

Budapesti Műszaki Szakképzési Centrum Verebély László
Szakgimnáziuma és Szakközépiskolája

54 213 05 OKJ Szoftverfejlesztői képzés

Konzolos kvízprogram és space shooter

Pénzenergia Generátor és SuperZekkel

Készítette: Székely Dávid Attila

Konzulens tanár: Juhász Zoltán

2021.04.10.

NYILATKOZAT A SZAKDOLGOZAT EREDETISÉGÉRŐL

Alulírott Székely Dávid Attila a **BMSZC Verebély László Technikum 54 213 05 OKJ Szoftverfejlesztői képzésében** részt vevő hallgatója büntetőjogi felelősségem tudatában nyilatkozom és aláírással igazolom, hogy

a Konzolos kvízprogram és space shooter: Pénzenergia Generátor és SuperZekkeli

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, 2021. április 15.


Hallgató aláírása

TARTALOMJEGYZÉK

I. BEVEZETÉS	5
Alapkonceptió és a megfelelő könyvtár fellelése	5
Lehetőségek feltérképezése	6
Vágó István-mód	7
II. FEJLESZTŐI DOKUMENTÁCIÓ	9
A SuperZekkeli felépítése	11
A főképernyő felosztása, vizuális modulok.....	11
A játéktéren kívüli képernyők	11
Könnyű módosíthatóság és moduláris jelleg	12
Közérthetőség	12
Balansz.....	13
A Pénzenergiagenerátor program megszervezése, megtervezése	14
Az Intro	15
A Plazma háttér	17
ANSI grafikák betöltése	18
Info képernyő.....	19
A játék indulása	21
A játékmenet.....	23
A háttér	28
A négy szörnytípus:	33
Grafikák	36
Alapanimációk.....	37
Bónusz animációk.....	37
Jelző karakteranimációk	38
Jelző háttéranimációk	38

Szörnyek és Bónuszok osztálydiagramja	40
Game Over & Victory	44
III. FELHASZNÁLÓI DOKUMENTÁCIÓ	45
IV. ÖSSZEGZÉS	47
V. ÁBRAJEGYZÉK	48
VI. FELHASZNÁLT IRODALOM.....	50
VII. FELHASZNÁLT PROGRAMOK.....	51

I. BEVEZETÉS

Alapkonceptió és a megfelelő könyvtár fellelése

Célom egy konzolos felületen működő, de egeret is támogató kvízprogram létrehozása volt. Esztétikailag és logikailag is a számomra gyerekkoromban oly' kedves MSDOS programok világát próbáltam megidézni. Azt gondoltam, hogy a SCI-FI tematikához jól illik a korábbi generációk számítógépes univerzuma. Mivel a C# programnyelvvel kezdtem a programozással való ismerkedést, ezért célszerűnek tűnt, hogy Visual Studio-ban a .NET Framework keretein belül kezdjem meg a munkát. Azt tudtam, hogy nem csak emulálni akarom a konzolos világot, hanem ANSI és ASCII grafikát szeretnék felhasználni, és lehetőleg saját készítésűt. Azt is elhatároztam, hogy megírt sablonok helyett kitalálok egy saját menü és játékstruktúrát, megírom úgy, hogy csak billentyűzetbevitel segítségével is működőképes legyen, és a stabil váz létrehozása után különböző opcionális funkciókkal fogom bővíteni. Kitaláltam azt is, hogy egy-két egyszerűbb klasszikus demoscene animációt fogok írni C#-ban konzolra.

Második opcionális célként azt tűztem ki, hogy létrehozok egy klasszikus nyolcvanas évekbeli Arcade stílusú space shooter játékot. Így a helyes válaszlót a kvízprogram után meg lehet jutalmazni egy játékkal.

A kvízzájátnak a Pénzenergia Generátor Sci-fi Edition nevet adtam.

A space shooternek a SuperZekkeli címet és a Return of the DimensionDrifter alcímet.

A két játék kompatibilis, mert a konzolméret megegyező és ugyanazt a könyvtárat használják. Tematikájuk egyező, ezzel szemben a játék stílusa teljesen különböző.

Elhatároztam már a legelején, hogy bár a lehető leghűségesebb leszek a felállított esztétikai és logikai alapelvekhez a programom létrehozásakor, de a kereteken belül maradva nem vetem meg az esetleges külső könyvtárak használatát. Így erre a célra a SadConsole nevű,

egyszemélyes fejlesztésű, nyílt forráskódú és ingyenesen használható library-t találtam a legalkalmasabbnak.¹

Ezt a könyvtárat Andy (De George) Thraka hozta létre, és fejleszti immár egy évtizede folyamatosan². A SadConsole a XNA-ra épülő nyílt forráskódú C# framework, ami talán a Stardew Valley különböző portjaihoz való felhasználása miatt lett igazán széles körben ismert.³ Mikor megkezdtem a munkámat, a SadConsole v8.9.1 volt a legújabb, és a programom kapcsán végig ki is tartottam emellett a verzió mellett, mert azt tapasztaltam, hogy a különböző verziók nem keresztkompatibilisek egymással.⁴

Az ANSI grafikák létrehozására egy Moebius nevű ANSI editort használtam. Ezenkívül a következő szerkesztő programokban dolgoztam: REXpaint nevű, kortárs fejlesztésű ASCII art editor, valamint a Aseprite, szintén kortárs fejlesztésű pixeleditort valamint Gimpet, ami egy ingyenes képmanipulációs szoftver.

Lehetőségek feltérképezése

A tényleges fejlesztés előtt felmértem, hogy milyen lehetőségek rejlenek a SadConsole használatában.

Előnyei, amelyek megkönnyítik egy konzoljáték készítését:

- Tetszőleges számú, méretű és színű és egyéb tulajdonságú konzolt tud egyidejűleg kezelni, és ezek a konzolok cellánként paraméterezhetők. Ezen kívül az előtér és háttér színeinek kezelését megkönnyítő függvényekkel rendelkezik.

- a fontokat png kiterjesztésű képfájlból tölti be. Többféle méretű és típusú konzolt lehet így egy programon belül használni.

1 <https://sadconsole.com/index.html>

2 <http://www.roguebasin.com/index.php?title=SadConsole> „,Released: Sep 01 2011 (1.0.0)”

3 <https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-monogame-the-right-game-engine-for-you>

4. MonoGameből a 3.7.1.189-est használtam NuGet package-nak.

-Támogatja a billentyűzet- és egér bevitelt, így ezek kezelésére már külön rendszert nem kell létrehozni, elég a meglévőt használni a lehetséges felmerülő input-esetekre.

-Be tud olvasni ANSI fájlokat, és kezelni is tudja azokat.

Hátrányai:

-A kezelés megtanulását segítő dokumentumok elég hiányosak, kevés elérhető tutorial és segédanyag van hozzá az interneten.

A megtalálható segédanyagok nagyrészt a roguelike típusú játékok létrehozását támogatják.

-Kevés az elérhető kész projekt, amiből meg lehetne könnyen tanulni a használatát.

Munkám során fedeztem fel a SadConsole Discord szerverét, ahol három addig megoldatlan problémámra kaptam választ a SadConsole fejlesztőjétől.

-Két hátrány viszont menet közben derült ki.

Az első az ékezetes karakterek eleve problémás használata. A második probléma is ehhez köthető, az 'ű' illetve 'Ű' karakter használatakor az egész rendszer képes összeomlani.

Vágó István-mód

Úgy gondoltam, hogy muszáj hűségesnek lennem a magyar kvízzjátékok történelméhez, ezért Vágó István digitális mását opcionálisan megjeleníthetővé kell tennem majd a játék során. (Természetesen ASCII grafika formátumban.)



1. ábra: Pénzenergia Generátor logo



2. ábra: SuperZekkeli kezdőképnyő

II. FEJLESZTŐI DOKUMENTÁCIÓ

A Pénzenergia generátor felépítése

Két részből áll:

1. Menü, melyet egy kis almodul, egy intro vezet fel
2. A kvízzjáték játékmenete

A kvízzjáték menüjének felépítése

A játék egy rövid intróval kezdődik, úgynevezett SplashScreennel, ahol egy animált háttéren feltűnik a játék neve. Ezután áttűnéssel, bejön egy menü egy gomb lenyomásakor. A menühöz szintén animált hátteret készítettem a következő menüpontokkal:

- 1.) Játék menü, amely elindítja a program Kvízzjáték részét
 - 2.) Beállítások menü, ahol a játékot lehet paraméterezni:
 - a kérdésekre adott idő hosszát
 - a játék megnyeréséhez szükséges helyes kérdések számát
 - a játék nyelvét (magyar v. angol)
 - illetve a fent említett Vágó-mód bekapcsolását.
 - 3.) Info menü, amely rövid textuális illetve vizuális információt ad a játék készítőjéről. (Egy betöltött ANSI grafika formájában.)
 - 4.) Egy Kilépés menü, amely befejezi programom működését.
- A Beállítások menü hátterét szintén ANSI grafikával oldottam meg. Különös gondot fordítottam arra, hogy a gombok feketék legyenek, fehér felirattal, amely, ha az egeret fölé visszük sárgára vált. Mindegyik gomb funkcióját el lehet érni billentyűlenyomással. A gombok billentyű parancsa a konvenciónak megfelelően "P)" formában rajta van a gombokon.

A beállítás menüben csuszkákat alkalmaztam az értékek megadásához. A billentyűparancsok alul találhatóak.

A játékmenet

A kvízzjáték magját egy iskolai feladat adta, amely egy egyszerű kvízzjáték. A program egy adott txt fájlból soronként beolvasta a játékra specifikus adatokat:

(opcionálisan: a kérdések száma)

-egy adott kérdésnél:

-a kérdés

-a négy lehetséges válasz

-a helyes válasz betűjele

és ez ismétlődött a forrás szövegfájlban ciklikusan. Ezt logikailag változatlan, formailag részben változatlan módon meghagytam a programban. Azzal az egy variációs lehetőséggel éltem, hogy többnyelvű támogatást építettem a programba, amelyhez több ilyen egymást tükröző txt-t lehet a programomba felvenni, például angol és magyar nyelvűt. Így a kvízprogram tartalma könnyen módosítható. Elég csupán a forrásként szolgáló txt-ket átírni és mások lesznek a kérdések. Sőt akár több nyelvvel is egyszerűen lehet bővíteni.

A SuperZekkeli felépítése

A főképernyő felosztása, vizuális modulok

A Space shooter-t a lehető legminimálisabb, legletisztultabb módon akartam kivitelezni. Egyetlen játékképernyőt, egy pályát hoztam létre, amely majdnem négyzet alakú. Oldalt találhatók a szükséges aktuális adatok (statisztikák).

-Az életerő

-Az energia

-Az elért pont

Ezenkívül alul van egy logkonzol, amin a megértést könnyítő éppen aktuális változások színes betűkkel jelennek meg, például, ha egy szörny megsemmisített egy bónuszt „Bonus destroyed”, pirossal.

A játéktéren kívüli képernyők

A játéktéren kívül három darab képernyő van, mindhárom tájékoztató jellegű ANSI grafika.

1. Kezdőképernyő.

Rajta a játék címe, a készítő adatai és az évszám.

A kezdőképernyő az Enter gomb lenyomásával átugorható.

2. Game Over képernyő

Ezt akkor jeleníti meg a program, ha az adott játékos elvesztette a játékot. Egy plazmában lángoló koponyát ábrázol, egyértelműsítve, hogy elérte hősünket (az elkerülhetőnek hitt) végzet.

3. Victory képernyő

Ez a játékos úrhajóját ábrázolja, amely diadalmasan tör a fellegek felé. Ez a képernyő nem teljesen statikus: egy mini konzol is a részét képezi, amely megjeleníti a játékos által elért pontszámot.

Könnyű módosíthatóság és moduláris jelleg

Munkám során ügyeltem a különböző elemek modulárisra tételére, a későbbi könnyebb kiegészítés, módosítás érdekében. A SadConsole-nál ez különösen fontos, hiszen az egész játékot különböző tulajdonságú konzolok összességéként kell meghatároznunk. Programom bemutatásakor is ezeknek a különböző modulokból álló struktúráknak és az ezeket kezelő programrészeknek legátláthatóbb bemutatására törekedtem.

Közérthetőség

Arra is ügyeltem, hogy a felhasználói oldalról könnyen értelmezhető legyen minden szituáció. A kvízzjátékban például a gombok és opciók billentyűparancsát egyértelműen jeleztem:



Ez a Beállítások menü egyik alsó sora. Zárójelben a billentyű, mellette a funkció, amiért felel.

A SuperZekkelinél a közérthetőséget a grafika és a játékmenet szempontjából is szem előtt tartottam. Arra törekedtem, hogy az információk megjelenítése teljesen egyértelmű és letisztult legyen.

Fontosnak tartottam azt is, hogy a pálya és a játékszituációk átláthatók, a grafikák minimalisták, kifejezők legyenek.

Balansz

A SuperZekkelit a fejlesztés során sokat teszteltem, és arra törekedtem, hogy a játék nehézsége olyan legyen, hogy én, a játék fejlesztője, körülbelül az esetek 50 %-ában tudjam végigvinni a saját játékomat.

A Pénzenergiagenerátor program megszervezése, megtervezése

A főprogramban statikusan meghatároztam a fontosabb konzolokat.

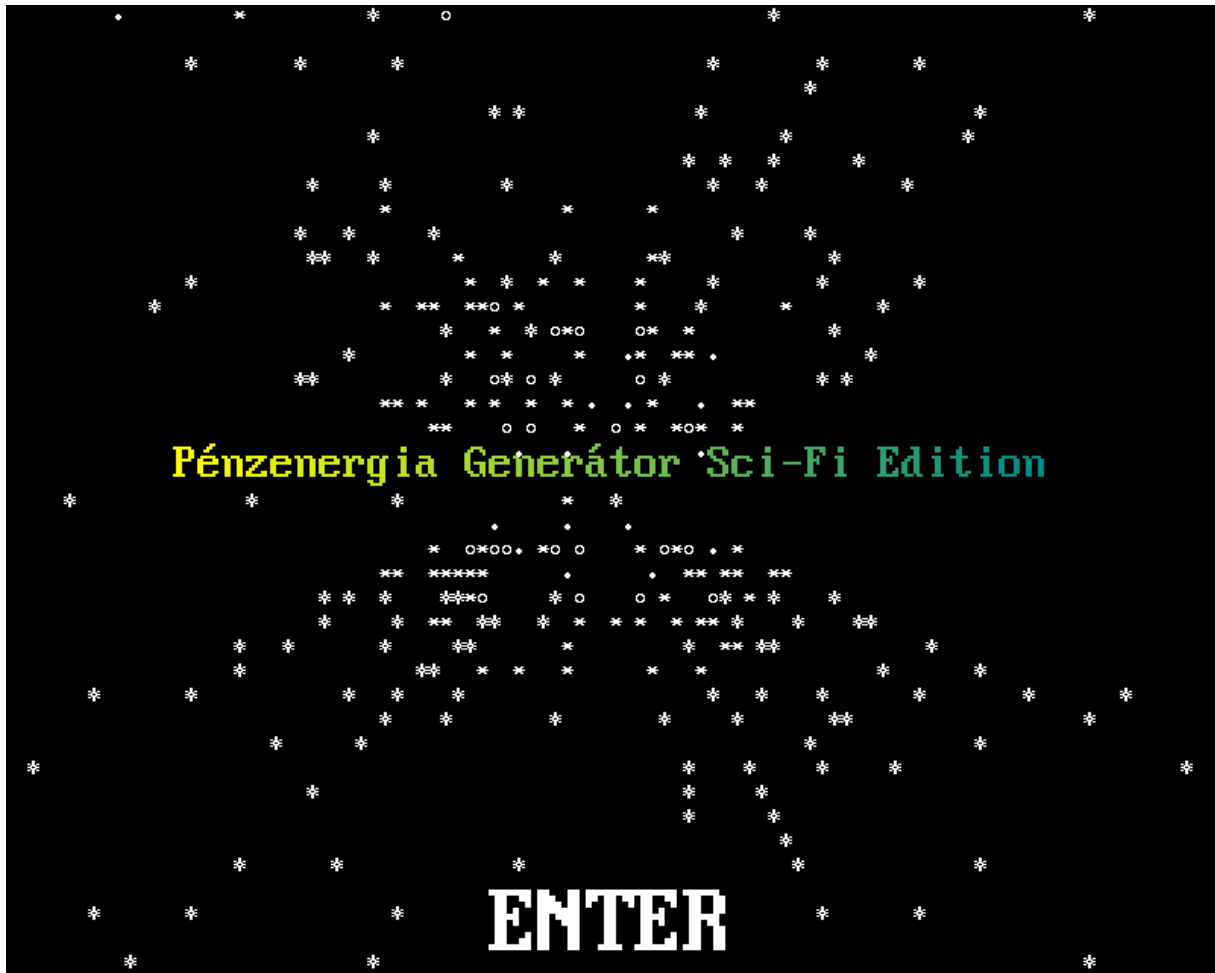
Létrehoztam mindegyik egységből (MainMenu, Options, MContainer, InfoConsole) egy-egy példányt, illetve egy MoveNextconsole metódust, amely abban segít, hogy a létrejött fő konzolok között váltani lehessen.

Mindig az oda beadott konzolt teszi aktívvá és láthatóvá, minden mást meg töröl.

Ezen kívül létrehoztam egy Reset metódust, amely a játék minden változó részét visszaállítja alapállapotba.

A következő oldalakon sorban végighaladok a program egyes részein, és bemutatom azokat.

Az Intro



3. ábra: Pénzenergia Generátor intro

Az intro a hagyományos Starfield animáció konzolos, grafikailag leegyszerűsített verziója. A Starfield animáció során mintha különböző csillagok között haladna a néző. Ezt a hatást úgy éri el az animáció, hogy a perspektíva miatt a távolabb lévő csillagok kisebbek és más színűek.

Mivel a konzolon méretváltoztatásra nincs lehetőség, ezért inkább egy „Csillagszórás” jellege van az animációnak.

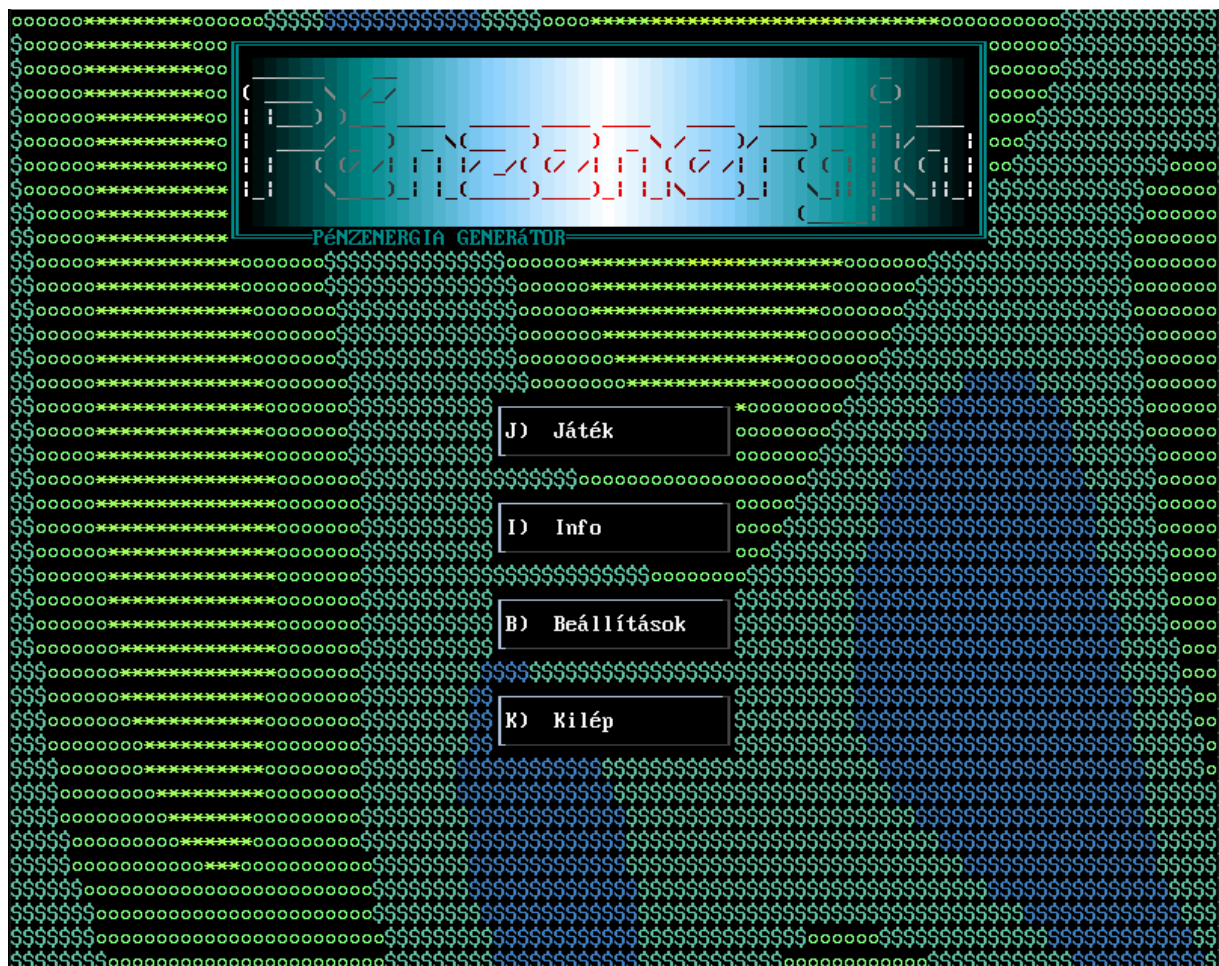
Az animáció alatt a játék címe jelenik meg, betűnként kiírva, színesen. Ez egy olyan része a programnak, ahol örömmel használtam a SadConsole beépített lehetőségeit, mind a színátmenet, mind az animáció tekintetében.

Alul egy villogó "Enter" felirat adja egyértelműen tudtára az érdeklődőnek, hogy mely gomb lenyomására fog történni valami.

A starfield animációkban három koordinátával határozzuk meg a csillagokat, háromdimenziós térben gondolkodunk. Mindegyiknek van x, y és z koordinátája.

Animációmban a csillagok mérete nem változik, így más hatást eredményez, mint egy klasszikus starfield animáció. A z koordináta függvényében nem a csillagok mérete, hanem a formája változik: más karakterként jelennek meg.

Az intro elején és végén egy egyszerű Fade (áttűnés) effekt van. Enélkül túl éles lett volna a váltás a két képernyő között.



4. ábra: A főmenü

A Menu header

A menü headerje a program címét tartalmazza, bekeretezve, ASCII art stílusú felirattal, mögötte színátmenetes háttérrel. A színátmenetet a SadConsole színátmenet funkcióját használva generatíván írtam meg, ahogyan az ASCII megjelenítését és a keretet is. Tehát nem egy statikusan tárolt képet látunk, hanem kódból generálja a program.

A Plazma háttér

A menü háttére szintén egy klasszikus demoscene animáció újragondolása.⁵ Az egyik alapvető különbség, hogy itt a különböző brightness level-ekhez különböző karaktereket rendeltem. A minta mozgását az adja, hogy tartalmaz egy Timer-t, ami az idő függvényében módosítja a minta szerint a karakterek formáját és/vagy színét. A hivatkozásban megnevezett matematikai képleteket használtam fel kis módosítással, beleültetve a konzolos közegbe. Ezen a háttéren található egy másik konzol, aminek átlátszó a háttére, és ez tartalmazza a menügombokat. Ezért a menügombok alatt folyamatosan pulzál a plazma-háttér.

A menügombok billentyűparancsokra és egérekattintásra is reagálnak. Ha a gombok fölé viszi a felhasználó az egeret, a rajtuk lévő felirat sárgára vált.

A gombok stílusát úgynevezett Theme-kben lehet megadni. A konzol a létrehozásakor meghív egy speciális metódust, amely a megfelelő színsémát beállítja. Itt adhatjuk meg, hogy a gomb egyes helyzetekben milyen színparaméterekkel rendelkezzen.

```
Theme = Themebeallit.SetupThemesMenu();
```

Ezen beállítások után, pedig ezeket a következő sorok érvényesítik:

```
consoleTheme.ButtonTheme.Colors.RebuildAppearances();
```

⁵ Az animációhoz alap matematikai háttéranyagot, a következő oldalon találtam:

<https://lodev.org/cgtutor/plasma.html>

```
consoleTheme.Colors.RebuildAppearances();
```

ANSI grafikák betöltése

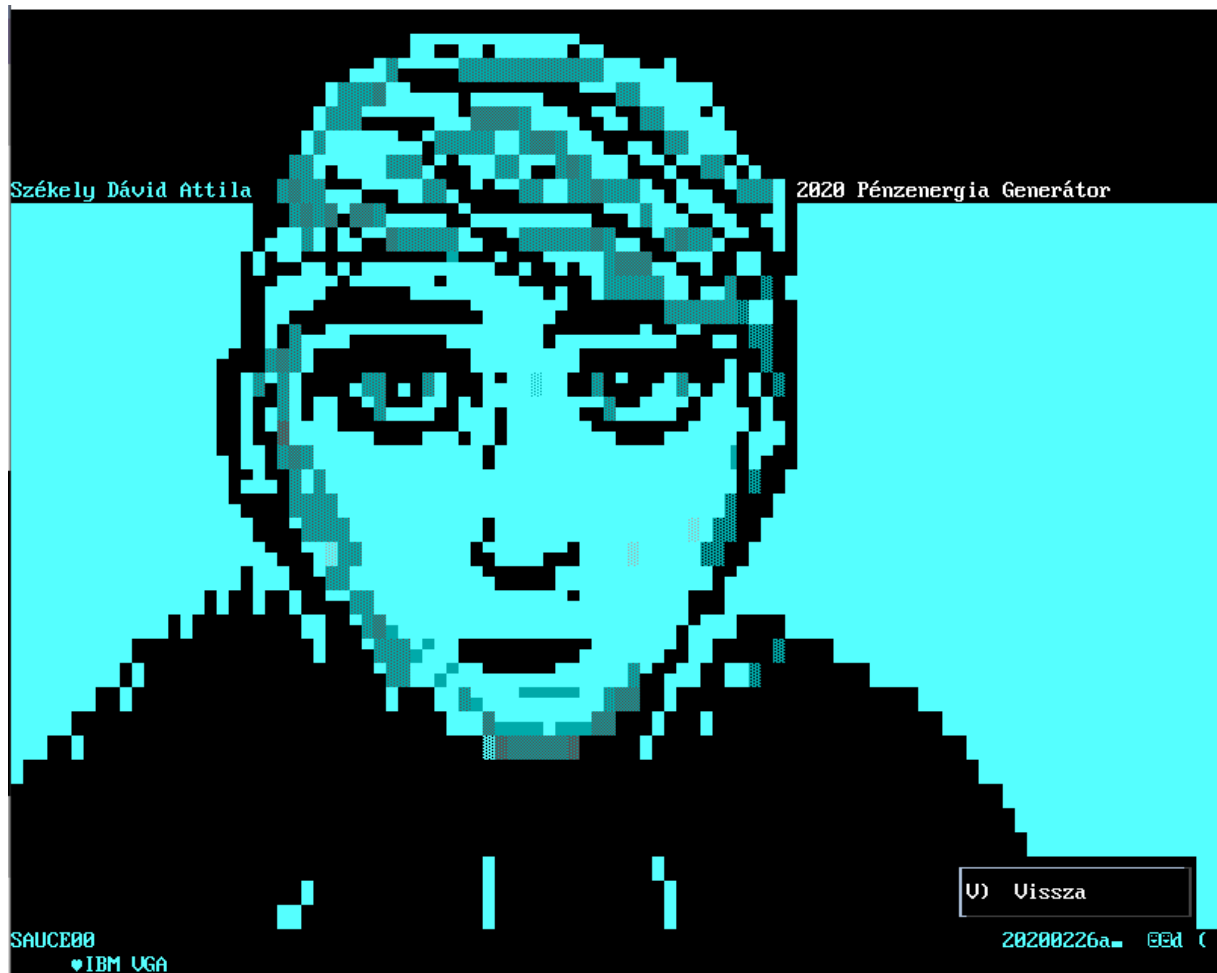
Ahhoz, hogy a grafikák ANSI jellegét egyértelművé tegyem, nem egyben, hanem idő függvényében, egységenként töltöttem be mindkét játékban. Így van a betöltésnek egy nyomon követhető folyamata.

Ezt a következő programrész valósítja meg:

```
ansiinfoswr.CharactersPerSecond = 720;
var ansitimer02 = new Timer(TimeSpan.FromSeconds(1));
double ansitime = 0;
ansitimer02.TimerElapsed += (timer, e) =>
{
    ansitime++;
    ansiinfoswr.Process(ansitime);
};
Components.Add(ansitimer02);
```

A kódból látható, hogy másodpercenként 720 karakter megjelenítésére korlátoztam a betöltést.

Info képernyő



5. ábra: Az Info képernyő

Az Info menüpontot választva előjön egy ANSI kép a készítőről és a játékról. Ezen kívül ebben az almenüben csak egy Vissza gomb található.

Beállítások



6. ábra: Beállítás menü

A Beállítás menü a korábbinál jóval gazdagabb opciókkal kecsegtet.

A következő paramétereket szabályozhatjuk itt:

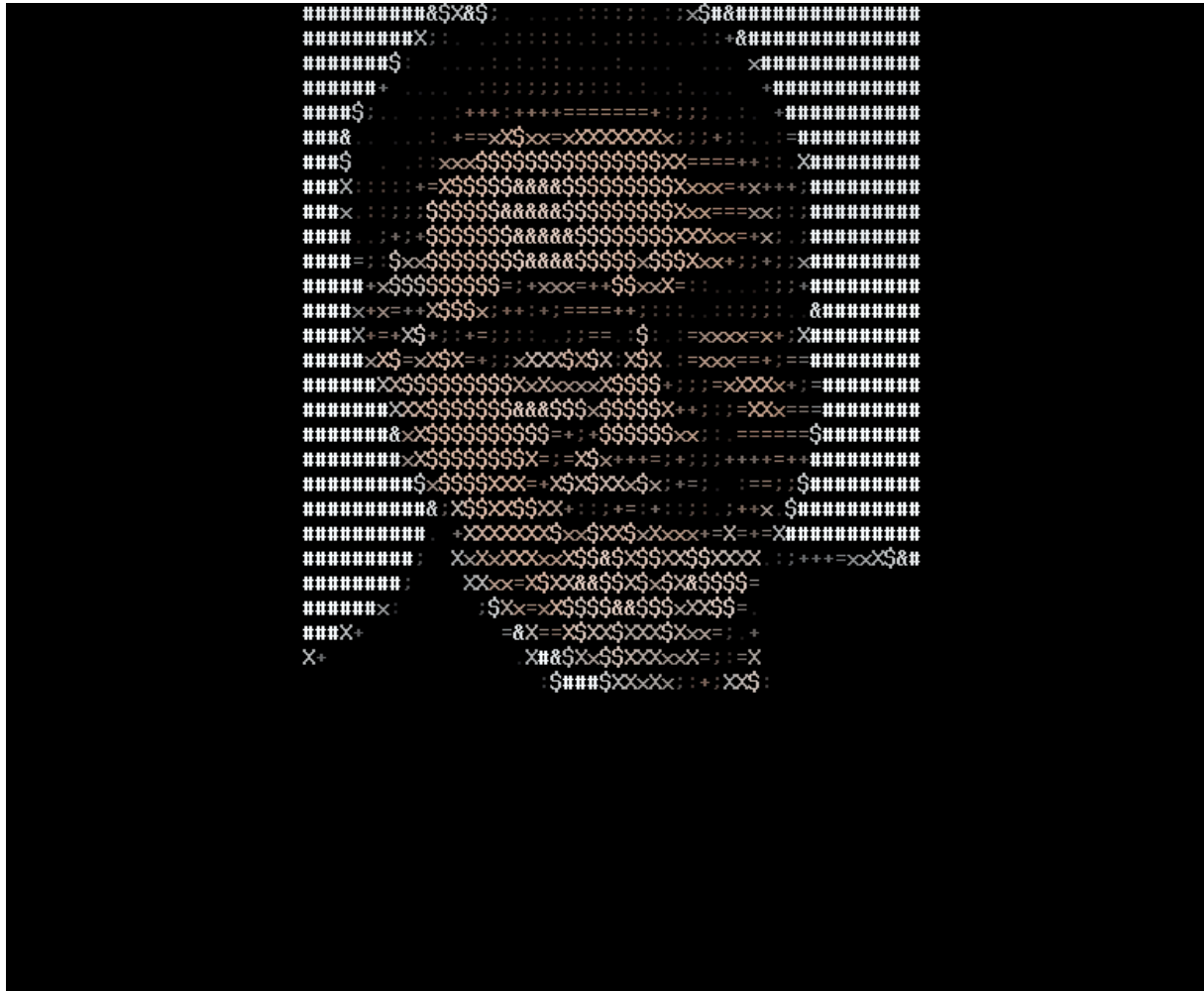
- Az egyes kérdésekre rendelkezésre álló idő mennyiségét
- Milyen nyelven folyjon a játék
- Illetve a Vágó-mód bekapcsolására nyílik még lehetőség



7. ábra: Vágó-üzemmód

A játék indulása

A Játékot a főmenü Játék gombja indítja, és ha a Vágó-mód aktív, megjelenik Vágó István ASCII arcképe:



8. ábra: Vágó István, ASCII art

Ez egy külön konzol. Hogy a SadConsole lehetőségeivel éljek, annak eszközeit használtam fel az arckép átalakításához. Az arckép forrását egy sima png kiterjesztésű fájl szolgáltatja. Ezt a képet megelőzőleg Gimp-pel kicsit átalakítottam, hogy erőteljesek legyenek a színek és a kontúrok, hogy az ASCII konvertálás után is felismerhető maradjon.



9. ábra: Vágó István, módosított fotó

Az intro során is alkalmazott Fade effektet itt is használtam a kép megjelenésekor és eltűnésekor. Az egész animáció egy InstructionSet-ként lett megvalósítva. Ebben található utasítások másodpercben, hogy az egyes fázisok között mennyit várjon a program. Ezt a fajta megoldást a kvízzjáték játékmenet részében is többször preferáltam.



10. ábra: Kvízzjáték játékmenete

A játékmenet

A játék képernyője 5 fő részből és egy háttérből áll:

- 1) Fekete felső infokonzol (kérdéseknek, vereségnek/győzelemnek)
- 2) Négy gomb a négy válaszlehetőséggel
- 3) Bal oldalt, alul egy számláló, amely visszaszámol, jelzi a hátralévő időt egy szám illetve egy színes csík („healthbar”) formájában is.
- 4) Középen alul az alsó infokonzol. Kiírja, hogy hányadik kérdésnél járunk, nullától kezdve a számozást. Kiírja az aktuális elért pontot is, hogy a tévés kvízzjátékok szellemiségét is megidézze az alkotás.
- 5) Jobb alul egy Vissza gomb, ami visszaléptet a főmenübe.

1) Felső infokonzol

Ez egy fekete háttérű konzol, amely legnagyobb (négyes) méretű betűkkel megjeleníti az adott kérdést, illetve tájékoztat a játék állásáról:

-győzelem:

”Gratulálunk! Nyertél”

majd kis szünet után: "Nyereményed: {x} egységnyi pénzenergia!"

itt az x helyén az elért pontszám található.⁶

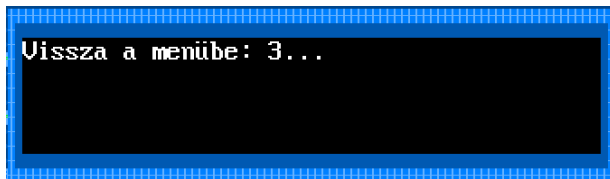
-vereség:

„Rossz válasz!”

Mindkét kimenetel esetén az információ után beindul egy számláló:

"Vissza a menübe: 3..." stb.

Mikor a végére ér, automatikusan kilépteti a felhasználót a főmenübe:



11. ábra: „Vissza a menübe” tájékoztatás

-az idő lejárt:

Ezt hasonlóan egy „Lejárt az idő!” felirat közli, és a számláló léptet vissza menübe.

Ezeket a várakozásokat és parancssorozatokat technikailag InstructioSetek segítségével valósítottam meg, ahol a .Wait parancs segítségével várakoztattam a megfelelő helyeken a programot. Az infokonzolt úgy állítottam be, hogy jelezze a kurzor helyét, hogy a konzolszerű megjelenés még karakteresebb legyen. Ez a kurzor mindig az aktuális szöveg végén van, és ott villog. A megjelenő kérdéseket egy txt fájlból tölti be a program a res(resources) mappából. A négy lehetséges válaszlehetőséggel, illetve az egy helyes válasz megjelölésével szerepelnek

⁶ Ezt a humor kedvéért pénzenergiának neveztem, utalva egy kétes tevékenységű jósnők által alkotott fogalomra, mely szerint a jövedelmet befolyásolja egy úgynevezett „pénzenergia”.

ebben a kérdések. A txt-ben elhelyeztem formázásra, betűszínekre vonatkozó parancsokat is, amiket magának a SadConsole lefordít, és már formázottan jeleníti meg a szöveget.

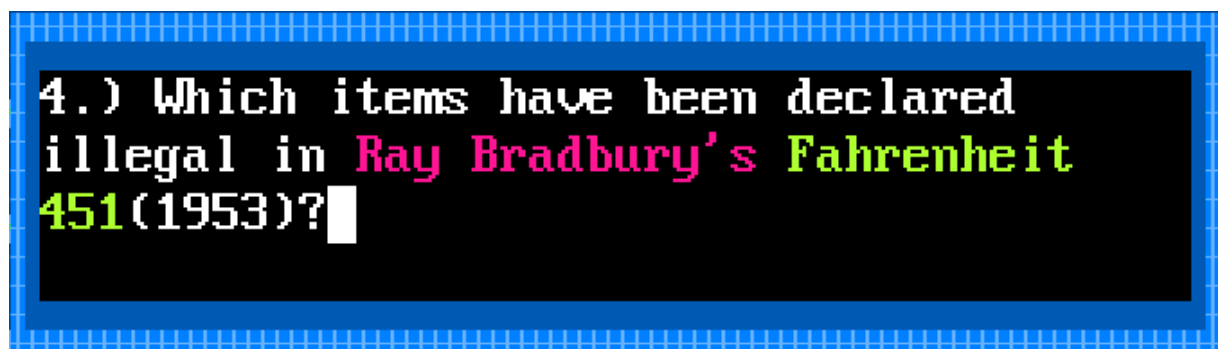
Példa egy feladatra a txt fájlból:

```
4.) Which items have been declared illegal in [c:r f:deppink]Ray  
Bradbury's[c:u]  
[c:r f:greenyellow]Fahrenheit 451[c:u](1953)?  
  
A. Books  
  
B. Thermometers  
  
C. Cars  
  
D. Televisions  
  
0
```

Itt látható, hogy a szerzőt rózsaszínnel (deppink), a könyvcímet meg sárgászölddel emeltem ki. Ezeket a jelöléseket következetesen használtam végig, ügyelve még a dátumok megjelölésére, nehogy azonos című művek esetén összekeverhető legyen. Illetve jelöltem, hogy egyes műveknél a regényre vagy a filmváltozatra vonatkozik a kérdés. Hogy a szavakat ne válassza szét, a megjelenítő a SadConsole beépített StringParserét használtam:

```
Surfaceszoveg.Cursor.UseStringParser = true;
```

```
Surfaceszoveg.Cursor.IsVisible = true;
```



```
4.) Which items have been declared  
illegal in Ray Bradbury's Fahrenheit  
451(1953)?
```

12. ábra: Formázott szöveg a konzolon

2.) A négy gomb a négy válaszlehetőséggel

A gombok szövegezésével nem volt különösebb probléma:

```
Abutton.Text = teszt[kerdesszam].ValaszA;  
Bbutton.Text = teszt[kerdesszam].ValaszB;  
Cbutton.Text = teszt[kerdesszam].ValaszC;  
Dbutton.Text = teszt[kerdesszam].ValaszD;
```

Nyugalmi állapotban a gombok színe sötétkék.

Az aktuális gomb, ami felett az egerünk van, citromsárga:



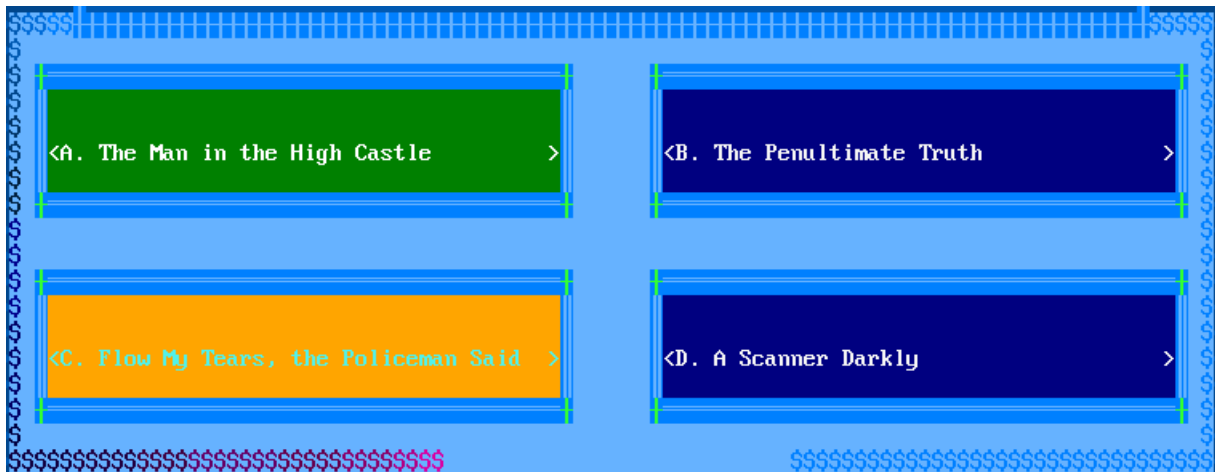
13. ábra: Citromsárgával megjelölt gomb

Miután rákattintottunk egy gombra, azt narancssárgával megjelöli a játék:



14. ábra: Narancssárgával megjelölt válasz

Ezután először citromsárgára majd zöldre vált a helyes válasz gombja:



15. ábra: Helyes válasz megjelölése

Ezt a színváltást/villogást úgy oldottam meg, hogy a program itt igazából nem a gomb színét, hanem a gomb színét meghatározó grafikai sémát (themet) is megváltoztatja a színváltáshoz.

3) Bal alsó számláló



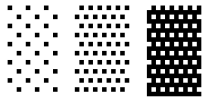
16. ábra: Bal alsó számláló

A számláló a hátralévő időt mutatja szám és színes csík (healthbar) formájában.

Egy `progressTimer` van a programban, ami másodpercenként lépteti visszafelé a kijelzőt. A számkijelzés az `Update`-ben van megvalósítva az `UpdateScreen` metódus által, és így mindig az aktuális értéket jeleníti meg a képernyőn. Ha a számláló a nullát eléri, megáll, és akkor lép életbe a `Lejartazido` függvény, ami megjeleníti a konzolon az aktuális játékhelyzetről az információt.

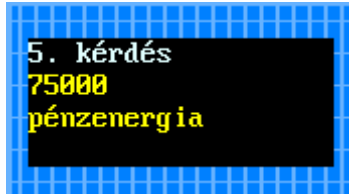
Az `UpdateScreen` része a csík is, az úgynevezett `Healthbar`. Ezt generált módon oldottam meg, nem statikusan, nem a lehetőségek egyesével való megírásával.

Azért, hogy az elhalványulás egyenletes legyen, ezt a három blokk-karaktert alkalmaztam:



, a kilencvenes évek stílusához igazodva.

4) Alsó középső infokonzol



17. ábra: Alsó infokonzol

A megjelenített értékeket egy int vektorban tároltam el, ezért minden kérdésnél, az aktuális számot jeleníti meg. Ezen kívül jelzi a kérdés indexét (sorszámát nullától kezdve).

5) Vissza Gomb

Ugyanúgy, mint a többi a főmenütől különböző rész, ez is tartalmaz egy Vissza gombot.

A háttér

A kvízzjáték háttérét egy REXPaint nevű (kortárs) ASCII editor programban rajzoltam meg. Mivel a SadConsole támogatja a REXPaint formátumát, ezért döntöttem a használata mellett.⁷ A képet pedig a SadConsole REXpaint Readerével olvasom be és helyezem egy konzolra, majd adom hozzá az üres háttérhez.

```
var rexi = REXPaintImage.Load(instream);  
var rexilay = rexi.ToLayeredConsole();  
hatterConsole.Children.Add(rexilay);
```

⁷ A RexPaintben segítségével most is sok izgalmas projekt készül. Ugyanaz a fejlesztő alkotta meg, aki a Cogmind című elképesztően összetett roguelikeot is programozta és fejleszti folyamatosan, jó pár éve.

A SuperZekkeli program megszervezése, megtervezése



18. ábra: SuperZekkeli játék közben

A játék során a játékos egy űrrepülőt irányít. A négy égtáj felé mozoghat. A játéktér egy négyzetrácsos fekete tábla, amely az űrt szimbolizálja. Az űrhajó a nyíl billentyűkkel irányítható, a tábla határain túlra nem mehet. A Space gomb lenyomásával zöld lézert lő ki. Összesen négyféle szörnyel kell megküzdenie a játék során. A szörnyek összesen három fázisban teremődnek meg a pálya bizonyos pontjain. Ezek a fázisok, (phase) az idő elteltével váltanak át a következőbe. A korai fázisban kevés szörny terem, a középsőben több, az utolsóban pedig sok. A játékos minden lövése felemészt egy egységnyi energiát.

Összesen 12 egységnyi energiája van. Ezt négy darab '>' jel szimbolizálja.

Egy '>' jel:

- ha zöld, hármat,
- ha sárga, kettőt,
- ha piros, egy energiát ér.



Ezen '>' jelek előtt egy energiát szimbolizáló villám található.

A játékos életét négy darab szívecske jelzi. Két esetben veszíthet életet a játékos. Az első eset, ha egy szörnyel ütközik, ilyenkor a szörny megsemmisül, a játékos pedig egy életet veszít.

A második eset, ha egy szörny lövése eltalálja. A szívecskek előtt egy vörös pluszjel található. Bár úgy éreztem, a szívek elég egyértelműen jelzik, miről van szó, ide is kellett egy ikon az egységesség érdekében.

19. ábra: Statisztikák

Az élet alatt található a játékos pontszáma.

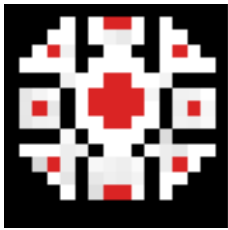
A játékos az ellenségek lelövésekor pontot kap.

A különböző ellenfelek (Monster ősosztályúak- ezt szörnyre fordítom), különböző mennyiségű pontot érnek.

A játékos tud plusz élethez és plusz energiához jutni.

A plusz életet egy speciális, kétszeresen szimmetrikus sci-fi objektum jelöli, ami pirosan villog. Egy életet ad.

Plusz élet:



A plusz energiát, egy töltés animáció jelzi. Szándékosan hasonló az elektronikai eszközök töltés jelzéséhez. Felvételekor négy egységnyi energiát ad.

20. ábra: Élet bónusz

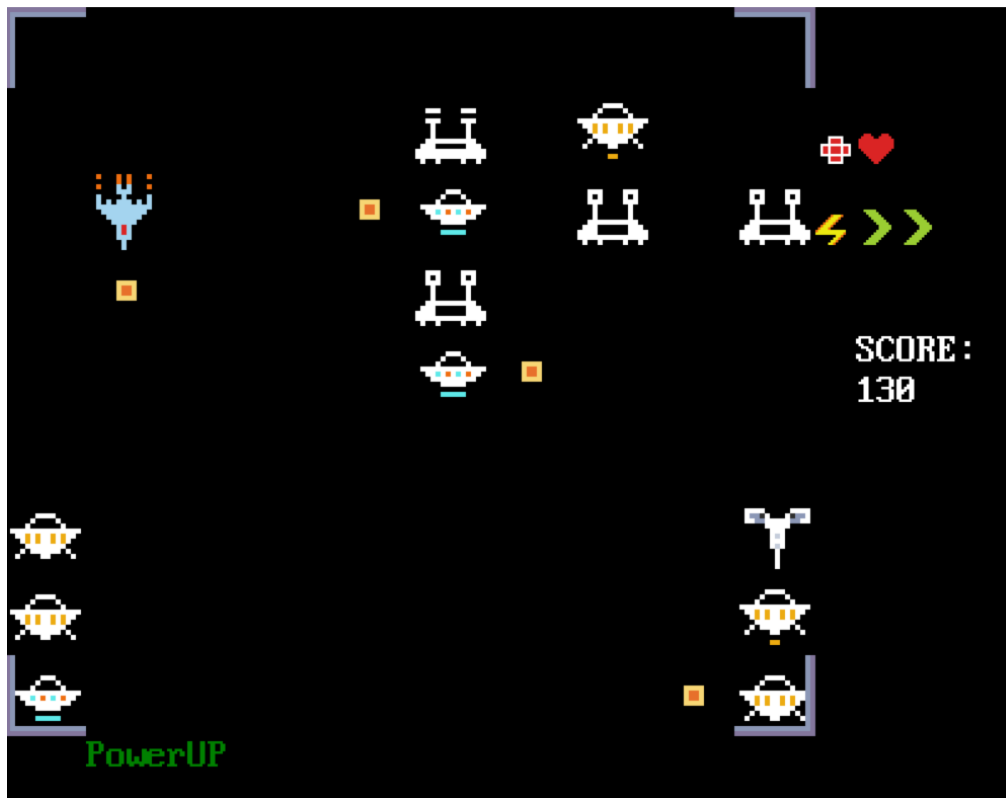
Plusz energia:



A játék folyamatait leütésekben mérem. 0.25 másodpercenként van egy leütés. A játék három fázisból áll, amelyekben egyre nehezedő körülmények mellett kell küzdenie a játékosnak.

21. ábra: Energia bónusz

A játékos karakter mozgása nem igazodik a mért időhöz, kizárólag a billentyűlenyomások sebességétől függ. Ez szükséges, mert egyébként nem lenne esélye ellenfeleivel szemben. A lézere is gyorsabb. A játékos zöld lézergömbje a kilövés irányában negyed másodpercenként tesz egy lépést. Ezzel szemben a szörnyek lézere két leütésenként, azaz fél másodpercenként mozog, ahogyan ők maguk is. Stratégiaileg fontos a játékos számára, hogy jól gazdálkodjon az energiájával miközben az életét is megőrizze. Kezdetben könnyedén le tudja vadászni a szörnyeket, de a játék vége felé már fordul a kocka, a túlélésre kell koncentrálnia.



22. ábra: Nehezedő játékmenet

A játék első 28 leütése a címképernyő megjelenésére van fenntartva. Be van állítva, hogy aki át akarja ugrni ezt a bevezető képet, az egy Enter lenyomásával rögtön a játéktérre lép. Az 50. leütéstől kezdődik a valódi játék, az előtte eltelt idő csak a felkészülésé.

A három fázis:

50 - 180 leütés

180 - 330 leütés

330 - 400 leütés

Két paraméter is változik az adott fázisokban. Először is van egy ciklikusidő: ennyi időközönként vizsgálja meg a program annak a valószínűségét, hogy egy adott szörnyet (vagy bónuszt) teremtsen.

Ez az úgynevezett „spawnolás”. A másik pedig egy valószínűség, amit egy függvény számol ki, a `valszamDobas`.

pl. a `valszamDobas(3)` 1/3 esélyt jelez.

Az élet spawnja nem változik a játék során. 18 leütésenként 1/3 eséllyel teremődik. Energia nélkül viszont nagyon nehéz a játék.

Egyenlőtlenséget okozna, ha csak a véletlentől függene az energiaellátás.

Így a játék 100 leütésenként ad energiát, ha létre tud jönni. Ugyanis a játék ötször próbál mindig bónuszt elhelyezni, ha ötödikre nem sikerül, vagyis foglalt a mező, akkor tovább már nem próbálkozik.

Ezen kívül a harmadik fázisban pluszban(!) 15 leütésenként egyharmad eséllyel létrejön egy energia bónusz.

Erre azért van szükség, mert a szörnyek, ha rálépnek egy olyan mezőre, amin bónusz van, megsemmisítik a bónuszt. Ebből következik, hogy ha több szörny van, nagyobb eséllyel semmisítik meg a létrejövő bónuszokat. Itt most még ki kell térnem a játék tervezésével kapcsolatban egy fontos szempontra. A játék nagyon igazságtalan lenne, ha a szörnyek a játékos tetejére, vagy közvetlenül mellé is tudnának teremtni, mert ilyen esetekben a játékosnak nem lenne esélye megóvni önmagát, ki lenne szolgáltatva a vak szerencsének. Ezért a játék vizsgálja, hogy egyrészt se egy másik szörny tetejére, se a játékos egy mező sugarú körzetében ne jöhessen létre szörny.

A négy szörnytípus:



23. ábra: SmallSaucer



24. ábra: BigSaucer



25. ábra: Mosquito



26. ábra: Bloktopus

Az első három szörny egy állandó mozgásminta alapján mozog a pályán. A Bloktopus véletlenszerűen lép a négy irány valamelyikébe. Ezen kiszámíthatatlan mozgása miatt nem egyszerű lelőni. Kis számban nem jelent nagy veszélyt a játékosra nézve. Ő az egyetlen szörny, aki nem tud lőni.

A Mosquito mozgása mindig a pálya tetejéről az alja felé történik, ezért ők csak egy irányba, lefelé tudnak lőni. Ha eléri a pálya alsó végét, akkor eltűnnek.

A két csészealj, a SmallSaucer és a BigSaucer véletlenszerűen lő a négy irány valamelyikébe. Az ő mozgásmintájuk úgy van megírva, hogy körkörös pályára álljanak egy idő után. Ők fenn és alul is megszülethetnek a felső illetve alsó két sorban.

Sem a szörnyek lövése, sem a játékos lövése nem tudja elpusztítani a bónuszokat.

Az alábbi ábrán a SmallSaucer mozgásmintája látható, ami egy 2 dimenziós karaktervektor. A pálya minden mezőjének megfeleltet egy Égtáj szerinti utasítást. A 'E'-nel északra, a 'K'-nál keletre, a 'D'-nél délre és az 'N'-nél nyugatra megy majd. Az egyes szörnyosztályok példányai tartalmazznak egy-egy ilyen táblát:

```
23 private char[,] mozgásminta = new char[9, 10]
24 {
25     {'D', 'D', 'D', 'D', 'D', 'D', 'D', 'D', 'D', 'D'},
26     {'D', 'D', 'D', 'D', 'D', 'D', 'D', 'D', 'D', 'D'},
27     {'K', 'D', 'D', 'N', 'N', 'N', 'N', 'N', 'N', 'D'},
28     {'K', 'D', 'N', 'N', 'N', 'N', 'N', 'N', 'E', 'N'},
29     {'K', 'K', 'K', 'K', 'K', 'K', 'K', 'K', 'E', 'D'},
30     {'K', 'K', 'K', 'K', 'K', 'K', 'K', 'K', 'E', 'N'},
31     {'E', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N'},
32     {'D', 'E', 'E', 'E', 'E', 'E', 'E', 'E', 'E', 'E'},
33     {'E', 'E', 'E', 'E', 'E', 'E', 'E', 'E', 'E', 'E'}
34 }
```

27. ábra: Mozcásmlnta

A program felépítését két fő részre szerveztem:

egy fő programra, amelyben a SadConsole alapvető metódusai futnak és egy MapScreen konzolra, ahol a játékot vezérlő metódusok is vannak.

A SadConsole tipikus Main függvénye:

```
static void Main(string[] args)
{
    SadConsole.Game.Create(Width, Height);

    SadConsole.Game.OnInitialize = Init;

    SadConsole.Game.OnUpdate = Update;
    SadConsole.Game.Instance.Run();
    SadConsole.Game.Instance.Dispose();
}
```

Az update a következőket vizsgálja:

-GameOver helyzet van-e

-Victory helyzet van-e

-Eltelt-e 28 leütés. Mert ha igen, akkor mindenképp eltűnik a kezdőképernyő, és a játékeretet látjuk.

-Nézi folyamatosan az update-ben, hogy lenyomunk-e egy nyilat, amire mozgatja az űrhajót, vagy Space-t, amire tüzel a játék.

-Kezeli a játékos alapanimációját és jelző animációit. (Ezeket később az animációkról szóló részben kifejtem.)

-Nézi, hogy a játékos ütközött-e szörnnyel. Ha igen, megteszi a szükséges lépéseket. Csökkenti az életerőt, és a szörnyet eltávolítja a játékból vizuálisan és logikailag is.

-Nézi, hogy a játékos ütközött-e bónusszal. Ez esetben is megteszi a szükséges lépéseket: eltávolítja a bónuszt, és megnézi, hogy energia vagy életerő bónusz volt-e, és aszerint növeli az adott értékeket.

A játékban a játéktér, melyen az események zajlanak egy 10x9-es nagyságú, téglalap alakú glyph-ekből álló rács. Erre vannak több rétegben (layerben) elhelyezve a játékos, a szörnyek és lézerek valamint a bónuszok. Később a layerek szervezését bővebben is részletezem majd.

Grafikák

A játékot SadConsole library felhasználásával implementáltam, ezért konzolban kell gondolkodnunk. A különböző objektumokat jelző sprite-okat egy fontban tároltam. A grafikák ennek segítségével jelennek meg a képernyőn és az ANSI ABC valamelyik glyph-jének felelnek meg.

```
public override void phaseChange(byte inb)
{
    if (inb == 0)
    {
        this.actualGlyph = 'F';
    }
    if (inb == 4)
    {
        this.actualGlyph = 'V';
    }
}
```



F

V

28. ábra: A Mosquito két animációs fázisa

Tehát például a Mosquito nevű szörnyem az 'F', vagy 'V' karakternek felel meg. Ez úgy lehetséges, hogy egy karaktertáblát módosítottam. Ezeket a SadConsole png fájlban és egy a tábla információit tartalmazó font fájlban kezeli:

```
{"FilePath":"chess.png","GlyphHeight":16,"GlyphPadding":2,  
"GlyphWidth":16,"Name":"chess","SolidGlyphIndex":64}
```

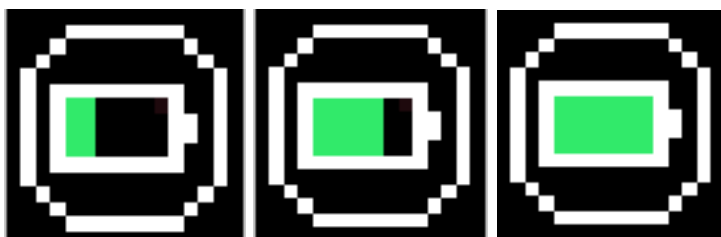
Egy eleve játék céljára template-ként összerakott fontot (chess font) módosítottam, mert az már optimalizálva volt erre a felhasználásra. Ezután az Aseprite nevű pixel editorral belerajoltam a megfelelő helyekre az animációk egy-egy fázisát.

Alapanimációk

A pályán lévő dolgok animáltak, ezért külön programrészek felelnek. Minden animáció kétfázisú, a fázisok közötti váltás az idő függvényében történik. Maga a játék méri a kezdés óta eltelt összes időt, és ebből maradékos osztás segítségével kapjuk meg azt a számot, amely bemeneti paraméterként segít megállapítani, hogy egy adott objektum milyen animációs fázisban van. Így elérhető az, hogy például, ha a játékos irányt vált, nem szakad meg az animáció, ami az ő lézerhajtóművének a pulzálása. A szörnyek ugyanígy folyamatosan két fázis között váltogatnak.

Bónusz animációk

A bónusz animációk különböznek az előbbiektől. A fázisok közötti váltás gyorsabb: leütésenként megy. Ezenkívül az energiabónusz animációja háromfázisú, és ezzel az egyetlen háromfázisú animáció az egész játékban.



29. ábra: Energia bónusz animációjának három fázisa

Jelző karakteranimációk

Készítettem egy külön animációfajta, amely a karaktert ért hatásokat jelzi az úrhajón, vizuálisan. Ezt nem megrajzoltam, hanem használtam a SadConsole lehetőségét az előtér színének beállítására.

pl.:

```
_player.Animation.CurrentFrame[0].Foreground = Color.LightGreen;
```

A színváltozás kezdeti időpontját eltárolja:

```
PLvillogkezd = _mapscreen.timeSum;
```

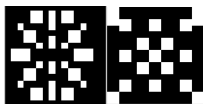
Van egy villogAnim (int INtimesum) függvény, ami, ha a kezdeti időpont és az aktuális időpont közötti eltelt idő nagyobb mint 4 leütés, visszaállítja az úrhajót az alapszínére. A függvény folyamatosan fut, benne van az update-ben, tehát minden színváltozás után visszaállít.

Jelző háttéranimációk

A jelző háttéranimációk a gamegrid nevű rácson az események háttérében jelennek meg.

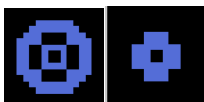
Kétféle létezik:

- Robbanás, egy szörny eltűnésekor:



30. ábra: Robbanás animáció két fázisa

- Egy koncentrikus körökből álló kék animáció, ami a bónusz felvételét jelzi:



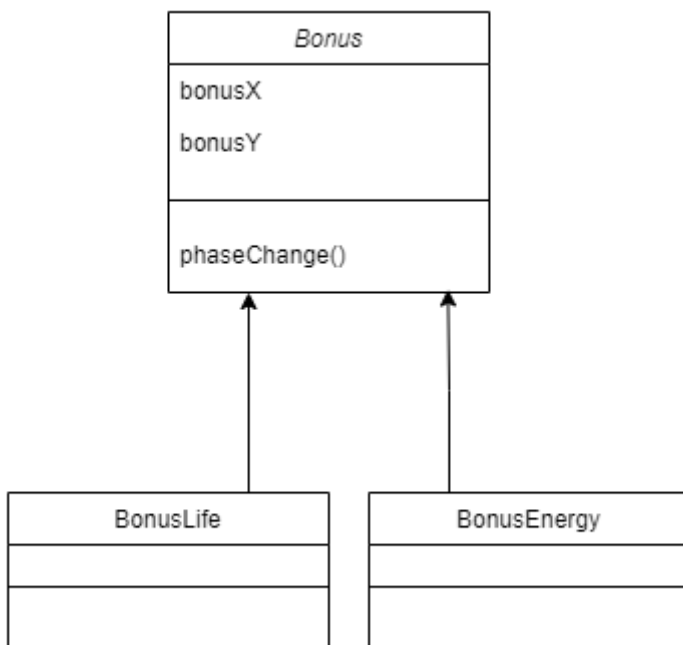
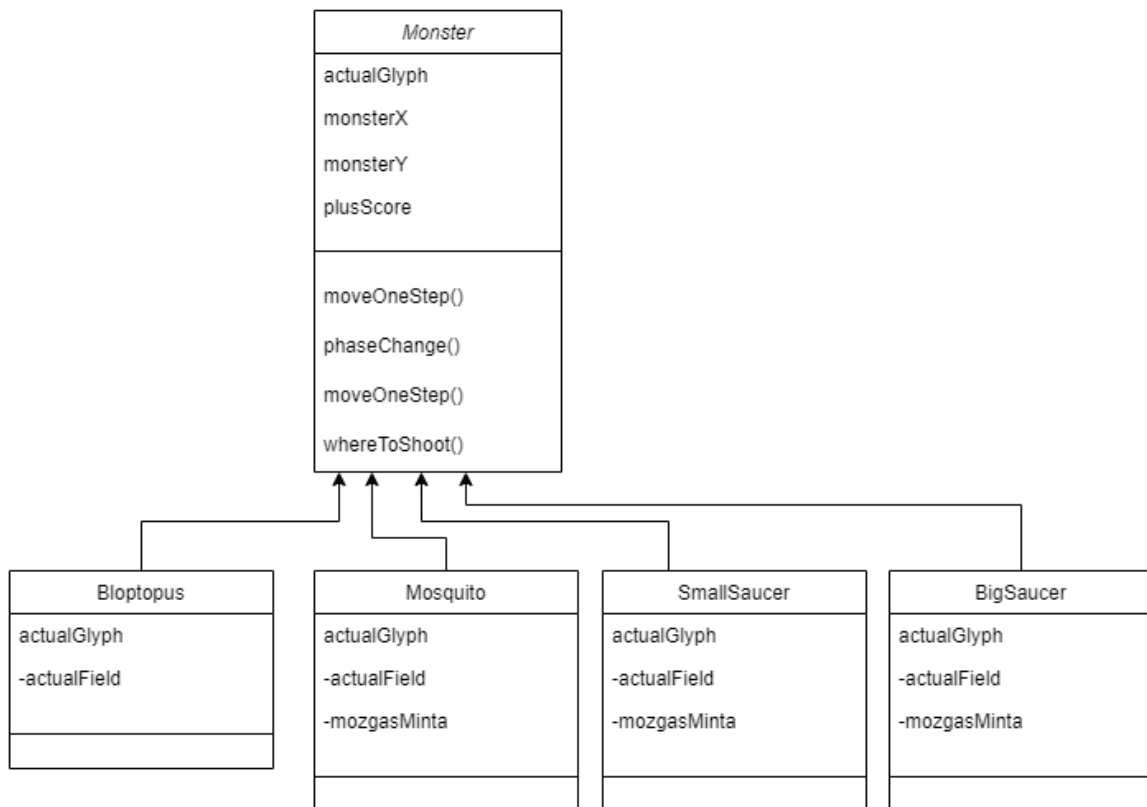
31. ábra: Powerup animáció két fázisa

Ezeket karakterként tárolja el a program a gamegrid rácson, és mikor frissíti a megjelenítést, akkor egyik fázisból a másikba vált.

pl.:

```
if (gamegrid[i, j] == 'G')
{
    this.gridConsole.SetGlyph(i, j, 'W');
    this.gridConsole.SetForeground(i, j, Color.White);
    this.gridConsole.SetBackground(i, j, Color.Transparent);
}
```

Szörnyek és Bónuszok osztálydiagramja



32. ábra: Szörnyek és bónuszok osztálydiagramja

Mint az ábrán látható a szörnyek egy Monster ösosztályból származnak, a bónuszok pedig egy Bonus ösosztályból. Erre azért van szükség, mert a listákban, ahol ezeknek a példányai le vannak tárolva, nem tudhatjuk előre, hogy melyik fajtából mennyi lesz. Az Update ciklusokban a program ezeken a listákon halad végig és hívja meg az ábrán látható függvényeket. Ezek a szörnyeknél :

- az animációra: phaseChange()
- a mozgásra : moveOneStep()
- a támadásra: whereToShoot() vonatkoznak.

Mindegyik szörnynek van:

- egy aktuális fázisa: actualGlyph
- X és Y koordinátája : monsterX, monsterY
- egy szám, amely mutatja, hogy mennyi pontot ér a levadászása: plusScore.

Mint korábban kifejtettem a négy szörnyosztályból háromnak van mozgásmintája.

Minden bónusznak van:

- egy függvénye, amely a fázisaik között vált: phaseChange().

Ezen kívül

- X és Y koordinátájuk : bonusX, BonusY property-je

A két lista:

```
public List<Monster> monsterList { get; set; }  
public List<Bonus> bonusList { get; set; }
```

A listából való eltávolításnál a végéről iterál a lista az eleje felé, hogy ne okozzon zavart az eltávolítás a folyamatban.

Úgy döntöttem, hogy a játékban a háttér fekete lesz, de az egyértelműség kedvéért a pálya sarkát jelezni szerettem volna. Ezért rajzoltam Aseprite-ban négy sarokelemet. Ezeket a playerLaserGrid-re helyeztem, nem akartam nekik külön rácsot szánni. Így a játékos lövése kitakarja a sarokelemeket, ha odaér, ezzel is mutatva, hogy az elemeknek csak jelzésértékük

van, kijelölik a pálya határát, játékot befolyásoló funkciójuk nincsen. A Játékkal kapcsolatos programrészek a főprogramban vannak, hogy elkülönüljenek a többi résztől, amelyek a leütéseknek vannak alárendelve.

A játékot kezelő MapScreen osztály tartalmazza azokat a rácsokat, amelyeken a játék objektumai megjelennek és/vagy mozognak.

A bónuszok és a szörnyek és a robbanás illetve powerup animációk a gamegrid nevű rácson vannak.

A playerLaserGrid tartalmazza a játékos lövéseit, a monsterLaserGrid pedig a szörnyek lövéseit.

A lövésekkel kapcsolatos mozgás egyszerű. Minden lövésnek van egy kezdeti iránya, és abba az irányban halad folyamatosan, ameddig el nem talál valamit vagy ki nem megy a játéktérrel.

Minden rács egy kétdimenziós karaktervektor, ahol az adott karakter egyértelműen jelzi a játékot mozgó mechanizmus számára, hogy mi a teendője.

Az alábbi kódrészlet (a renderTheGrid függvény része) szemlélteti a játéknak ezt a mechanizmusát:

```
for (int i = 0; i < 10; i++)
{
    for (int j = 0; j < 9; j++)
    {
        if (gamegrid[i, j] == '?')
        {
            removeMonster(i, j);
            gamegrid[i, j] = '0';
        }
        if (gamegrid[i, j] == '0')
        {
            this.gridConsole.SetGlyph(i, j, '0');
```

```

        this.gridConsole.SetForeground(i, j, Color.Transparent);
this.gridConsole.SetBackground(i, j, Color.Transparent);
    }

else
    {
        char actualGlyph = gamegrid[i, j];
        this.gridConsole.SetGlyph(i, j, actualGlyph);
        this.gridConsole.SetForeground(i, j, Color.White);
        this.gridConsole.SetBackground(i, j, Color.Transparent);
    }
}
}

```

Az alsó logkonzol

Az alsó logkonzol értesít a játék eseményeiről. Kezdetben az ősi mondást írja ki: "KNOW YOUR ENEMY!"(...)⁸

Utána a játék eseményeit kommentálja színes feliratok formájában. A konzol mindig a `theMessageInLog` stringet írja ki. Ez a string kap új értékeket a játék során.

⁸ Szun Vu : A háború művészete

Game Over & Victory

A játék végét egy bool típusú változó tárolja (gameover). Ha ez true, akkor jelenik meg a Game Over képernyő és áll le a játékméchanizmus. Hasonlóan működik a győzelmet (Victory) jelző kép is. Ott egy külön apró konzol kijelzi az elért pontot, amit a győzelem pillanatában a játék eltárol.



33. ábra Game Over képernyő



34. ábra Victory képernyő, a kijelzett pontszámmal

III. FELHASZNÁLÓI DOKUMENTÁCIÓ

A SZDA2021_01 számú programcsomag tartalma:

PÉNZENERGIAGENERÁTOR /Sci-Fi Edition/,

- egy feleletválasztós kvízzjáték
- valamint a SuperZekkeli /Return of the DimensionDrifter/, ami egy modern space shooter játék.

Rendszerkövetelmény

A játék Windows 10 operációs rendszeren jól működik.

Napjaink (2021) átlagos felhasználói számítógépén hiba nélkül fut.

A PÉNZENERGIAGENERÁTOR használata

A rövid intronál az Enter gomb lenyomásával a főmenü következik.

Zárójellel egyértelműen jelölve vannak az adott funkcióhoz tartozó billentyűk.

Az Info menüben a programról találhat információkat.

A Kilépés gombbal kiléphet a játékból.

A Beállítás gomb lenyomásával a Beállítás menüben a következő dolgok beállítására nyílik lehetőség:

- A kérdésekre rendelkezésre álló idő
- A kérdések maximális száma
- A kérdések nyelve (magyar/angol)
- Bekapcsolható itt a program Vágó-üzemmódja is.

A Játék gombbal új játék indul. Itt a konzolon feltett kérdésre kell megadni négy válaszlehetőség közül a helyesnek gondolt választ az adott időkereten belül. A négy gomb egyikének megnyomásával jelölheti meg választát. Ha a válasz helytelen, a játék véget ér. Ha a válasz helyes, a játék folytatódik, minden helyes válasszal, egészen a legvégső kérdésig.

A SuperZekkeli használata

A játék egy kezdőképernyővel indul, ami Enterrel átugorható.

A játék során a játékos egy űrhajót irányít, SuperZekkeli járgányát.

Egy 10x9 méretű játékmezőn négyféle irányba lehet haladni a nyíl billentyűk lenyomásával. A Space gombbal a játékos tüzel. Ez egy egységnyi energiát emészt fel.

Az energia szintjét az energia logo melletti > jelek mutatják.

Az élet szintjét a vörös pluszjel logo melletti szívecskék jelzik.

Az energia energiabónuszok, az élet élethónuszok segítségével növelhető.

A játék állásáról a játéktér alatti logkonzol tájékoztat.

A győzelmet Victory képernyő jelzi az elért ponttal együtt.

A veszteségről Game Over képernyő tájékoztat.

A játék célja: minél több pont elérése úgy, hogy mind a három támadási hullámot túléljük.

Jó szórakozást kíván a fejlesztő, Székely Dávid Attila!

IV. ÖSSZEGZÉS

Szakdolgozatom során azt a célt tűztem ki magam elé, hogy azonos formátumban és rendszerben, egy konzolos környezetben, két egymástól eltérő jellegű játékot is megvalósítsak. Egy szellemi, intellektuális és egy ügyességi játék képezi ezt a csomagot, egy 100x40 karakterből álló konzolon megjelenítve, C#-ot és Monogame-t (Microsoft XNA-ra épül) használva.

A játék során sok figyelmet fordítottam a korai kilencvenes évek esztétikáját megidéző ANSI látványvilágra és a különböző konzolos hatások elérésére. A SadConsole sok lehetőséget kínált a konzolos látványvilág színvonalas megvalósítására.

Mindkét játék igény esetén könnyen továbbfejleszthető.

A fejlesztés sikerrel zárult, mindkét játék elkészült, és minden célkitűzésemet sikerült teljesítenem. A kész játékokat sokat teszteltem, nincsenek általam ismert hibái.

V. ÁBRAJEGYZÉK

1.	ábra: Pénzenergia Generátor logo	8
2.	ábra: SuperZekkeli kezdőképernyő.....	8
3.	ábra: Pénzenergia Generátor intro.....	15
4.	ábra: A főmenü	16
5.	ábra: Az Info képernyő	19
6.	ábra: Beállítás menü.....	20
7.	ábra: Vágó-üzemmód.....	20
8.	ábra: Vágó István, ASCII art.....	21
9.	ábra: Vágó István, módosított fotó.....	22
10.	ábra: Kvízzjáték játékmenete	23
11.	ábra: „Vissza a menübe” tájékoztatás	24
12.	ábra: Formázott szöveg a konzolon.....	25
13.	ábra: Citromsárgával megjelölt gomb	26
14.	ábra: Narancssárgával megjelölt válasz	26
15.	ábra: Helyes válasz megjelölése.....	27
16.	ábra: Bal alsó számláló.....	27
17.	ábra: Alsó infokonzol	28
18.	ábra: SuperZekkeli játék közben	29
19.	ábra: Statisztikák	30
20.	ábra: Élet bónusz	31
21.	ábra: Energia bónusz	31
22.	ábra: Nehezedő játékmenet	32
23.	ábra: SmallSaucer.....	33
24.	ábra: BigSaucer	33
25.	ábra: Mosquito.....	33
26.	ábra: Bloptopus	34
27.	ábra: Mozgásminta	34
28.	ábra: A Mosquito két animációs fázisa	36

29.	ábra: Energia bónusz animációjának három fázisa	37
30.	ábra: Robbanás animáció két fázisa	38
31.	ábra: Powerup animáció két fázisa.....	38
32.	ábra: Szörnyek és bónuszok osztálydiagramja.....	40
33.	ábra Game Over képernyő.....	44
34.	ábra Victory képernyő, a kijelzett pontszámmal	44

VI. FELHASZNÁLT IRODALOM

SadConsole dokumentáció és tutorialok

<https://sadconsole.com/>

SadConsole Github

<https://github.com/Thraka/SadConsole>

The SadConsole Roguelike Tutorial Series

<https://ansiware.com/>

SadConsole rövid ismertető

<http://www.roguebasin.com/index.php?title=SadConsole>

Demoscene elmélet

<https://lodev.org/cgtutor/plasma.html>

VII. FELHASZNÁLT PROGRAMOK

Microsoft Visual Studio Community 2019

<https://visualstudio.microsoft.com/>

Aseprite

<https://www.aseprite.org/>

Moebius

<https://blocktronics.github.io/moebius/>

Gimp

<https://www.gimp.org/>

RexPaint

<https://www.gridsagegames.com/rexpaint/>

SadConsole library

<https://sadconsole.com/>