Budapesti Műszaki Szakképzési Centrum  
Neumann János Számítástechnikai Szakgimnáziuma

Szakképesítés neve: Szoftverfejlesztő

OKJ száma: 54 213 05

ZÁRÓDOLGOZAT

Tales of The Labyrinth

Oláh Katalin Szegedi Achilles Károly

konzulens 14.rsz

Budapest, 2017.

Tartalomjegyzék

[A szakdolgozatról 2](#_Toc479092818)

[Témaválasztás 2](#_Toc479092819)

[A dolgozat nehézségei 2](#_Toc479092820)

[Felhasználói dokumentáció 3](#_Toc479092821)

[A játék célja 3](#_Toc479092822)

[Rendszerkövetelmény 3](#_Toc479092823)

[Telepítés. A történet születése! 4](#_Toc479092824)

[A program használata. Kezdődjék egy újabb mese! 6](#_Toc479092825)

[A játéktér 6](#_Toc479092826)

[Irányítás 6](#_Toc479092827)

[Játékmenet 6](#_Toc479092828)

[Hiba esetén 7](#_Toc479092829)

[Ismert hibák 7](#_Toc479092830)

[Fejlesztői dokumentáció 8](#_Toc479092831)

[Felhasznált szoftverek 8](#_Toc479092832)

[A program szerkezeti felépítése 9](#_Toc479092833)

[Fő egységek 9](#_Toc479092834)

[A program futása 10](#_Toc479092835)

[Program szintű változók és a config.json 11](#_Toc479092836)

[2DGrapicsElement, általános osztály felépítés 13](#_Toc479092837)

[Textúrák, TextureFromFile osztály 14](#_Toc479092838)

[Labirintus cellák, Iconection interfész és Cell Osztály 14](#_Toc479092839)

[A térkép generálása, MapArea osztály 14](#_Toc479092840)

A szakdolgozatról

Témaválasztás

A téma kiválasztásnál az első döntés szinte azonnal megvolt mivel tudtam, hogy játékot akarok készíteni. Egy egyszerű okból mivel korábbról már volt egy projektem, ahol C# alapon porbáltam játékot készíteni, ezért döntöttem úgy fel akarom használni az akkor kifejlesztett rendszer kezdeményt, ami a mostani program alapját hozta létre egy erös bővítés és átalakítás után.

A második gondolat a játék működését adta meg mivel a téma választáskor épp egy kisebb regényt olvastam, amiben az egyik faj tagjai robotok voltak. Ebben a fajban mindenki tartozott egy fürtnek neveztet csoportba, ahol a fürt célja a minél optimálisabb módon megsemmisíteni az ellenséget. Ekkor jött az ötlet miért nem próbálom meg ezt a viselkedést egy stratégiai játék köntösbe ültetni mintha a játékos a ezeket a fürtöket irányító legfelsőbb robot lenne, A legfelső döntéshozó. Ez az egyik oka miért is nem képesek a játékban az egységek önállóan bármit cselekedni az ölésen kívül.

A dolgozat nehézségei

Az egyik legnagyobb nehézség az volt, hogy a választott keretrendszer miatt meg kellet szoknom, hogy a grafikai elemek képernyőre helyezése és a viselkedésük két teljesen külön függvény. A következő nehézség, hogy a program fejlesztése során nem volt elég csak néhány osztályal foglalkozni hanem közel 30 osztály volt négy különálló projektben, ezért nem lehetett a szerkezet tervezésekor nem jól végig gondoltan elkezdeni a fejlesztést mivel egy ilyen hiba akár a teljes programot tönkre is teheti, mint ahogy majdnem meg is történt két változó felcserélésekor. Az ekkor létrejött hiba több mint egy hónappal később derült ki amikor a következő elem erre a részre támaszkodott volna.

Végül talán a legnagyobb kihívás, hogy a stabilitás és a felhasználói élmény miatt elvesztettem a Windows Form ablakok kényelmét és minden kis képernyőn megjelenő elemet meg kellett valósítani. Ez több dolgot adott egyszer a teljes szabadságot, mivel én döntöttem mi és hogyan működjen, másodszor a legnagyobb nehézséget is mivel a fejlesztés egy kalanddá változott, ahogy alakult a rendszer a világ mögött. Szinte, mint egy gyermek születése még bármi lehet belőle és mégis ott voltak a keretrendszer akadályai ezért egy krimi is volt. Mi a megoldás? Miért történt valami? Miért az történt? De végül minden kérdésre lett egy válasz kivéve egyre teljes lett-e a program. Természetesen nem, mivel ez csak az első mese volt a labirintusból mégpedig a Keletkezés meséje.

Felhasználói dokumentáció

A játék célja

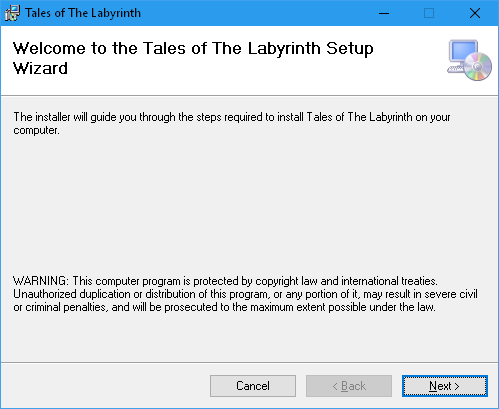
A játékban egy labirintusban kell két fürt segítségével legyőzni az ellenséges bázist és csapatokat. Ezt nehezítendő a fürt tagjai nem támadnak meg csak olyan ellenséget, amely keresztezi az útjukat. Ezért a ennek a feladata a legfelső döntéshozóra rád a játékosra hárul. Vajon a labirintus mesék ezen fejezete egy boldog vég felé vezet, vagy egy újabb kudarc lesz. Ez csak rajtad múlik, aki kinyitja e könyvet és újabb mesét ír a labirintus mesék közé.

Rendszerkövetelmény

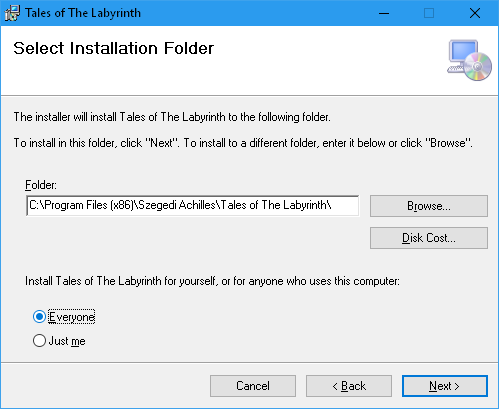
* processzor: legalább 1 GHz
* memória: 100 MB szabad
* lemezterület: minimum 40 MB
* DirectX 11 vagy újabb
* Windows 7 service pack 1 vagy újabb Windows rendszer
* .NET 4.5.2 vagy azt tartalmazó későbbi verzió

Telepítés.  
A történet születése!

A telepítés a setup.exe elindításával kezdhető meg.

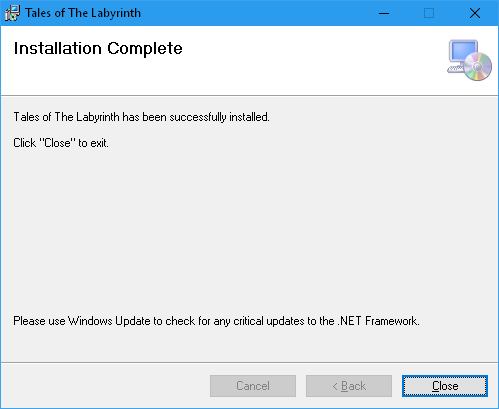


Az első képernyőn a next gomb megnyomásával kezdhető meg a telepítési folyamat.



A következő képernyőn látható a telepítési mappa, amely a browse gomb megnyomásával változtatható meg.  
A Disk Cost gomb megnyomásával látható a szükséges lemezterület meghatókra bontva. Az Everyone jelölő segítségével választható ki, hogy a telepítő minden felhasználónak telepítse-e a programot. A just me jelölő segítségével csak a telepítést végző felhasználónak kerül telepítésre. A next gombbal léphetünk a következő képernyőre, ahol ismételt next gombbal megkezdhető a telepítési folyamat.

Figyelem: Windows 8 és későbbi rendszeren nyomjunk az igen gombra a felhasználói fiók felügyelete ablakon a telepítés folytatásához.



Amennyiben a telepítés sikerült a close gombbal zárhatjuk be az ablakot.

Figyelem: A program működéséhez DirectX 11 szükséges, amelynek az esetleges telepítése a felhasználó felelőssége, mivel a hivatalos tájékoztatás szerint Windows 7-töl a rendszer tartalmazza ezt vagy egy kompatibilis verziót.

Megjegyzés: Az utolsó képernyőn a telepítő figyelmeztet, hogy ne felejtsük el telepíteni a Windows update segítségével az esetleges: NET frissítéseket mivel ennek hiánya biztonsági kockázatot jelent.

A telepítés után a start menüben Tales of The Labyrinth néven indítható a program vagy az asztalon létrejövő ugyanilyen nevű ikonnal

A program használata.  
Kezdődjék egy újabb mese!

A játéktér

Indítás után a képen látható játéktér fogad négy fő elemmel.



* a labirintus, ahol a játék történik
* a fürtök állapot visszajelzője a képernyő bal oldalán
* a saját bázisunk összes élete /megmaradt élete a képernyő bal sarkában
* az ellenséges bázis összes élete/megmaradt élete a képernyő jobb sarkában

Ezen kívül még két fontos elem van mindkettő a labirintuson belül egy kék színű épület a saját bázisunk és egy ugyan olyan piros épület az ellenséges bázis.

Irányítás

|  |  |
| --- | --- |
| A fürt lehívása/kiválasztása | X |
| B fürt lehívása/kiválasztása | C |
| kilépés | E |
| fürt mozgatása | Bal egérgomb |

Játékmenet

A játéktérben a fürtöket a kiválasztás után a labirintuson való kattintással lehet utasítani, ekkor az adott fürt megpróbál eljutni a kijelölt pontra, ha lehetséges. Ha az a pont az ellenséges bázis, akkor a megérkezéskor azonnal meg fogja támadni az épületet.

A fürtök azonnal meg fogják támadni az ellenséges csapatokat, ha keresztezik az útjukat de ezért nem fognak megállni, ha van kijelölt céljuk.

A játék véget ér a következő esetekben, ha valamelyik bázis megsemmisül, ha ez az ellenség bázisa volt akkor természetesen a játékos nyer és egy boldog mese kerül a könyvbe, ha a sajátja volt akkor viszont egy szomorú történet kerül be mivel ez egy vesztes csata lett. Ha megsemmisül az összes ellenséges egység akkor a játékos nyer. Ha megsemmisül a játékos mindkét fürtje akkor a játékos veszít.

Fogd meg hát a tollat kedves olvasó és adj egy újabb történetet a labirintus mesék közé!

Hiba esetén

Hiba esetén a [steamhunter97@gmail.com](mailto:steamhunter97@gmail.com) email címre lehet azt bejelenteni vagy a <https://github.com/steamhunter/TotL/issues> oldalon új hibajegy létrehozásával, amit a new issue gombbal lehet megtenni, itt megtekinthetőtek az eddig már ismert hibák.

Ismert hibák

A jelenlegi 0.3-as verziónak egy ismert hibája van. Egyes gépeken a program nem indul mivel nem sikerül betöltenie a működéshez szükséges betűkészlet fájlt, ilyenkor a programot –no-text paraméterrel szükséges indítani. Ezt megtehetjük, ha készítünk egy parancsikont a TotL.exe-ről és a tulajdonságok cél mezőbe egy szóköz után beírjuk, hogy –no-text.

Fejlesztői dokumentáció

Felhasznált szoftverek

* Visual studio 2015/2017 programozás
* SharpDX DirectX wrapper (sharpdx.org) a DirectX egyszerűsített használatáért.
* Fast A-Star implemetation for C# Christoph Husse által az egységek utvonalkereséséhez amely a rendszerben a PathFinder.AStar névtérben kapott helyet.  
  (https://www.codeproject.com/Articles/118015/Fast-A-Star-D-Implementation-for-C)
* Microsoft Visual Studio 2017 Installer Projects a telepítő elkészítéséhez  
  (https://marketplace.visualstudio.com/items?itemName=VisualStudioProductTeam.MicrosoftVisualStudio2017InstallerProjects)
* GIMP kép szerkesztő textúrák elkészítéséhez

A program szerkezeti felépítése

Fő egységek

A program az alábbi fő egységekre botható.

Megjegyzés: az Összes saját fejlesztésről elmondható, hogy a dll nevével megegyező projektben található a solution-ön belül.

-SharpDX wrapper amelyben a DirectX rendszerrel való kapcsolat történik a programba nuget csomag segítségével került be. Az alábbi dll-ek tartoznak hozzá

* -Assimp32/64
* -AssimpNET
* -SharpDX.\*

-PathFinder A SharpDX tetejére kerülő már saját fejlesztésű rendszer, amely tartalmazza az alapvető grafikai és működési feladatokhoz tartozó kódot lényegében ez mondható a játékmotornak. nevéből adódóan a PathFinder.dll ben található.

-PathFinder.AStar az egységek útvonal keresését végző kódot tartalmazza astar algoritmus szerint. Hozzá igazítva a TotL.Labyrinth. névtér IConnection interfészéhez. A PathFinder.Astar.dll ben található.

-TotL a játék fő logikáját tartalmazza, többek közt a pálya generálását. Ebből a modulból készül a program egyetlen futtatható fájlja.

-TotL.Labyrith a labirintus celláit tartalmazza az útvonalkereső csatlakozási pontjával együtt.

A program futása

A program indításakor a futás az alábbi kóddal indul, ahol elsőként ellenőrzésre kerül kapott-e a program bármilyen indítási paramétert.

namespace TotL

{

class Program

{

static void Main(string[] args)

{

foreach (string ar in args)

{

if (ar == "-console")

{

cons.onDebug = true;

cons.debugMessage("debug enabled");

}

if (ar=="-path-debug-draw")

{

Vars.path\_debug\_Draw = true;

}

if (ar=="-no-text")

{

Vars.noTextMode = true;

}

}

using (TotL game = new TotL())

{

game.Run();

}

}

}

}

A -console paraméter esetén a program indulásakor egy parancssor is létrejön, ahol információt kaphatunk az egyes alrendszerekről.

A –no-text paraméter segítségével kikapcsolható az összes szöveg a játékon belül, mivel egy ismert hiba miatt egyes gépeken nem sikerül a használt font betöltése.

A -path-dedug-draw paraméterrel az útvonalkeresést lehet vizsgálni.

Megjegyzés: egy ismert probléma miatt a 0.3-as jelenlegi verzióban ez a paraméter nincs használatban

A következő elem a TotL segítségével létrejön a játék fő osztálya. Amely a SharpDX.Toolkit.Game osztályból származik a következő lánc szerint.

TotL-> PathFinder.Toolkit.PathFinderGame ahol az alábbi SharpDX függvények kapnak védelmet egy állapot rendszer segítségével.

|  |  |
| --- | --- |
| **SharpDX függvény** | **PathFinder Függvény** |
| Initialize | Init |
| LoadConent | Load |
| Draw | TickDraw |
| Update | TickUpdate |

A rendszer a következő állapotokban lehet. Ezeket az állapotokat a PathFinder.gamestates enum határozza meg, ami a PathFiner.Vars.cs fájlban található és futás közben a gamestate változó tárolja a Vars osztályban.

|  |  |
| --- | --- |
| **Állapot** | **Leírás** |
| notinitialized | a rendszer nem inicializálta magát. Az Init függvényt meg kell hívni. |
| initializing | Az Init függvény meg lett hívva de az inicializálás még nem történt meg. |
| initialized:notLoaded | Az inicializálás megtörtént. A Load függvényt meg kell hívni |
| initialized:loaded | Mind az inicializálás mind a betöltés megtörtént a felhasználó már a játékteret látja. |

Az egyes függvények az alábbi táblázat szerinti állapotokban fognak érdemi munkát végezni, és változtatnak az állapoton.

|  |  |  |
| --- | --- | --- |
| **függvény** | **szükséges állapot** | **befejezéskori állapot** |
| Init | nincs (notInitialized ajánlott) | initialized\_notLoaded |
| Load | initialized\_notLoaded | initialized\_loaded |
| Draw | initialized\_Loaded | initialized\_Loaded |
| Update | initialized\_Loaded | initialized\_Loaded |

PathFinderGame->SharpDX.Toolkit.Game

Program szintű változók és a config.json

Az összes olyan változó, amely a teljes programban használva van a PathFidner.Vars osztályban találhatóak és az alábbi táblázat mutatja.

|  |  |
| --- | --- |
| gamestates gamestate | a program fő állapota. (lásd korábbi fejezet) |
| internalstates mapstate | a pálya betöltését segítő állapotok. |
| SpriteBatch spriteBatch | a Draw függvény által használt a kirajzolást végző SharpDX objektum. |
| SharpDX.Direct3D11.Device device | a grafikus eszköz objektuma. |
| int ScreenWidth | képernyő széleség. |
| int ScreenHeight | képernyő magasság. |
| Game game | a játék objektuma valójában a TotL osztály indításkor létrejövő példánya. |
| int seed | a labirintus generálásnál használt kezdő érték. |
| Random random | a seed el létrehozott központi véletlen generátor. |
| configjson.configstructure config | a config.json adatait tartalmazó objektum |
| float unitSize | egy labirintus cella mérete a képernyőn (1932x1086 esetén kb. 64 pixel) |
| bool path\_debug\_Draw | az ugyanilyen nevű paraméter jelenlétét jelzi |
| KeyboardManager mykeyboardmanager | billentyű eseményeket kezelő SharpDX objektum |
| MouseManager mymousemanager | egér eseményeket kezelő SharpDX objektum |
| SharpDX.Toolkit.Graphics.SpriteFont font | a rendszerben használt font objektuma a myfont.tkb által betöltött Windows font (Arial) |

A config.json fájl tartalmazza a program konfigurációját a weight sorok a cellák súlyozását jelölik, amelynek összege mindig 100, másik elem, hogy a program teljes képernyőn induljon-e. Alap eseteben ez a következőként néz ki.

{

"fc\_weight":1,

"cross\_weight":9,

"deadend\_weight":10,

"oneside\_weight":25,

"twoside\_weight":25,

"tunnel\_weight":30,

"isFullScreen": true

}

2DGrapicsElement, általános osztály felépítés

Helyileg a PathFinder.\_2D névtérben található és a program összes grafikai elem alapját adja.

A következő kódrészlet mutatja a belső felépítését:

namespace PathFinder.\_2D

{

public class \_2DGraphicsElement

{

protected ShaderResourceView texture;

public Rectangle rectangle;

protected float \_locationX;

protected float \_locationY;

public static Texture2D getTexture(string entity, Game game)

{

throw new DeprecatedMethodException("getTexture is not useable");

}

public \_2DGraphicsElement()

public virtual void update()

public virtual void Load()

public virtual void Initialize()

public virtual void draw()

}

}

Az összes itt definiált kódról elmondható, hogy nem tartalmaz működő kódot ez alól csak a draw kivétel, ami a rectangle és texture változók segítségével a locationben pixelben meghatározott helyre kirajzolja az adott elemet.

Megjegyzés: A draw függvényen kívül az összes függvény NotImplementedException("hívás a 2D Graphics Element alap függvényre") üzenettel hibát dob, kötelező a származtatott osztályban felül írni.

Az összes osztály, amely valamilyen szinten kapcsolatba kerül a játék központi osztályával pontosan ugyan ezeket a függvényeket tartalmazza segítve a fő függvényekkel való könnyebb kapcsolatot.

Textúrák, TextureFromFile osztály

A program összes textúráját a textureFromFile osztályban található TextureProcessor osztály végzi. Jelenleg egyetlen függvénye van a getTexture, amely a kiterjesztés nélküli fájlnév alapján megkeresi és ShaderResourecView objektumként visszaadja a kért texturát. Lazyload megvalósításban készült ezért csak akkor tölti be a textúrát, ha valahonnan a getTexture függvénnyel kérjük azt. Ezt kiegészíti az, hogy a rendszer csak egyszer tölti be az adott textúrát utána mindig ugyanazt az objektumot adja vissza.

Labirintus cellák, Iconection interfész és Cell Osztály

Az első lényeges elem az Iconnection interfész mivel a benne található változók lényegi szerepet játszanak abban, hogy későbbiekben a térkép generáláskor és az útvonal kereséskor egy egyszerű szerkezet segítségével foghassuk meg a pálya elemei. Ezt a működést segíti a Connection osztály, ahol egy általános megvalósítása található az Iconnection-nek mivel a generáláskor az adatszerkezet nem fogad el tisztán interfészt. Első sorban a lényeges változok a négy irány, amely a cella adott irányba való nyitottságát jelzi(true=nyitott), illetve a closedsides változó, amely a zárt oldalak száma ez később a generálást fogja gyorsítani.

A Cell osztály adja a labirintus összes cellájának alapját. A \_2DGraphicsElement-böl származik és implementálja az Iconnection interfészt mintegy összefűzve a program grafikai és logikai szerkezetét és megadva a rendszer alapját egy stabil működéshez.

A térkép generálása, MapArea osztály

Az első fontosabb rész a game system init ahol két menet segítségével generálunk egy seed-et 10 000 000 és 99 999 999 között, amit a központi random generátor kap meg seedként.

A következő lényeges régió a Border generator ahol miután létrejön egy mátrix, ami minden irányba eggyel nagyobb, mint a tényleges térkép, ezt a peremet az Iconnection interfész szerint minden irányba zárt állapotúvá tesszük mintegy körül zárva a térkép helyét.

Következő régió a generate start base ahol a központi random generátor segítségével kiválasztjuk a játékos és az ellenség bázisának helyét úgy, hogy legalább 10 cella távolság legyen köztük.

Ezután a valid cell gen régióban kigeneráljuk a cellákat itt a rendszer a CheckFitting segítségével megkísérli a kiválasztott helyre a cellát beilleszteni és ezt addig ismételi, amíg nem sikerül egyet sikeresen beilleszteni.

Megjegyzés: a rendszerben nem okozhat problémát, ha esetleg a generálás nem vezet a cellába utat mivel a teljesen zárt cella 1 % esélyt kap a generálás során így mindenféleképpen be fog fejeződni a művelet.

Ezekután egy lényeges elem van ellenőrizni, hogy van-e útvonal a két bázis között, ha nincs a művelet újraindul a seed generálásától.

A MapArea tartalmazza még a fürtökkel kapcsolatos feladatokat is, de ezek már az Update függvényben találhatóak ClusterA, ClusterB és EnemyCluster régiókban.  
Itt egy érdekes rendszer indul el mivel a különböző tick és index változók segítségével a működés több update híváson keresztül fut le ezzel is enyhítve a rendszer terhelésén, ezért is látható, hogy a fürt tagjai nem egyszerre kezdik meg a mozgási feladatok elvégzését.

Az Update-ben található még a fürtök találkozásakor a sebzést végző kód is ahol a rendszer megvizsgálja található-e egymás közelében két olyan egység, ami nem egy oldalon harcol.

Végül az utolsó függvény, ami még itt a található az a Draw ahol a grafikai elemek rajzolása történik. Először ellenőrzi kell e valamilyen játék vége képernyőt kirajzolni. Másodiknak felkerül a két életet jelző szövegrész. Ezek után kirajzolásra kerül a térkép majd a fürtök végül pedig a két visszajelző elem kerül fel.