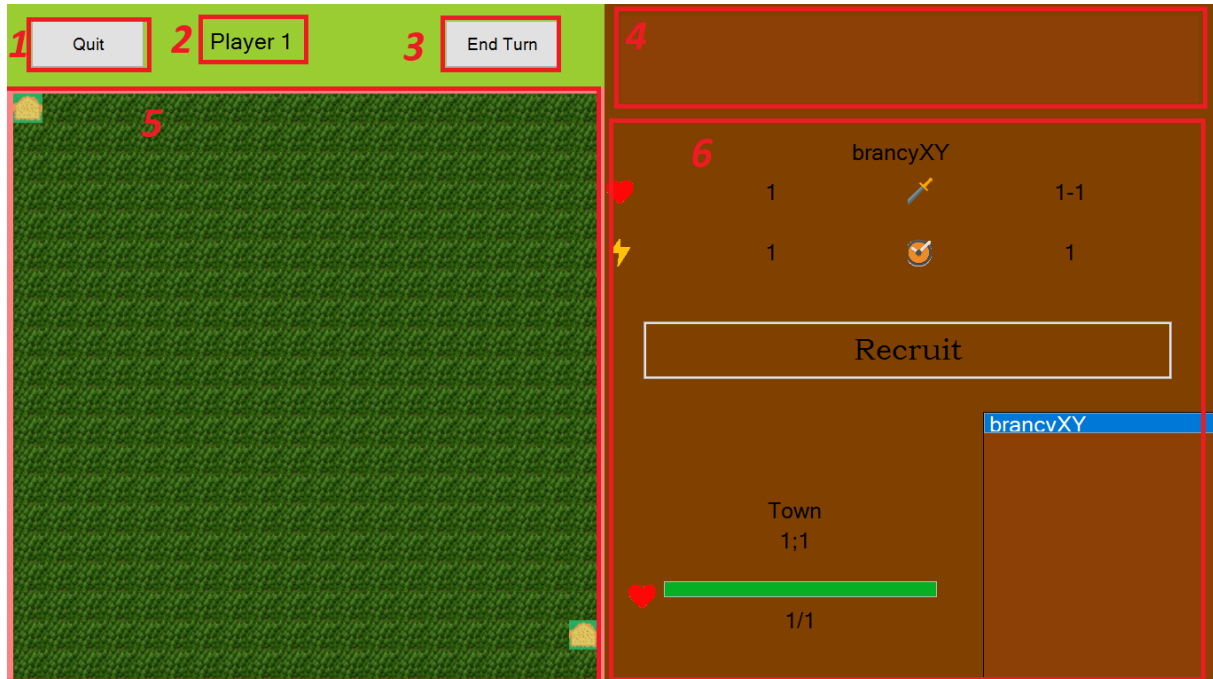
To the right of the table, a message states: 'Successfully loaded 3 maps!'. Below the table is a large grey rectangular area. To the right of this area is a light blue button labeled 'Select'.

5. ábra: Pályaválasztó

- Select
- Quit

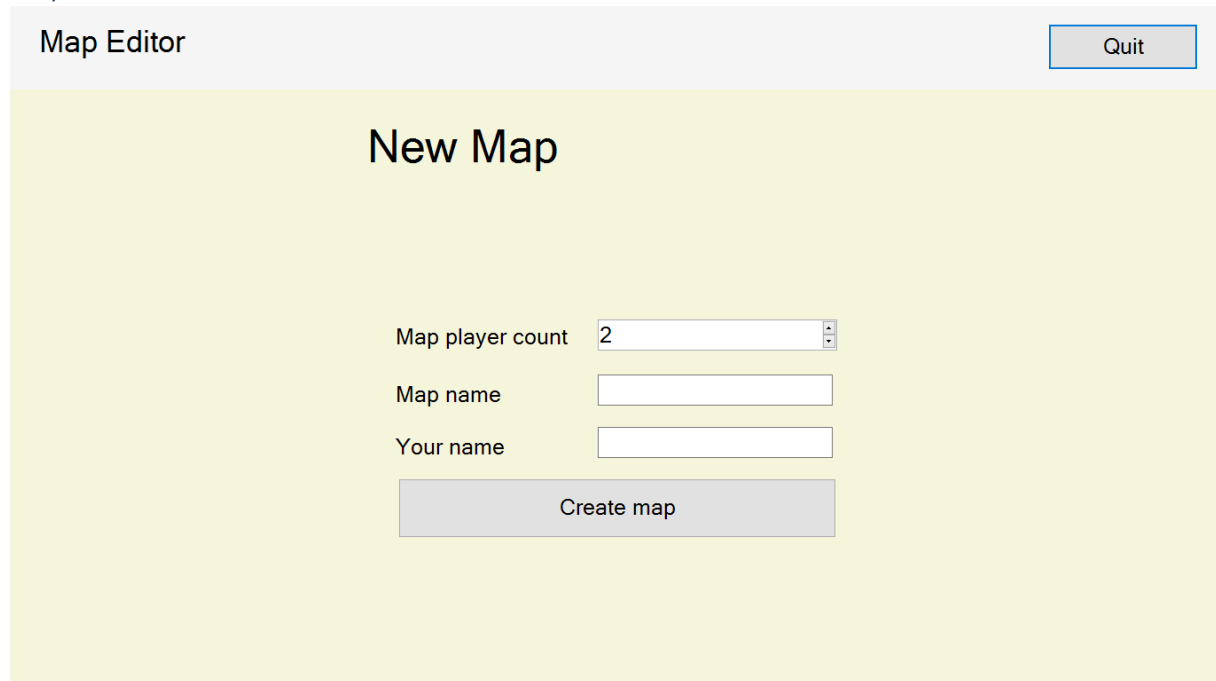
A kiválasztott térképet betöltve behozza a játéklablakot
Visszatérés a főmenübe

Játéklablak



6. ábra: Játéklablak

1. Visszatérés a főmenübe
2. A soron lévő játékos száma
3. „Kör Vége” gomb.
4. „logger”
5. A játékterület
6. A kiválasztott objektum megjelenő menüje



Map Editor

Quit

New Map

Map player count

Map name

Your name

Create map

7. ábra: Pályaszerkesztő

A pályaszerkesztő kezdőmenüje így néz ki.

- Map player count: A térkép játékoszáma (2-12)
- Map name: A térkép neve
- Your name: A térkép készítőjének neve
- Create map: A térkép létrehozása

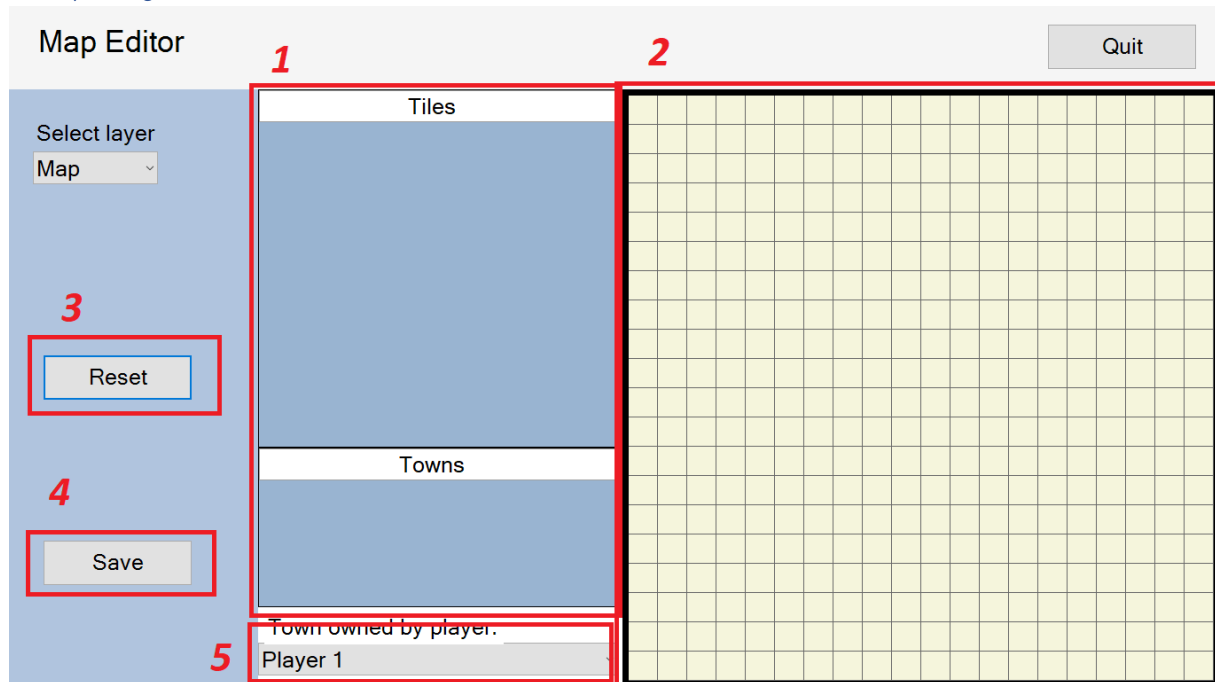
The screenshot shows the 'Map Editor' window. At the top right is a 'Quit' button (1). On the left is a 'Select layer' dropdown menu (2) with 'Tile' selected. In the center is a large blue area representing the map (3). To the right of the map is a list of units/tiles (3) with a 'Name' header. Below the list is a form (4) for editing a selected unit, containing fields for 'Name' (set to 'Town'), 'AP reduction' (0), 'Defense bonus' (0%), 'Town Health' (1), and 'Town Regeneration' (0). There is a checkbox for 'is Town?'. Below the form is a button (5) with a small image icon and the text 'Click to choose image'. Below that is a checkbox (6) labeled 'Edit Mode'. At the bottom are three buttons: 'Add' (7), 'Edit' (8), and 'Delete'.

8. ábra: egység/csempe réteg

1. Kilépés a térkép mentése nélkül
2. Más rétegre váltás
3. Az egységek/csempék listája
4. Az egység/csempe adatai
5. Kép választása az egység/csempe számára
6. Szerkesztés mód bekapcsolása
7. Új bejegyzés felvétele
8. A kiválasztott egység/csempe szerkesztése

Az egység/csempe definiálása csak egy kép választásával történhet.

Térkép réteg



9. ábra: Térkép réteg

1. A csempék, illetve a falvak listája
2. A térkép rajzterülete
3. A rajzterület törlése
4. A térkép mentése
5. A kiválasztott falu birtokló játékosa

A csempe lerakásához kattintson egy csempére és mozgassa az egeret amíg rajzolni kíván. A rajz mód kikapcsolásához kattintson egy csempére a végén. A falvakat nem lehet egymás mellé lerakni

A térkép mentéséhez a következő feltételeknek kell teljesülnie:

- A térképben legalább egy egységnek és falunak definiálva kell hogy legyen.
- Minden csempét be kell rajzolni
- Minden játékosnak kell rendelkeznie legalább egy faluval.

Játékmenet

A játékot több játékos játszva felváltva local multiplayerben.

A játék célja az ellenségek falvainak lerombolása a falvakból képzett egységekkel. Minden egységnek meghatározott akciópontja van, melyet adott mértékben csökkentenek a csempék. A csempék stratégiai szerepet is játszhatnak, mert a kevesebb lépésszám mellett sebzéscsökkentést is adhatnak. Az egységre való kattintás után az egység körül négy irányban megjelennek színes akciógombok. A zöld akciógombra való kattintás után az egység odalép, a sárgára kattintva a gyógyítók gyógyíthatnak, a pirosakon meg támadni lehet. Amennyiben több teendőnk nincs a kör végén a „kör vége” gombra kattintva a következő játékosnak adhatjuk az irányítást. A saját egységeink zöld, míg az ellenséges játékosoké piros háttérrel rendelkeznek. A játékos, akinek lerombolták az összes falvát kiesik, és az összes egysége meghal. A játék véget ér, amikor csak egy játékos marad életben, kinek lehetősége van egyedül folytatni a játékot freeplay módban.

A megadott pályák mellett saját térképeket is készíthetünk a pályaszerkesztő segítségével.

Továbbfejlesztési lehetőségek

Játék

- „Triggerek” bevezetése, melyek feltétel-akció párokból álló események a pályákon belül. Ezek hatalmas dinamikát vezethetnek be a játékba
- Online többjátékos mód bevezetése
- A pályákon belüli fájlok kezelése a pálya kicsomagolása nélkül
- Nyersanyagok bevezetése, melyek tovább segítik a stratégiai vonalat
- Több egységtípus bevezetése, például egy távolharci egység vagy egy egység, ami hatékony a falvak ellen.

Pályaszerkesztő

- Jelenleg a pályaszerkesztő technikai korlátozások miatt kevesebbre képes, mint a játék által adott funkciók. Bizonyos korlátozások azért vannak, hogy a játék le tudjon zajlani minden esetben, azonban néhány dolog (például az, hogy egy névvel csak egy egység illetve csempe rendelkezhet) a szerkesztő gyengesége. Ezen javítanék.

Összegzés

Eddig ez a legnagyobb projekt amivel foglalkoztam és egy játékmotor helyett egy játékfejlesztésre sokkal kevésbé alkalmas környezetben bonyolítottam le. Sokat tanultam az objektum orientált programozási módszerről illetve a C# nyelvről. Az Unity által előre megírt magas szintű utasítások és objektumkezelés hiányában nekem kellett sok mindent megírnom, ezért nagyobb belátásom lett egy játékmotor mint program működésébe is.

A pályaszerkesztő hiányos funkciói választ adtak egy ősidők óta aggasztó kérdésre is.

Források

- A teszt térképekhez használt csempeképtextúrái: <https://opengameart.org/content/medieval-rts-120>
- IniParser könyvtár: <https://github.com/rickayah/ini-parser>
- Óra ikon: <https://icon-icons.com/icon/preferences-system-time/94511>
- Kard ikon: <http://clipartmag.com/download-clipart-image#sword-png-12.png>
- Pajzs ikon: https://www.iconexperience.com/g_collection/icons/?icon=shield
- Szív ikon: <https://www.istockphoto.com/photo/red-heart-isolated-on-white-background-gm637711124-113894037>
- Energia ikon: https://www.iconfinder.com/icons/3859139/energy_forecast_lightning_storm_weather_icon

Ábrajegyzék

1. ábra: A menürendszer	3
2. ábra: A játékterület rétegei	18
3. ábra: A főmenü	25
4. ábra: Beállítások	26
5. ábra: Pályaválasztó	26
6. ábra: Játéklablak	27
7. ábra: Pályaszerkesztő	28
8. ábra: egység/csempe réteg	29
9. ábra: Térkép réteg	30