Brute Force

avagy a puszta erő nem elég

Készítette: Maráczi Marcell

Tartalomjegyzék

1. Bevezetés

Napjaink egyik legnépszerűbb szoftvertípusa a játékszoftver. Manapság a játékok szinte mindenhol ott vannak. A lakások nappaliját elfoglalták a különböző konzolok, rajtuk a különböző egyszerűbb-összetettebb programokkal, melyekben sokszor még mozgásérzékelő is van, hogy teljesen beleélhesse magát a felhasználó. Ha ez nem volt elég, út közben, várakozáskor, vagy csak egyszerű időtöltésre rengeteg telefonos játék található meg a különböző platformok saját alkalmazásboltjaikban, ahol naponta rengeteg új szoftver lesz elérhető hála a hobby fejlesztőknek. Ezek többsége rendkívül egyszerű, például különböző ritmus-reakcióidőt próbáló programok, viszont vannak bonyolultabbak is, mint amiben csúzliból lövünk madarakat a gonosz malacokra, egészen a rendkívül összetett 3D-s alkalmazásokig, amik nem egy esetben egy már korábban asztali számítógépes játék kis mértékben leegyszerűsített változata.

Valamint természetesen ott van a játékok „ősplatformja”, az asztali számítógép. Az első videojáték a 20. század közepére tehető, és a műfajnak a sikere azóta is töretlen, mai napig is játékok tömkelege jelenik meg minden évben, hála a nagy kiadóknak, és a kis költségvetésű, úgynevezett „indie” fejlesztőknek, akiknek inkább a szórakoztatás a céljuk, míg a kiadóknak sajnos sokszor csak a pénz számít, és ez a játékélmény rovására megy.

2. Játékfajták

Az elmúlt hatvanöt évben többfajta, egymástól sokszor nagymértékben eltérő játéktípusok alakultak ki, függően attól, hogy mi a programban az elsődleges feladata a játékosnak. Ezek pedig az alábbiak:

2.1. FPS

First-Person Shooter, vagy belső nézetű lövöldözős, talán a legnépszerűbb a játékok között. A monitoron ilyenkor azt látjuk, amit a játékbeli hősünk, akit irányítunk, a „karakterünk” is lát, az ő szemszögéből kell meghoznunk döntéseinket, mozdulatainkat. Ugyan a nevéből az adódik, hogy csak lövöldözős, például háborús játék tartozhat ide, de ide értünk minden olyan játékot, amiben első személyből látjuk a világot. Ilyen például a Call of Duty sorozat, valamint a Counter-Strike sorozat.

2.2. TPS

Third-Person Shooter, külső nézetű lövöldözős, nagyrészt ugyan az, mint az FPS, azzal a különbséggel, hogy a képernyőnkön látjuk az egész karakterünket, egy, a háta mögött lebegő kamera nézőpontjából. Természetesen ebben a nézőpontban sem szükség az, hogy lőni kelljen. Sokszor, ha a játék szabályai nem teszik szükségessé az egyik kameranézethez való rögzülést, egy gombnyomással, vagy a program beállításai között lehetőség van rá, hogy a felhasználó az FPS és a TPS nézet között egyszerűen váltson. TPS például a Prince of Persia sorozat, és a Grand Theft Auto széria is, az első részeit leszámítva.

2.3. Platformer

Általában 2D-ben játszódik, a játékosnak a karakterével különböző ügyességi feladatokat kell teljesíteni, végigugrálni a pályán illetve legyőzni az ellenfeleket. A kamera az esetek többségében a hőssel együtt mozog, a pálya egy szeletét mutatja minden esetben úgy, hogy fel lehessen készülni a közelgő feladatokra. A korai játékok a technikai „fejletlenségnek” hála szinte csak platformer volt, nem volt meg a lehetőség a nagyobb látványvilág megalkotására, de ez nem akadályozta meg a fejlesztőket a

máig híres alkotások létrehozásától, hisz ki nem hallott volna a Super Mario című játékról, és annak további részeiről.

2.4. RTS

Real-Time Strategy, valós idejű stratégia, felülnézetes, egységek irányítása a feladat a cél elérése érdekében. A játékos általában építhet különböző épületeket, ahonnan új egységeket szerezhet, vagy a már meglévőknek bónuszt. RTS a ’90-es évekből az Age of Empires és a Warcraft.

2.5. Istenszimulátor

Legtöbb esetben egy várost kell benne felépíteni, azt fejleszteni, politikai-gazdasági kapcsolatok kiépítésével életben tartani. Ha nem megfelelőek az életkörülmények, az emberek akár meg is halhatnak. Bizonyos programokban előfordulhatnak különböző katasztrófák, amik a játékmenetet befolyásolják, például egy földrengés miatt kettéválik a városunk alatt a talaj, s vele a város, vagy egy hurrikán lerombolja a házakat.

Természetesen egyéb, másfajta játékok is léteznek, de véleményem szerint ezek a legnépszerűbbek, a boltok polcain is nagyrészt a felsorolt típusokat találnánk meg.

3. Specifikáció

Mint azt már említettem, manapság a játékok mindenhol ott vannak. Ezért döntöttem úgy, hogy szakdolgozatom gyanánt én is egy játékprogramot készítek el. Véleményem szerint a jövőben a játékipar csak növekedni fog, így játékfejlesztőkre a 21. század folyamán végig szükség lesz, így ez a program munkakeresés közben is nagyon hasznos példamunka lesz a tudásomról, képességeimről. Játékomat Unity3D fejlesztőkörnyezetben készítettem el a Microsoft által fejlesztett C# programozási nyelven, fő célplatformként Windowsra, azon belül is Windows 7-re és Windows 8.1-re, de nincs kizárva a kompatibilitás más Windows verziókkal sem, egyetlen feltétel, hogy .Net futtatókörnyezet 3.0 verziója, vagy újabb telepítve elérhető legyen a számítógépen. A program kevés módosítással fordítható lenne Apple és Linux platformokra is, viszont mivel Windowst tűztem ki célplatformnak, csak erre készítem el.

3.1. Szoftver alapvető jellemzői

- Szoftver elnevezése, címe: „BruteForce, avagy az erő nem elég”
- Szoftver kategóriája: Logikai játék
- Szoftver alapvető funkciója: A játék folyamán a cél, hogy a játékos, vagy játékosok a játékot jelentő labirintuson átjussanak, kikerülve vagy megoldva az esetleges csapdákat, elmenekülve a „szörny” elől.
- Fejlesztői környezet: Unity3D, MonoDevelop, Microsoft Visual Studio 2013
- Programozási nyelvek: C#
- Platform: Microsoft Windows 7, 8.1

3.2. Szoftver funkciói

Egy- vagy kétjátékos módú labirintus játék, melyben a fő cél az, hogy a játékos a labirintus bejáratától eljusson annak kijáratáig. A pálya minden játékindításkor véletlenszerűen kerül legenerálásra, így biztosítva, hogy a játékélmény ne legyen repetitív a pálya ismétlődésétől, valamint lehetőséget adva, hogy a legkülönfélébb méretű pályákon is lehessen játszani, természetesen az észszerűség határain belül. A pályagenerálás

szabályos labirintust generál, amiben nincsen „kör”, viszont a játékélmény fokozása érdekében a labirintus mérete alapján további utakat hozok létre, hogy könnyebb legyen a kiutat megtalálni.

A program első személyű, 3D-s ügyességi játék. Karakterünket a WASD billentyűkkel irányítjuk, valamint az egér használatával tudjuk a kameranézetet módosítani. Két fő játékmód áll a felhasználó rendelkezésére, egyjátékos mód, illetve kétjátékos mód. További opciók, hogy lehet időkorlátot engedélyezni, így a megmaradt idővel a játékos felkerül az eredménytáblára, amit később a főmenüből megtekinthet, hogy bent van-e a legjobb játékosok között. A játék folyamán a kép egy külső ponthoz viszonyított ponthoz való távolság alapján teljesen elsötétedik, illetve normál szintre kivilágosodik.

Egyjátékos módban ez a külső pont a célban, a labirintus másik végén található, ahogy haladunk, úgy lesz egyre nehezebb látni (természetesen ez teljesen letiltható, amennyiben a játékos csak egy egyszerű labirintusjátékra vágyik), a végén szinte teljes sötétségben botorkálva. Lehet engedélyezni csapdákat, ezek úgy működnek, hogy ha a játékos rálép egy mezőre, akkor a kezdőhelyről elindul egy szörny, aki ha utoléri a karaktert, vége a játéknak, a felhasználó elbukta a próbálkozást. A szörny minél tovább él, annál gyorsabban követ, így lényegében egy további időkorlát az alap időkorlátos mód mellé. A játéknak akkor van vége, ha a játékos beér a célba, ekkor választhat, hogy visszatér a főmenübe, új, ugyanekkora pályát kér, vagy pedig egy nehezebbet.

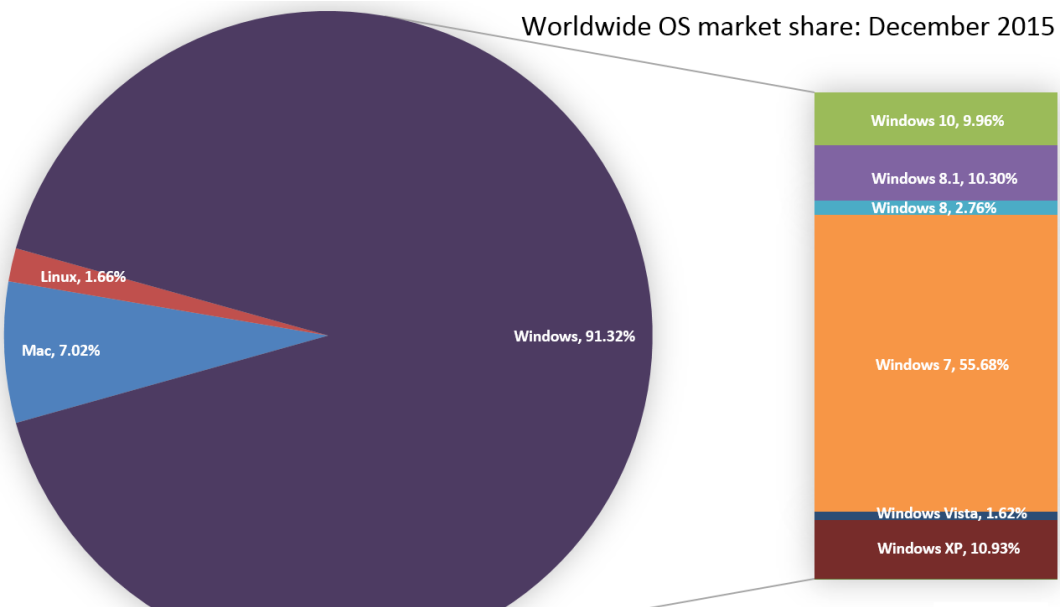
Kétjátékos módban mindkét játékosnak a sötétséghez használt viszonyítási alap a másik játékos helyzete. A felhasználók nem látják egymást, teljesen más labirintusban járnak, azok „egymás felett” helyezkednek el. Kétféle csapda lehetséges, egyik ugyan az, mint az egyjátékosban, itt a szörny csak azt üldözi, amelyik felhasználó a csapdába lépett. Másik csapda lehetőség esetén, ha az egyik játékos rálép a csapdára, a falak bezáródnak körülötte, ő azt a mezőt nem tudja elhagyni. Ekkor egy időzítő is elindul, ami ha úgy jár le, hogy a csapda még nincs „kikapcsolva”, a játékos megfullad, meghal,

és elbukták a próbát. A másik játékos feladata, hogy megkeresse ugyan azt a mezőt a saját labirintusában (számára ez eltérően néz ki a többihez képest) és rálljon ugyanarra a pontra, így kiszabadítva társát és leállítja az időzítőt. Természetesen aktivált csapda esetén az összes többi csapda le van tiltva, így nem fordulhat elő, hogy mindkét játékos rabságba kerül, kivéve nehéz fokozaton, ahol akár mindkét játékos is bele sétálhat a kelepcébe. A menetnek akkor van vége, ha mindkét felhasználó sikeresen beér a célba.

4. Alkalmazott technológiák

4.1. Windows

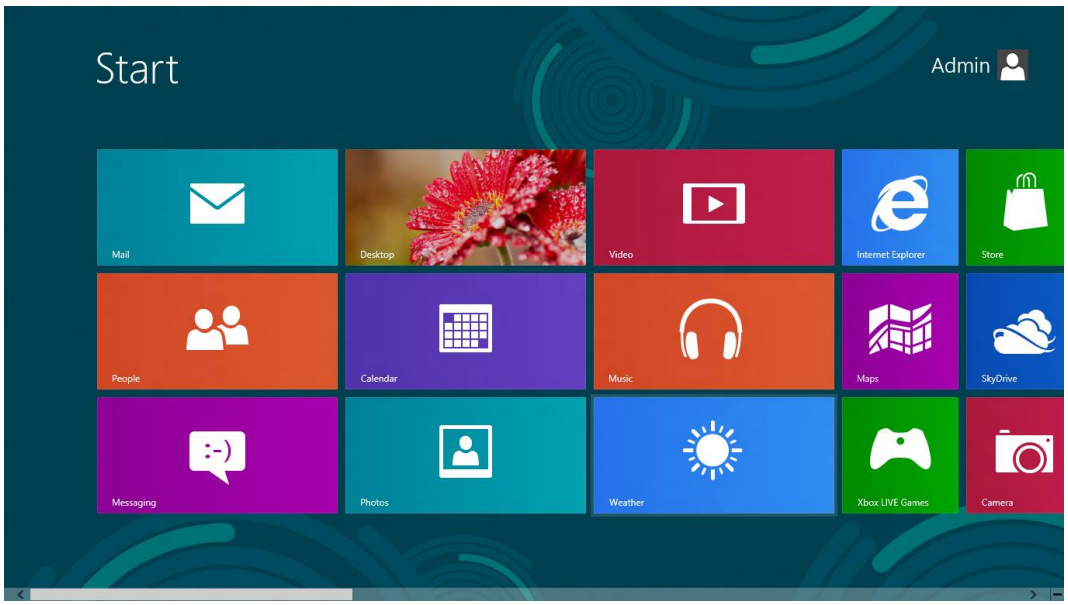
Asztali számítógépes operációs rendszerek között a Microsoft operációs rendszercsaládja, a Windows rendkívüli többségben van a többi, szintén asztali számítógép operációs rendszerekkel szemben, mint ahogy ez az 1. számú ábrán is látható.



1. ábra

Windows a térhódítását 1985-ében kezdte meg a Windows 1.0-val. Ez volt az első többfeladatos rendszer, ami hatalmas újítás volt a DOS időszaka után. Mai szemmel nézve nem tudott semmit sem, ám akkor egyszerre kezelni a fájlokat, naptárat használni, vagy telekommunikációs programot

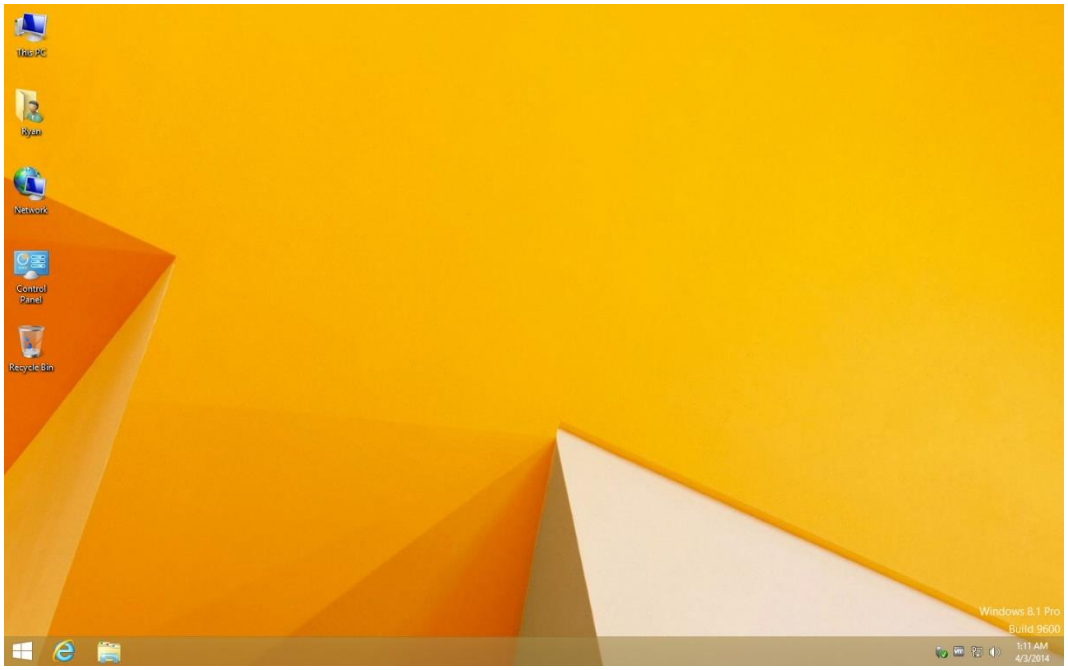
használni nagy megkönnyebbülés volt, hogy nem kell teljesen kilépni a programból. Windows 3.11 verzióig bezárólag MS-DOS alól kellett a Windows mint programot elindítani, azonban a Win95-től kezdődően önálló operációs rendszerként üzemel, nem szükséges hozzá a külön futtatókörnyezet. Több különböző, sikeres és kevésbé sikeres verzió után 2009-ben a Redmondi székhelyű vállalat kiadta a 7-es verziót, ami az asztali számítógépek körülbelül 50%-án a mai napig üzemel. Szinte minden szempontból nézve javult elődjeihez képest. Közvetlenül ugyan a Vista nevű verzió előzte meg, ám annyira instabil rendszerként sikerült kiadni, hogy sokan úgy tekintenek rá, mint ami nem is létezik, ezért elődként az XP-t értem. Számos új funkcióval bővült, egy közüle az optimálisabb kezelése a többmagos processzoroknak, ami a gyártás, valamint a játékipar fejlődésének nagymértékű meghajtó rugója volt. Lehetőséget adott számításigényesebb játékok megalkotására. 2012-ben megjelent a 8-as verzió, nem tekinthető sikeresnek. Az első verzió óta megszokott Start-menü teljesen ki lett véve a rendszerből, azt a négyzetes Metro felület helyettesíti, ami a 2-es ábrán látható.



2. ábra

Népszerűtlensége annak köszönhető, hogy a felület elsősorban érintőkijelzős bevitelre lett optimalizálva, kényelmetlenné téve ezt a

felhasználók nagy részének, akik a hagyományos monitor+egér összeállításhoz vannak hozzászokva. Megjelenése után megjelent a Windows 8.1, ami sikeresebb lett a közvetlen elődjéhez képest. Ugyan a Start menü teljesen nem jött vissza, azonban a felhasználók kaptak egy gombot a helyén, ami a Metro felületre visz.



3. ábra

A rendszer több optimalizáláson is átesett, ám ez sem volt elég, hogy a Windows 7 felhasználótáborát elcsábítsa. Ezután a Microsoft kihagyván a 9-es verziót, 2015 nyarán kiadta a Windows 10-et. Terveik szerint ez az utolsó Windows. Több platformon is elérhető (telefon, tablet és asztal számítógép), így biztosítva a kompatibilitást. Néhány évente megjelenő új verzió helyett ez lenne az, amit folyamatosan frissítenek, így egyszerűsítve a számítógép használatot, egy egységes rendszerrel, ami mindenhol ott van, és ugyan úgy működik.

4.2. Unity 3D

A szakdolgozatomként elkészített játékprogramomat a Unity 3D nevű 3D-s grafikus fejlesztőkörnyezetben készítettem el. Többek között azért esett erre a környezetre a választásom, mivel non-profit alkalmazások készítésére ingyenesen használható. Támogatja a manapság az „egyszerűségük” miatt népszerű objektum-orientált programozási nyelveket, az Oracle Java-t és a Microsoft .Net alapú programozási nyelvét, a C#-ot. Számomra ez tökéletes volt, mivel a programozási nyelvek közül a legjobban a C#-ban mélyítettem el a tudásom, így egyértelmű, hogy rá esett a választásom. A program kezelése egyszerű, könnyen lehet benne tesztelni a saját fejlesztéseket, futás idő közben lehet benne a program bizonyos részein módosítani, így megkönnyítve a hibakeresést és a kódoptimalizálást. Grafikus megjelenítője a fejlesztés közben látszik, ezért könnyedén lehet például a pályaelemeket is kézzel elhelyezni, ha azoknak úgy látom szükségét. Támogatja a népszerű forrás fájlformátumokat, emiatt könnyű a program grafikai elemeit is elkészíteni, nem kell egy konkrét képszerkesztő vagy modellezőprogram használatát külön megtanulni, hanem használhatjuk tovább az eddig is használtakat.

4.3. Adobe Photoshop

Grafikus tervezőprogram tekintetében az amerikai Adobe vállalat Photoshop nevű termékére tettem a választásom, több okból is. Egyik elsődleges szempont az, hogy az iskola számítógépein elérhető, ezért bármikor tudom azt használni, mikor szükségem van rá. A másik szempont az, hogy ez a program talán a legsokrétűbb grafikai szerkesztő. Minden funkció elérhető benne, amire az embernek szüksége lehet, azok használata sem bonyolult. Kezelőfelülete eléggé letisztult, a parancsok egy része bármikor előhozható a helyi menüvel, valamint a parancsok a menüsorból is elérhetők természetesen. A kész képfájlokat sokféle formátumban menthetjük,

ezeknek nagy részét kezeli a Unity 3D is, valamint képes az alkalmazás az egész projekt mentésére is, ekkor a munkát később onnan folytathatjuk, ahol azt korábban abbahagytuk.

4.4. Autodesk 3ds Max

3D tervező program, más néven modellezőprogram tekintetében a méltán népszerű 3ds Max nevű szoftverre esett a választásom. Sajnos az iskolában nem érhető el, viszont otthonra tanulóként tökéletes választás, mivel diákok számára van lehetőség, hogy ingyenes licenszkulcsot igényeljenek az Autodesktől. A programban a legkülönfélébb, a lehető legsokszínűbb alakzatok elkészítésére van lehetőségünk. A programban lehetőségünk van karakterek, pályaelemek, pályák készítésére, igazából bármit meg lehet benne modellezni, csak idő és türelem kérdése. Maxban egy pálya alapjainak elkészítése – ez alatt a falakat, valamint azokon az átkelőhelyeket értem – körülbelül egy óra alatt elkészíthető, míg azok elkészítése Unity Editorban több órát is igénybe vehet, mire úgy néz ki, ahogy azt elképzeltük. Nagyon hasznos tulajdonsága, hogy a benne készült modelleket natívan támogatja a Unity, sőt, még a projekt mentését is képesek vagyunk betölteni benne, így nem szükséges külön exportálni, ezáltal a későbbi szerkesztések is egyszerűbbek, mivel a fájlokat csak meg kell nyitni, és módosíthatóak, nem kell több példányt fenntartanunk különböző formátumokkal.

4.5. Visual Studio/MonoDevelop

A játékhoz készült scripteket a Microsoft Visual Studio-ban, illetve a MonoDevelop nevű fejlesztőkörnyezetben írtam C# nyelven. C# programozás esetén egyértelmű választás a VS, mivel a C# nyelv megalkotója a Microsoft, így egyértelműen teljeskörűen garantált a helyesen megírt kód működése, mivel a program fejlesztőik a nyelv minden részletét ismerik. Ezért is jó, hogy az iskolának van licensze a programhoz. Sajnos otthonra nem ingyenes a VS, viszont létezik helyette ingyenes

alternatíva, a MonoDevelop, ami akár a Unity telepítése közben azzal együtt feltelepülhet. Ez a program is teljesen működőképes, bárki számára szabadon elérhető és használható, akinek a Visual Studio nem megoldható.

4.6. Inno Setup

Ingyenes program, ami lehetővé teszi, hogy alkalmazásunkhoz telepítőt készítsünk. Használata rendkívül egyszerű, feltelepítése után egyből elkészíthetjük a telepítőnket. A folyamaton egy varázsló vezeti végig a felhasználót, melyben kijelölhetjük a szükséges fájlokat, megadhatunk egy licenszszerződést, amit a felhasználónak el kell fogadnia, ikont rendelhetünk a programunkhoz. Alkalmazásunkat a „Program Files” mappába telepíti alapértelmezésből, és létre hoz egy asztali parancsikont is hozzá.

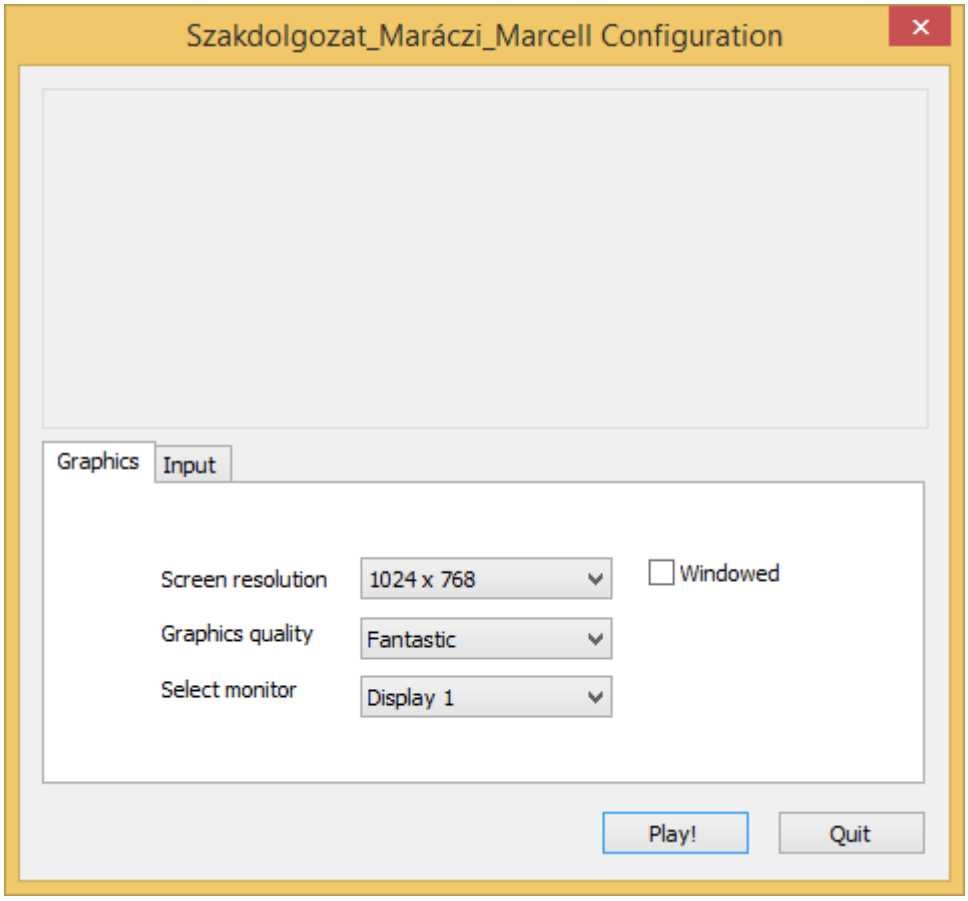
5. Felhasználói dokumentáció

5.1. Telepítés

A játék több fájlból áll, viszont a könnyebb hordozás érdekében rendelkezésre áll egy telepítő fájl, egy „.exe”. Ezt a fájlt futtatva elindul egy telepítő varázsló. A telepítéshez rendszergazdai jogok szükségesek. A licenszszerződés elfogadása után a varázsló lépéseit végigkövetve kiválasztatjuk, hogy hova települjön a játék, valamint eldönthetjük, hogy kívánunk-e asztali parancsikont létrehozni. Miután befejeződött a telepítés, az állományok futtatásra alkalmasak, a játék az asztali parancsikonnal, vagy a program mappájában található „BruteForce.exe” nevű állománnyal indítható. A futtatáshoz szükséges, hogy a számítógépre fel legyen telepítve a Microsoft .Net keretrendszer 3.0 verziója, vagy annál újabb. Első futtatáskor az eredménytábla üres, mivel csak helyi eredményeket tárol, így azt feltölteni a játékos feladata a labirintus teljesítésével.

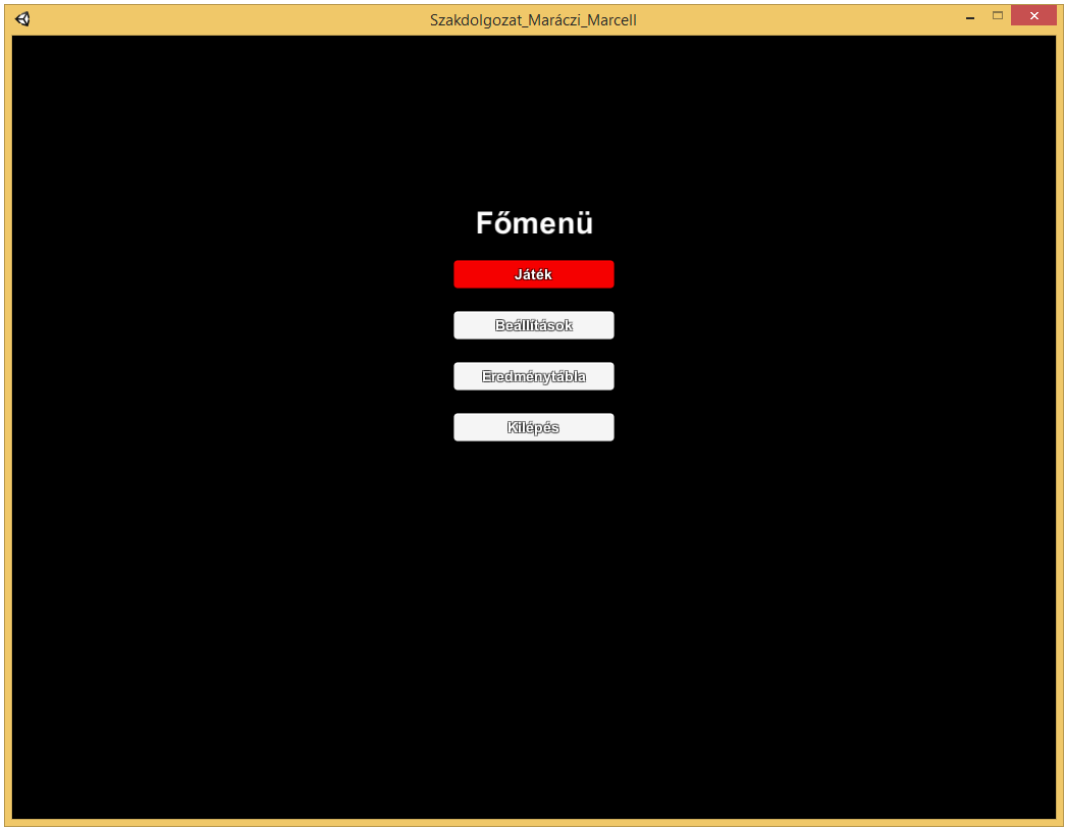
5.2. Játék főbb jellemzői

A játék minden indításkor lehetőséget ad felbontás választására, erre futtatás közben a játékosnak nincs lehetősége, ha meggondolja magát, újra kell indítani a játékot.



4. ábra Induló ablak

Itt kiválaszthatja a számára optimális felbontást, valamint dönthet arról, hogy ablakos, vagy teljes képernyős módban kíván játszani. Ezután a „Play!” feliratú gombra kattintva elindul a játék. A játék főmenüjéből lehetősége van új játékot kezdeni, játékmenetre vonatkozó beállításokat megváltoztatni, megtekinteni az eredménytáblákat, illetve kilépni a programból. Menüpontok közül egérrel, fel-le nyilakkal, illetve a „W” és az „S” billentyűkkel választhat.



5.3. Beállítások

5. ábra Beállítások

A főmenüből a „Beállítások” menüpontban van lehetőség a játékmenetet szabályzó beállítások megváltoztatására. A nehézség csúszkával

lehetséges a pálya méretét kiválasztani, a megadott érték minél nagyobb, annál nagyobb lesz a pálya. A csúszka fölött látható szám a kiválasztott nehézséget jelöli.

A szörny csapda jelölőmező megjelölése esetén a pályán lesznek mezők, amikre ha rámegy a játékos, egy szörny fogja a játékost üldözni, amíg utol nem éri, így veszítve, vagy pedig a célba éréssel megnyeri a menetet. A labirintusban több szörnycsapda is lehetséges, mennyisége a nehézségtől függ. Amennyiben több csapdát talál a játékos, annyi szörny lesz mögötte, ahány csapdába belesétált. Ezek az idő múlásával egyre gyorsabban fognak haladni, így ösztönözvén a játékost a haladásra.

Kelepce opció csak hálózatos, kétjátékos módban lép életbe, ezek a szörnycsapdákhoz hasonló fenyegetések. Ha valamelyik játékos belesétál egy kelepcébe, az utak bezáródnak körülötte, képtelen kijönni a csapdából. A partnerének feladata, hogy a saját labirintusában megkeresse ugyan azt a mezőt, és ráálljon. Ez a feladat nem lehetetlen, számára az a mező feltűnő színűen világít. A kelepcébe került játékos életereje folyamatosan fogy, amíg ki nincs szabadítva, ez a sebességére van kihatással, lassabban képes csak haladni, rosszabb időeredményt képes így csak elérni, illetve nem tud könnyedén elmenekülni az esetleges szörny elől.

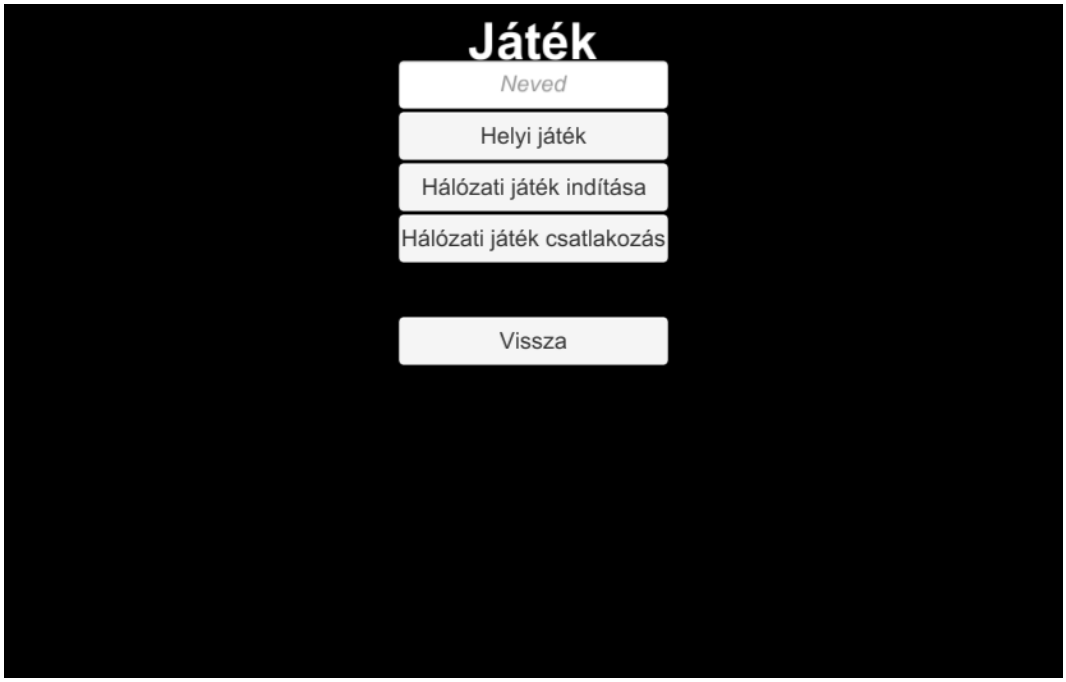
Veszély mód csak akkor választható, amennyiben a Kelepcék engedélyezve vannak. Ilyenkor előfordulhat az, hogy a játék sikertelensége garantált. Kelepce esetén, ha aktív egy kelepce, akkor a másik játékos számára ki vannak ezek kapcsolva, így nincs akadályozva a partnerének segítésében, viszont veszélynél továbbra is aktívak maradnak. Előfordulhat, hogy mindketten bezáródnak, ekkor a játéknak vége. Amennyiben a felhasználók időre kívánnak játszani, úgy a pályaméret függvényében egy visszaszámlálás indul, amint elhagyják a kezdőhelyet. Amennyiben ez az idő letelik, a játéknak vége, a felhasználó vesztett.

5.4. Eredménytábla

Eredménytábla menüpontba belépve kettő választási lehetőség áll előttünk, egyik az egyjátékos eredmények, másik pedig a hálózatos, kétjátékos mód

eredményei. Mindkettőben ugyanaz a felület, a legjobb 10 eredményt lehet megtekinteni a teljesítő játékos(ok) nevével, illetve az elért pontjával, valamint elérhető egy vissza gomb. A pálya mérete és a rákalkulált időhöz képes minél gyorsabban lett teljesítve, minél több nehezítő funkció lett engedélyezve, annál több pontot lehet elérni.

5.5. Játék



6. ábra Játékmenü

Játékot a főmenü Játék menüpontjának kiválasztásával lehet kezdeményezni. Ekkor megjelenik a Játékmenü, ahol több opció áll a felhasználó rendelkezésére. Első lépésként célszerű egy nevet beírni, ez az eredménytáblán fog szerepelni. Ezután a játékos választhat, hogy hogyan kíván játszani.

Helyi játék esetén a játékos a pálya legenerálódása után rögtön bekerül a játékba. Amennyiben időre játszik, úgy az óra azután indul, hogy a kezdőmezőt elhagyva átlép a szomszédos mezőre. Amennyiben a szörnyek engedélyezve vannak, a csapdák élesek, de azoktól a pálya első 10%-ában nem kell tartani, utána viszont egyre gyakrabban fordulhatnak

elő. A célhoz egyre közelebb kerülvén egyre sötétebb lesz a játék, így nehezítve a haladást.

Hálózatos játék esetén az egyik játékos indítja a játékot, a másik pedig az indított játékba fog becsatlakozni. Első lépésként mindkét felhasználónak a saját számítógépén engedélyezni kell, hogy a szoftver hálózati kommunikációt végezzen. Ezt megteheti úgy, hogy kikapcsolják a számítógép tűzfalát, viszont ez nem ajánlott, illetve a tűzfalszabályok kivételei közé fel lehet venni vagy a másik számítógép elérését, így gondtalanul biztosított a kommunikáció, illetve felvehető kivételnek a játékprogram indítófájlja is, a „BruteForce.exe” is. Ez esetben a játék más számítógéppel is képes a jövőben kommunikálni. Természetesen, amennyiben a felhasználó megfelelő jogosultságokkal rendelkezik a számítógépen, az operációs rendszer, a Windows is felkínálja a lehetőséget a kivétel felvételére, amikor hálózaton szeretne kommunikálni a program, ha a rendszer alapértelmezett beállításokkal rendelkezik.

Először egyik játékosnak indítania kell egy hálózatos játékot. Miután rákattintott a menüpontra, a játéka várakozik, játszani még nem lehet, csak kilépni. A másik játékos a saját számítógépén ezután rákattint a „Hálózati játék csatlakozás” gombra. Ekkor megjelenik egy beviteli mező, ahová a másik számítógép hálózati címét, úgynevezett IP-címét kell beírnia. Ha a beírás megtörtént, rákattint a csatlakozás gombra.

Amennyiben a felhasználó a helyes hálózati azonosítót adta meg, és a tűzfalszabályok is helyesek, a két számítógépen levő programok között a kapcsolat létrejön, mindkettő játékos számára megjelenik a saját, egyedileg generált labirintusuk, és elkezdődhet a játék. A beállításokban kiválasztott opciók annál a játékosnál számítanak, aki indítja a menetet, ide az összes opciót értve, de a másik játékosra is vonatkozik.

Ahogy egyre távolabb kerül egymástól a két játékbeli karakter, úgy lesz egyre sötétebb, így bátorítva a csapatmunkát, az együtt előre haladást. A csapdák és a kelepcék a kezdéskor rögtön léteznek, viszont a kelepcék csak akkor börtönzik be a játékost, ha a másik játékos számára létezik olyan

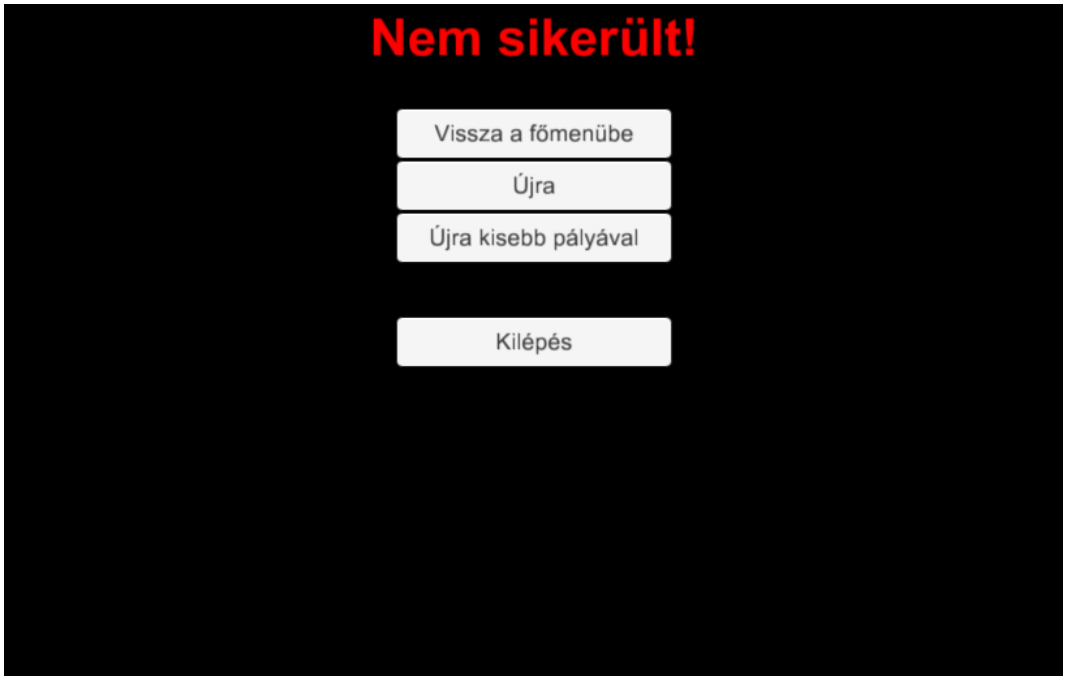
útvonal, amelyen világosban el tud jutni a másik játékos mezőjére. A két játékos számára a labirintus teljesen más, viszont vannak egyszerűsítő levágások mindkettőben, hogy együtt lehessen haladni.

Amennyiben az egyik játékos beaktivál egy szörny csapdát, a szörny csak őt üldözi. Ha az egyik játékos kelepcébe kerül, partnerének feladata, hogy kiszabadítsa. „Játékos1” aktiválja a kelepcét, bezáródnak körülötte a falak, nem tud sehova sem menni. Életereje a bent töltött idővel arányosan csökken, „gyengül”. Amennyiben túl sokáig van bent, teljesen legyengül, a menetnek vége, a párosnak nem sikerült kijutnia. „Játékos2” ez idő alatt megpróbálja megkeresni a párjához tartozó mezőt, ha sikerül, Játékos1 kiszabadul, és folytathatják tovább a kiút keresését.

Az igazi kihívás akkor jön létre, ha az egyik játékos úgy kerül kelepcébe, hogy egy szörny üldözi, mivel így a szörny a kelepce falánál fog várakozni, ha eléri azt, és kiszabadításkor azonnal lerohanja a frissen kiszabadult játékost.

5.6. Játék vége

A játéknak kettő vége van. Egyik eset az, mikor valami oknál fogva nem sikerül kijutni a labirintusból, a másik pedig az, amikor sikeres a kijutás, a játékos vagy játékosok sikeresen megmenekültek az útvesztőből.



7. ábra Nem sikerült

Amennyiben a játékos veszít, a 7. ábrán látható kép jelenik meg. Ez akkor történhet meg, ha a játékost utol éri a szörny, aki azonnal elveszi annak életerejét, így azonnali vesztést okozván. Megtörténhet akkor is, ha időre játszott, és kifutott a megadott időkeretből. Ez az időzítő akkor indul be, ha a játékos kimegy a kezdőpontról, így nem történhet meg, hogy lejár az idő, amíg a játékos nem kezdi el az útvesztő felfedezését. Harmadik lehetőség, ha kétjátékos üzemmódban a kelepcében ragadva a játékos életereje elfogy, mivel a társának nem sikerült időben kiszabadítania. A menet sikertelennek minősül akkor is, ha a két számítógép között hálózati probléma lép fel, vagy az egyik fél menet közben kilép a programból, illetve leáll a számítógépe. Ilyen esetekben a 7. ábrához hasonló állapot fogadja a még játékban levő játékost/játékosokat azzal a különbséggel, hogy az „Újra” és az „Újra kisebb pályával” gombok le lesznek tiltva, hiszen nem érhető el a partner számítógépén futó program a kapcsolat megszűnését követően. A kapcsolódási probléma megszűnése után a főmenüből új játék indítható számukra. Egyéb esetekben a 7. ábra látható, kivéve a kétjátékos módban, ahol az újra kisebb pályával menüpont nem érhető el.

„Vissza a főmenübe” gombra kattintva a játékos visszakerül a főmenübe, hálózatos módban ezzel megszakítva a kapcsolatot, mindketten a főmenübe kerülnek.

„Újra” opció kiválasztásánál egy ugyanakkora méretű, de teljesen új labirintus kerül a felhasználók elé. Társas módban ez mindkét fél számára elérhető, hiszen beállítást nem változtat, enyhe szabadságot adva a másodlagos játékos számára is.

„Újra kisebb pályával” csak egyjátékosban és kétjátékos esetén annál, aki indította a menetet érhető el. Ilyenkor a nehézség eggyel csökken, és egy másik útvesztő kerül legenerálásra a játékosok számára.

„Kilépés”-re kattintva az alkalmazás bezárul, hálózatos játék esetén a partner visszakerül a főmenübe.

8. ábra Sikerült

Amennyiben sikerül az útvesztőből az összes játékosnak kijutnia, sikeresnek tekinthető a próbálkozás. Ekkor a fenti 8. számú ábrához hasonló kép tájékoztat a sikerességről.

A pontszám számítása pozitívan figyelembe veszi a kiválasztott nehezítő tényezőket, mint hogy időre kell játszani, szörny csapdák és kelepce engedélyezése, illetve, hogy mennyire sikerült gyorsan átverekedni az útvesztőn. Negatívan hat ki a megtalált csapdák száma, illetve az összes elvesztett életerő.

A gombok ugyanazokat a szerepeket töltik be, mint sikertelen próbálkozásnál, azzal a különbséggel, hogy kisebb pálya helyett nagyobb pályán való próbálkozásra van lehetőség.

6. Fejlesztői dokumentáció

Alkalmazásom fejlesztése közben elsődleges célkitűzésem volt, hogy megismerjem a Unity3D fejlesztőkörnyezetet, annak adottságait és lehetőségeit. Ennek is köszönhető, hogy a játékprogramom többféle feladatod lát el, nem pedig egy nagy összetett alkalmazás az egész. A szakdolgozat fejlesztése közben szerzett ismeretek és tapasztalatok a Unityben és a különböző szerkesztőprogramokban lehetővé teszik, hogy a jövőben bonyolultabb, összetettebb alkalmazást is könnyebben készíteni tudjak.

6.1. Első lépések

Legelső lépésként a főmenüt készítettem el. Ez alapból nem okozott volna nagy problémát, mivel a Unity beépítetten támogatja a gombok közötti váltást a billentyűzetről. A probléma akkor jelentkezett, amikor almenüket hoztam létre. A külön menüim külön canvason helyezkednek el, menüváltásnak az aktuálisat letiltom, és azt a menüt, amire át akarok váltani, engedélyezem. Viszont így a Unity nem tudta, hogy melyik a következő gomb, így ezt kóddal kellett megoldani.

```
    public void SetToDefacultMenuPoint()
    {
        switch (currentMenu)
        {
            case menütype.Főmenü:

EventSystem.current.SetSelectedGameObject(játék.gameObject);
                break;
            case menütype.Játék:

EventSystem.current.SetSelectedGameObject(újjáték.gameObject);
                break;
            case menütype.Beállítások:

EventSystem.current.SetSelectedGameObject(nehezsegslider.gameObject);
                break;
            case menütype.Eredménytábla:

EventSystem.current.SetSelectedGameObject(vissza.gameObject);
                break;
        }

    }
```

A fenti kód segítségével, amikor más almenübe váltok, megváltoztatom az alapértelmezett menüpontot a célmenü első menüpontjára, így mindig a megfelelő menüpont az aktív menüpont, és nem fordulhat elő, hogy az épp kiválasztott lehetőség a már nem látszó canvason, a már nem látszó menüben van.

6.2. Játékos mozgása

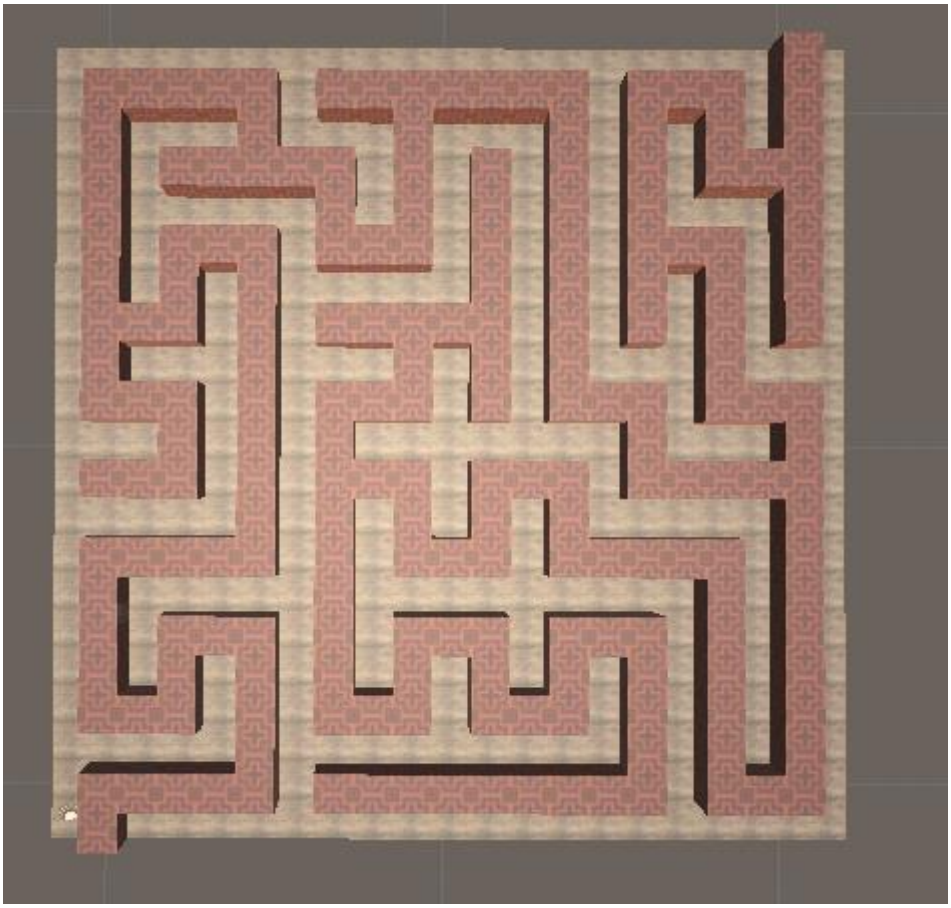
Mivel a játékos önmagát, illetve az esetleges másik játékost nem látja, nem volt szükséges annak külsejét megalkotni. Játékosom ezért egy kapszulába, egy „tic-tac”-ba helyezett kamera, különböző inputtal lehet ezt irányítani. Először úgy próbáltam, hogy a billentyűzet lenyomása alapján változtatom a karakter pozícióját, mint ahogy az a következő kódrészletben

```
        if (Input.GetKeyDown(KeyCode.A))
        {
            Vector3 position = this.transform.position;
            position.x--;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.D))
        {
            Vector3 position = this.transform.position;
            position.x++;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.W))
        {
            Vector3 position = this.transform.position;
            position.y++;
            this.transform.position = position;
        }
        if (Input.GetKeyDown(KeyCode.S))
        {
            Vector3 position = this.transform.position;
            position.y--;
            this.transform.position = position;
        }
```

látszik.

Viszont ezzel az volt a probléma, hogy nem követte a kamera mozgását. A megoldás végül az volt, hogy a kamera aktuális iránya és az eredeti irány közötti szöget felhasználva egy irányvektort elforgatok, és azzal a vektorral mozgatom a játékos karakterét. A kamera aktuális nézőpontját az egérrel lehet változtatni, az egér mozgása forgatja el annak nézőpontját.

6.3. Labirintus generálás



Az első nehéz feladat akkor jelentkezett, mikor a játék magját jelentő labirintust kellett generálni. Különböző módszereket találtam, amik között voltak jobbak és rosszabbak is, valamint olyanok is, amik működése olyan volt, ami által generált labirintus szerintem élvezhetetlen lenne a játékosoknak, mivel az így generált útvesztőnek majdnem az egészét végig kéne minden esetben járnia.

Ez az élvezhetetlen módszer papír alapú, írószerrel könnyedén teljesíthető algoritmus úgy működik, hogy kijelöl kettő, egymásra merőleges falat, ezeken csinál átjárót mind a négy így létrejött negyedben. A negyedeket így tovább negyedeli, mindaddig, míg elfogy a maradék terület. Így kialakul a labirintus, aminek megoldása túl sokáig tart, ha nem látjuk azt felülről.

Ezután találtam rá arra a megoldásra, ami alapján végül megírtam az útvesztő legenerálására szánt függvényt. Ez először egy 2D mátrixban megalkotja a pályát síkban, majd a függvény befejeződése után alkotja meg a ténylegeset térben.

Kezdéskor létrehozok egy int[,] típusú 2D mátrixot, aminek a kiterjedése mindkét irányban ugyan annyi, mértékét pedig a beállításokban kiválasztott nehézség alapján határozom meg. A mátrix (1,1) indexű értékének 0-át veszek fel, itt kezdődik a útvesztő tényleges része. Ezután meghívom a „labirintus” függvényemet az (1,1) értékkel. Azért (1,1), mert ez az első mező indexe, és azért nem (0,0), mivel a pályának hagyok egy külső körvonalat. A függvény elején letárolom a számokat egytől négyig véletlen sorrendben egy vektorban, ezek a négy lehetséges irányt jelölik a derékszögű koordináta rendszerben. Azért véletlen sorrendben, mivel így van biztosítva, hogy a labirintus különböző irányokba haladjon törés után. Végigmegyek az irányokon, és megvizsgálom, hogy az adott irányban a szomszédos pont után üres hely van-e, vagy fal. Amennyiben üres hely van, úgy arra nem szabad haladni, a program továbblép a következő irányra. Amennyiben ott fal van, úgy „kivájja” a szomszédos mezőt, a mátrixban 0-ra állítja a mezőhöz tartozó értéket, és meghívja újra a „labirintus” függvényt annak a pontnak a koordinátájával. Mindez addig ismétlődik, amíg a lehetséges mezők el nem fogynak az adott pontból. Ha ez megtörténik, a program visszatér az előző pontból való vizsgálódáshoz, hogy onnan tovább tud-e még valamilyen irányba haladni. Amikor a függvény végez, a mátrix első és utolsó oszlopában véletlenszerűen kiválasztok egy mezőt, ezeket szintén 0-ra állítom, így megalkotván a be és kijáratot.

```
        int[] irany = new int[] { 0, 1, 2, 3 };
        palya[y, x] = 0;


        for (int i = 0; i < 4; i++)   //irányok véletlenszerű beállítása
        {
            int r = Random.Range(0,3) % 4;
            int  temp = irany[i];
            irany[i] = irany[r];
            irany[r] = temp;
        }


        for (int i = 0; i < 4; i++)
            switch (irany[i])
            {
                case 0:
                    if (y >= 2 && palya[y - 2,x] != 0)
                    {
                        palya[y - 1,x] = 0;
                        labirintus(x, y - 2);
                    }
                    break;
                case 1:
                    if (x >= 2 && palya[y,x - 2] != 0)
                    {
                        palya[y,x - 1] = 0;
                        labirintus(x - 2, y);
                    }
                    break;
                case 2:
                    if (y < meret - 2 && palya[y + 2,x] != 0)
                    {
                        palya[y + 1,x] = 0;
                        labirintus(x, y + 2);
                    }
                    break;
                case 3:
                    if (x < meret - 2 && palya[y,x + 2] != 0)
                    {
                        palya[y,x + 1] = 0;
                        labirintus(x + 2, y);
                    }
                    break;
            }
    }
```

6.4. Szörny

Következő kihívás a szörnyek voltak. A labirintus legenerálása után véletlenszerűen kiválasztottam folyosómezőket, amikre üres collidereket raktam. Amennyiben a játékos ütközik ezzel a colliderrel, azt törlöm, és a kezdőmezőben spawnolok egy szörnyet. Ez a szörny az aktuális pontja és a játékos pozíciója között egy útvonalkereső algoritmussal megkeresem a

legoptimálisabb útvonalat. Ezt az útvonalat folyamatosan frissítem az alapján, ahogy a játékos mozog. A szörny ezen az útvonalon üldözi a játékost, először nagyon lassan, majd az idő haladásával folyamatosan, egyre gyorsabban üldöz a szörny, míg utol nem éri a játékost. Ha a játékos collidere ütközik a szörnyével, akkor vége a játéknak.

6.5. Kelepce

Következő megoldandó problémám az kelepcék működése volt. Korábban még nem foglalkoztam animációk készítésével, így ezt nem volt a legegyszerűbb kivitelezni. Ezt úgy oldottam meg, hogy ha a csapdába besétál a játékos, akkor egy Trigger esemény hatására elindul a plafon süllyedésének az animációja, majd ha a másik játékos besétál a saját mezőjébe, az ellentétes irányú animáció fut le, így felemelkedvén a falak a karakter körül.

6.6. Hálózatos játék

A hálózati részt kettő aszinkron kapcsolat létrehozásával oldottam meg, így kialakítva egy szinkron kapcsolatot. Indítás után a „server” gép, amelyik a hálózati játék indítása gombbal indítja a játékot, egy kapcsolódásra van. A másik játékos, a „kliens” pedig ehhez a serverhez kapcsolódik. Miután a kapcsolat létrejött, a kliens is nyit egy server kapcsolatot, amire az eredeti server kapcsolódik kliensként.

A párhuzamos kapcsolatoknak köszönhetően a kétirányú adatátvitel gondba nem ütközhet. A legnagyobb probléma az lehetett volna, hogy valamelyik irányba megakad a kommunikáció, ez pedig teljesen leállíthatta volna az adatforgalmat. Ezért így, ha meg is akad egy irányban az adatfolyam, a másik irányban továbbra is él.

6.7. Eredménytábla

A játékosok által elért eredményeket két külön állományban, a helyi számítógépen tárolom. A fáljban le van tárolva, hogy ki érte el azt az eredményt, mikor, és hogy hány pontot szerzett. Kétjátékos

eredménytáblán a két név abc rend szerint rendezve nev1+nev2 formában jelenik meg. Az elért pontot az alábbi formula alapján számolom ki:

a pályára tervezett időből kivonom a teljesítés idejét, minden másodperc +3*pályaméret pont, amennyiben negatív érték jön ki, úgy ez a rész 0 pont

ehhez adok +100 pontot, mint teljesítési bónusz

+1000 pont, ha időre játszott

+2000 pont, ha engedélyezve voltak a szörnycsapdák

+2000 pont, ha a kelepcék engedélyezve voltak

+5000 pont, ha veszély módban történt a játék

-250 pont minden aktiválódott kelepce és szörnycsapda után

minden elveszített életpont után -15 pont

7. Összefoglalás

A szakdolgozat elkészítése közben lehetőségem volt különböző eljárásokat megismerni, amik rendkívül hasznosak tudnak lenni az informatika több területén. Ismeretet szereztem összetettebb képszerkesztésben, modellek készítésében, a Unity3D keretrendszer használatában, ami elképzelhető, hogy a jövőben az egyik legnépszerűbb játékmotor lesz, valamint hálózati kommunikáció létrehozásában programban, ami nemcsak játékprogramokban, hanem egyéb üzleti szoftverekben is sokszor szükséges.

8. Felhasznált irodalmak

- https://en.wikipedia.org/wiki/Maze_generation_algorithm (2016.02.15)
- https://msdn.microsoft.com/en-us/library/system.net.sockets.tcplistener(v=vs.110).aspx (2016.02.15)
- http://stackoverflow.com/questions/12361924/difference-between-tcp-listener-and-socket (2016.02.15)
- https://infoc.eet.bme.hu (2016.02.15)
- C# Network Programming by Richard Blum ISBN:0782141765
- https://unity3d.com/learn/tutorials/projects/roll-a-ball/moving-the-player (2016.02.15)
- https://en.wikipedia.org/wiki/Pathfinding (2016.02.15)
- https://en.wikipedia.org/wiki/A*_search_algorithm (2016.02.15)
- https://unity3d.com/learn/tutorials/modules/beginner/physics/colliders-as-triggers (2016.02.15)
- http://docs.unity3d.com/ScriptReference/UI.Button.html (2016.02.15)
- http://docs.unity3d.com/ScriptReference/Input.html (2016.02.15)
- http://docs.unity3d.com/410/Documentation/Manual/Lightmapping.html (2016.02.15)

9. Melléklet

Melléklet:

- 1 darab CD

CD mellékelt tartalma:

- 1 darab „.exe” kiterjesztésű állomány
- 1 darab mappa, ami a szakdolgozat állományait tartalmazza