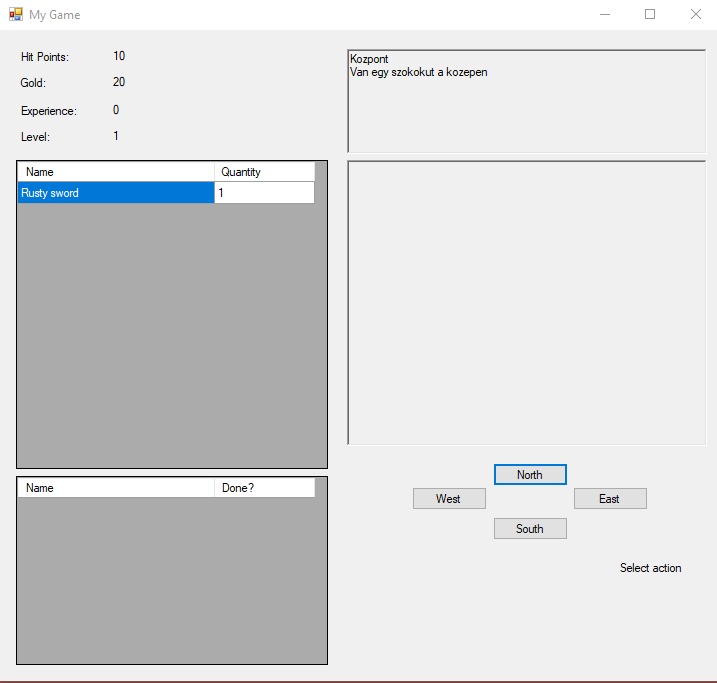
Szakdolgozat

Super Adventure - Szöveg alapú RPG (Role Playing Game)



**Felkészítő tanár:**  **Készítette:**

**Kotró László-Lehel** **Csányi Patrik-Tibor**

**XII.D**

Tartalomjegyzék

1. Bevezetés 3
2. Szoftver és Hardver követelmények 4
3. A dolgozat leírása 5
4. Egy komplexebb modul bővebb leírása 9
5. Használati utasítások 18
6. Bibliográfia 20

Bevezetés

A dolgozatom megírása során egy egyszerű, szöveg alapú RPG-t (Role Playing Game) készítettem. Ezt a projektet több okból választottam. Első sorban a videó játékok készítése egy számomra nagyon érdekes téma, és karrierem során szeretném magam kipróbálni ebben a körben is, ezért úgy véltem, hogy ez egy tökéletes pillanat az alapok megtanulásához. Továbbá egy videójáték elkészítése egy érdekes kihívásként is hangzott. Mind ezek mellett szerintem a végeredmény, a játék, egy sokkal szórakoztatóbb és érdekesebb termék, mint akármi más amit jelenlegi tudásommal tudtam volna alkotni. Ennek ellenére tudatában vagyok a videójátékok elkészítésével járó munkával valamint igényelt tudással, ezért úgy döntöttem, hogy megpróbálok egy egyszerűbb játékot készíteni, mely ugyanakkor kihívás is lesz számomra. Továbbá ezt a lehetőséget arra is használtam, hogy tanuljam a programozás alapjait, valamint tovább fejlesszem magam, így a végeredmény, bár nem lenyűgöző mint program, véleményem szerint bemutatja a fejlődést melyen végig mentem a program megírása során. A program megírásához a Microsoft Visual C# alkalmazást használtam, első sorban, mivel a Windows Form Application könnyebben és gyorsabban használható videójátékok megírásához. Továbbá a C# egy magasabb szintű programozási nyelv, és bár nem használtam a Visual Stúdióban jelenlévő LINQ-et egy videó játék megírása könnyebbnek tűnt C#-ban. A program relatív egyszerű, így nem hinném, hogy eredetiek a megoldásaim a problémákra, amelyekkel szembe ütköztem, viszont ezeknek a megoldásoknak a megírása/megvalósítása tükrözi a gondolkodásmódom.

Szoftver és Hardver követelmények

A program futtatásához szükséges szoftver és hardver követelmények megegyeznek a C# és .net rendszerek követelményeivel. A .net rendszerhez szükséges:

* 1 GHz-es processzor,
* 512MB RAM,
* 4,5 GB szabad tárhely,

Operációs rendszer szempontjából pedig Windows 7 vagy ennél később megjelenő operációs rendszerek használhatóak. A C# követelményei alacsonyabbak minden szempontból. Továbbá a program működtetéséhez szükség van a Microsoft Visual Stúdió 2019 nevű szoftverre.

A dolgozat leírása

A programomat két projektre osztottam. Az első, az “Engine”, amely a Klasszok és a program nagy részét tárolja, míg a “SuperAdventure” a UI-t tárolja valamint konkrétan a játékhoz kapcsolódó programot. Ennek oka, hogy ily módon az “Engine” projektet újra lehet használni más programok/játékok megírásához, mely alapvetően különbözhet ettől.

Az “Engine” projekt 14 Klasszból áll, melyek tárolják a játékos, a szörnyek, küldetések, helyszínek és a játékban megjelenő tárgyak tulajdonságait, de egy random szám generátor és a “World” klassz is megjelenik, mely tárolja az összes világban megjelenő tárgyat, szörnyet, helyszínt és küldetést. A “LivingCreature” valamint az “Item” klasszok base klasszok, melyeknek a “Player” és “Monster” valamint “HealingPotion” és “Weapon” klasszok öröklik tulajdonságait

LivingCreatures

* currentHitPoint
* maximumHitPoints

Player

* currentHitPoint
* maximumHitPoints
* Gold
* ExperiencePoints
* Level
* CurrentLocation
* Inventory
* Quests

Monster

* currentHitPoint
* maximumHitPoints
* ID
* Name
* MaximumDamage
* RewardExperiencePoints
* RewardGold
* LootTable

Item

* ID
* Name
* NeamePlural

HealingPotion

* ID
* Name
* NeamePlural
* AmountToHeal

Weapon

* ID
* Name
* NeamePlural
* MinimumDamage
* MaximumDamage

A “LootTable” tulajdonsága, a “Monster” klassznak egy lista, hisz több tárgyat tartalmazhat. Hasonló módon, a “Player” “Inventory” és “Quests” tulajdonságai is listák, hisz több tárgyat, valamint küldetést tartalmazhatnak. A “Player” klasszban megfigyelhetünk 6 alprogramot is, melyek sorrendben, azt ellenőrzik, hogy: szükséges-e egy tárgy egy bizonyos helyszínre való belépéshez, valamint, hogy ez a tárgy a játékos tulajdonában áll-e; a játékos megkapott-e valamint teljesített-e egy bizonyos küldetést; megvan-e minden tárgy egy küldetés teljesítéséhez, valamint, ha igen ezen tárgyak eltávolítása; végül pedig a játékos jutalmának hozzáadása az “Inventory”-hoz és a küldetés teljesítettnek jelölése. Ezen alprogramok mind a “SuperAdventure” projektben lesznek használva. Továbbá a “RandomNumberGenerator” valamint a “World” klasszokon kívül mindegyik tartalmaz egy-egy constructor-t is mellyel könnyebben meg lehet adni a klasszok segítségével készített tárgyak tulajdonságait

A “World” klassz 5 főbb részre osztható. Az első részben 4 listát hozunk létre, melyek a tárgyakat, szörnyeket, küldetéseket, és helyszíneket fogják tárolni.

public static readonly List<Item> Items = new List<Item>();

public static readonly List<Monster> Monsters = new List<Monster>();

public static readonly List<Quest> Quests = new List<Quest>();

public static readonly List<Location> Locations = new List<Location>();

Ezt követően, minden helyszínnek, tárgynak, szörnynek és küldetésnek megfeleltetünk egy-egy “ID”-t. (Identification Number)

public const int ITEM\_ID\_RUSTY\_SWORD = 1;

public const int ITEM\_ID\_RAT\_TAIL = 2;

public const int ITEM\_ID\_PIECE\_OF\_FUR = 3;

public const int ITEM\_ID\_SNAKE\_FANG = 4;

public const int ITEM\_ID\_SNAKESKIN = 5;

public const int ITEM\_ID\_CLUB = 6;

public const int ITEM\_ID\_HEALING\_POTION = 7;

public const int ITEM\_ID\_SPIDER\_FANG = 8;

public const int ITEM\_ID\_SPIDER\_SILK = 9;

public const int ITEM\_ID\_ADVENTURER\_PASS = 10;

public const int MONSTER\_ID\_RAT = 1;

public const int MONSTER\_ID\_SNAKE = 2;

public const int MONSTER\_ID\_GIANT\_SPIDER = 3;

public const int QUEST\_ID\_CLEAR\_ALCHEMIST\_GARDEN = 1;

public const int QUEST\_ID\_CLEAR\_FARMERS\_FIELD = 2;

public const int LOCATION\_ID\_HOME = 1;

public const int LOCATION\_ID\_TOWN\_SQUARE = 2;

public const int LOCATION\_ID\_GUARD\_POST = 3;

public const int LOCATION\_ID\_ALCHEMIST\_HUT = 4;

public const int LOCATION\_ID\_ALCHEMIST\_GARDEN = 5;

public const int LOCATION\_ID\_FARMHOUSE = 6;

public const int LOCATION\_ID\_FARM\_FIELD = 7;

public const int LOCATION\_ID\_BRIDGE = 8;

public const int LOCATION\_ID\_SPIDER\_FIELD = 9;

A tárgyak, helyszínek, küldetések, és szörnyek tulajdonságainak megadását 4 részre osztottam, hogy könnyebben átlátható és kezelhető legyen. Ezeket tölti be a játék a program indításakor

static World()

{

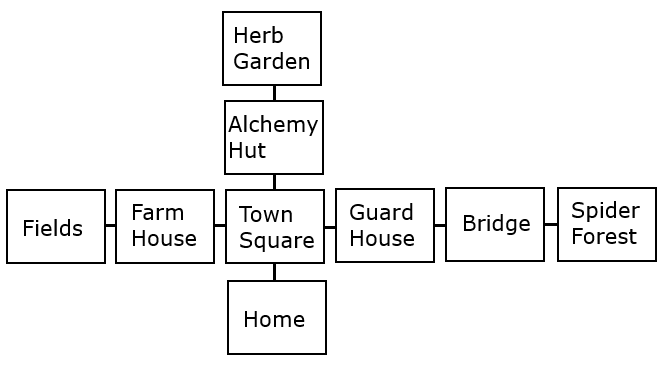
PopulateItems();

PopulateMonsters();

PopulateQuests();

PopulateLocations();

}

A 4-ik rész ennek a négy alprogramnak a megírását tárolja, míg az 5-ik rész ismét négy alprogramot tartalmaz, melyeket a 4-ik rész megírásában használtam. A világ helyszínei a következő összefüggésben állnak egymással 

Egy komplexebb modul bővebb leírása

A legbonyolultabb része a programnak a “SuperAdventure” projekt. Itt találkoznak, és vannak felhasználva az “Engine”-ben megírt klasszok. Ezt jelzi a using Engine; sor is. A program elején a játékos, és a “\_currentMonster” változók vannak bevezetve. A “\_currentMonster” változó azért szükséges, mivel a játékos szörnyekkel való harcolása során, ha a “World” klasszban megjelenő szörnyekkel harcolna, ezek “meghalnának” és nem lehetne ismételten ugyan azt a szörnyet legyőzni. A változó által, a szörny mellyel a játékos harcol el van mentve a “\_currentMonster” változóba, és ezzel harcol a játékos. Ezt követően, a játékos alap tulajdonságait adjuk meg, és írjuk ki az UI-re, valamint egy rozsdás kardot adunk neki mint kezdő fegyver.

private Player \_player;

private Monster \_currentMonster;

\_player = new Player(10, 10, 20, 0, 1);

MoveTo(World.LocationByID(World.LOCATION\_ID\_HOME));

\_player.Inventory.Add(new InventoryItem(World.ItemByID(World.ITEM\_ID\_RUSTY\_SWORD), 1));

lblHitPoints.Text = \_player.CurrentHitPoints.ToString();

lblGold.Text = \_player.Gold.ToString();

lblExperience.Text = \_player.ExperiencePoints.ToString();

lblLevel.Text = \_player.Level.ToString();

A következő lépésben, a mozgás gombokra való klikkeléssel a “MoveTo” funkciót indítjuk el, ez ad lehetőséget a helyszín változtatására. Itt számos dolgot leellenőrzünk. Először megnézzük hogy van-e szükség egy bizonyos tárgyra a kiválasztott helyszínre való belépéshez. Ha nem a játékost beengedjük, és jelezzük a játékosnak, hogy helyet változtatott. Ha ez nem lehetséges az ellenkezőjét közöljük a játékossal. Ezen ellenőrzések megvalósításához a “Player” klasszban lévő alprogramokat használjuk

private void btnNorth\_Click(object sender, EventArgs e)

{

MoveTo(\_player.CurrentLocation.LocationToNorth);

}

private void btnEast\_Click(object sender, EventArgs e)

{

MoveTo(\_player.CurrentLocation.LocationToEast);

}

private void btnSouth\_Click(object sender, EventArgs e)

{

MoveTo(\_player.CurrentLocation.LocationToSouth);

}

private void btnWest\_Click(object sender, EventArgs e)

{

MoveTo(\_player.CurrentLocation.LocationToWest);

}

private void MoveTo(Location newLocation)

{

//kell-e item hogy bemenjunk

if(!\_player.HasRequiredItemToEnterThisLocation(newLocation))

{

rtbMessages.Text += "You must have an " + newLocation.ItemRequiredToEnter.Name + " to enter this location." + Environment.NewLine;

return;

}

//Uj helyre lepes

\_player.CurrentLocation = newLocation;

Az új helyszínen megnézzük, hogy milyen irányokba lehet haladni, és csak a megfelelő gombokat engedjük meg a játékosnak, hogy lássa. A játékos élete minden alkalommal maximumra van állítva, új szobák belépésekor.

btnNorth.Visible = (newLocation.LocationToNorth != null);

btnEast.Visible = (newLocation.LocationToEast != null);

btnSouth.Visible = (newLocation.LocationToSouth != null);

btnWest.Visible = (newLocation.LocationToWest != null);

\_player.CurrentHitPoints = \_player.MaximumHitPoints;

lblHitPoints.Text = \_player.CurrentHitPoints.ToString();

Az új helyszínre való belépéskor először ellenőrizzük, hogy elérhető-e a megadott helyszínen egy küldetés. Ha igen megnézzük, hogy a játékos megkapta-e már a küldetést, valamint, hogy teljesítette-e már. Ha megkapta, de nem teljesítette, leellenőrizzük, hogy be tudja-e fejezni. Ha igen, elvesszük a teljesítéshez szükséges tárgyakat, hozzáadjuk a játékos Inventory-ba jutalmát, valamint teljesítettnek jelöljük a küldetést. Ha nem kapta még meg a küldetést, akkor ezt hozzáadjuk a játékos küldetéseinek listájához. A végeredményt minden esetben közöljük a játékossal. (Ezek ismét a “Player” klasszban található alprogramokkal történnek)

//van-e quest

if (newLocation.QuestAvailableHere != null)

{

//megvan-e a quest, teljesitve van-e

bool playerAlreadyHasQuest = \_player.HasThisQuest(newLocation.QuestAvailableHere);

bool playerAlreadyCompletedQuest = \_player.CompletedThisQuest(newLocation.QuestAvailableHere);

//megkapta mar a questet

if (playerAlreadyHasQuest)

{

//ha nincs befejezve

if (!playerAlreadyCompletedQuest)

{

//be tudja fejezni?

bool playerHasAllItemsToCompleteQuest = \_player.HasAllQuestCompletionItems(newLocation.QuestAvailableHere);

//Megvan minden ami kell

if (playerHasAllItemsToCompleteQuest)

{

//uzenet kiirasa

rtbMessages.Text += Environment.NewLine;

rtbMessages.Text += "You complete the '" + newLocation.QuestAvailableHere.Name + "' quest." + Environment.NewLine;

//itemek elvevese

\_player.RemoveQuestCompletionItems(newLocation.QuestAvailableHere);

//jutalom adasa

rtbMessages.Text += "You recieve: " + Environment.NewLine;

rtbMessages.Text += newLocation.QuestAvailableHere.RewardExperiencePoints.ToString() + " experience points" + Environment.NewLine;

rtbMessages.Text += newLocation.QuestAvailableHere.RewardGold.ToString() + " gold" + Environment.NewLine;

rtbMessages.Text += newLocation.QuestAvailableHere.RewardItem.Name + Environment.NewLine;

rtbMessages.Text += Environment.NewLine;

\_player.ExperiencePoints += newLocation.QuestAvailableHere.RewardExperiencePoints;

\_player.Gold += newLocation.QuestAvailableHere.RewardGold;

//jutalom hozzaadasa az inventoryhoz

\_player.AddItemToInventory(newLocation.QuestAvailableHere.RewardItem);

//quest megkeresese, befejezettnek jelolese

\_player.MarkQuestCompleted(newLocation.QuestAvailableHere);

}

}

}

else

{

//nincs meg meg a quest, uzenet kiirasa

rtbMessages.Text += "You recieved the " + newLocation.QuestAvailableHere.Name + "quest." + Environment.NewLine;

rtbMessages.Text += newLocation.QuestAvailableHere.Description + Environment.NewLine;

rtbMessages.Text += "Come back with these:" + Environment.NewLine;

foreach (QuestCompletionItem qci in newLocation.QuestAvailableHere.QuestCompletionItems)

{

if (qci.Quantity == 1)

{

rtbMessages.Text += qci.Quantity.ToString() + " " + qci.Details.Name + Environment.NewLine;

}

else

{

rtbMessages.Text += qci.Quantity.ToString() + " " + qci.Details.NamePlural + Environment.NewLine;

}

}

rtbMessages.Text += Environment.NewLine;

//hozzaadjuk a questekhez ezt a questet

\_player.Quests.Add(new PlayerQuest(newLocation.QuestAvailableHere));

A következő lépés leellenőrizni, hogy találunk-e szörnyet az új helyszínen. Ha van a helyszínen szörny, akkor ennek a másolatát behelyezzük a “\_currentMonster” változóba, valamint megjelenítjük a játékos fegyvereinek és bájitalainak listáját, valamint használatukhoz szükséges gombot

if (newLocation.MonsterLivingHere != null)

{

rtbMessages.Text += "You see a " + newLocation.MonsterLivingHere.Name + Environment.NewLine;

//uj szorny gyartasa

Monster standardMonster = World.MonsterByID(newLocation.MonsterLivingHere.ID);

\_currentMonster = new Monster(standardMonster.ID, standardMonster.Name, standardMonster.MaximumDamage, standardMonster.RewardExperiencePoints, standardMonster.RewardGold, standardMonster.CurrentHitPoints, standardMonster.MaximumHitPoints);

foreach (LootItem lootItem in standardMonster.LootTable)

{

\_currentMonster.LootTable.Add(lootItem);

}

cboWeapons.Visible = true;

cboPotions.Visible = true;

btnUsePotion.Visible = true;

btnUseWeapon.Visible = true;

}

else

{

\_currentMonster = null;

cboWeapons.Visible = false;

cboPotions.Visible = false;

btnUseWeapon.Visible = false;

btnUsePotion.Visible = false;

Az előbb említett ellenőrzések után frissítjük a játékos Inventory-ját, küldetéseinek listáját, valamint a fegyver, és a bájitalok listáját, azon célból, hogy a játékos is lássa a megtörtént változások végeredményeit. Az utolsó részében a programnak, azoknak a gomboknak van megírva a kódja amelyekkel a játékos fegyvereit, valamint a bájitalait használja. A fegyver használatánál van alkalmazva a random szám generátor is (az ellenféltől levett életpontok megadására) valamint itt ellenőrizzük, hogy a szörny él-e még, a játékos támadását követően, ha nem, megadjuk a a szörny legyőzéséért járó jutalmat (random szám generátorral van kisorsolva a jutalom), ha viszont él még a szörny, ennek a visszatámadását fogjuk látni. Ha ennek következtében a játékos meghal, hazavisszük, az elsö helyszínre, ha mindketten életben marad, folytatódik a csata.

//frissitjuk az inventory listat

UpdateInventoryListInUI();

//frissitjuk a quest listat

UpdateQuestListInUI();

//a fegyver combox frissitese

UpdateWeaponListInUI();

//a potion combobox frissitese

UpdatePotionListInUI();

}

private void UpdateInventoryListInUI()

{

dgvInventory.RowHeadersVisible = false;

dgvInventory.ColumnCount = 2;

dgvInventory.Columns[0].Name = "Name";

dgvInventory.Columns[0].Width = 197;

dgvInventory.Columns[1].Name = "Quantity";

dgvInventory.Rows.Clear();

foreach (InventoryItem inventoryItem in \_player.Inventory)

{

if (inventoryItem.Quantity > 0)

{

dgvInventory.Rows.Add(new[] { inventoryItem.Details.Name, inventoryItem.Quantity.ToString() });

}

}

}

private void UpdateQuestListInUI()

{

dgvQuests.RowHeadersVisible = false;

dgvQuests.ColumnCount = 2;

dgvQuests.Columns[0].Name = "Name";

dgvQuests.Columns[0].Width = 197;

dgvQuests.Columns[1].Name = "Done?";

dgvQuests.Rows.Clear();

foreach (PlayerQuest playerQuest in \_player.Quests)

{

dgvQuests.Rows.Add(new[] { playerQuest.Details.Name, playerQuest.IsCompleted.ToString() });

}

}

private void UpdateWeaponListInUI()

{

List<Weapon> weapons = new List<Weapon>();

foreach (InventoryItem inventoryItem in \_player.Inventory)

{

if (inventoryItem.Details is Weapon)

{

if (inventoryItem.Quantity > 0)

{

weapons.Add((Weapon)inventoryItem.Details);

}

}

}

if (weapons.Count == 0)

{

//nincs fegyver,elrejtjuk a comboboxot es a gombot

cboWeapons.Visible = false;

btnUseWeapon.Visible = false;

}

else

{

cboWeapons.DataSource = weapons;

cboWeapons.DisplayMember = "Name";

cboWeapons.ValueMember = "ID";

cboWeapons.SelectedIndex = 0;

}

}

private void UpdatePotionListInUI()

{

List<HealingPotion> healingPotions = new List<HealingPotion>();

foreach (InventoryItem inventoryItem in \_player.Inventory)

{

if (inventoryItem.Details is HealingPotion)

{

if (inventoryItem.Quantity > 0)

{

healingPotions.Add((HealingPotion)inventoryItem.Details);

}

}

}

if (healingPotions.Count == 0)

{

//nincs heal potion, elrejt combobox es gomb

cboPotions.Visible = false;

btnUsePotion.Visible = false;

}

else

{

cboPotions.DataSource = healingPotions;

cboPotions.DisplayMember = "Name";

cboPotions.ValueMember = "ID";

cboPotions.SelectedIndex = 0;

}

private void btnUseWeapon\_Click(object sender, EventArgs e)

{

//a kivalasztott fegyvert hasznaljuk

Weapon currentWeapon = (Weapon)cboWeapons.SelectedItem;

//hany hp-t veszunk le

int damageToMonster = RandomNumberGenerator.NumberBetween(currentWeapon.MinimumDamage, currentWeapon.MaximumDamage);

//levesszuk a szornytol a hp-t

\_currentMonster.CurrentHitPoints -= damageToMonster;

rtbMessages.Text += "You hit the " + \_currentMonster.Name + " for " + damageToMonster.ToString() + "points" + Environment.NewLine;

//halott-e a szorny

if(\_currentMonster.CurrentHitPoints <= 0)

{

//kiirjuk hogy meghalt

rtbMessages.Text += Environment.NewLine;

rtbMessages.Text += "You defeated the " + \_currentMonster.Name + Environment.NewLine;

//jutalom megadasa

\_player.ExperiencePoints += \_currentMonster.RewardExperiencePoints;

rtbMessages.Text += "You recieve " + \_currentMonster.RewardExperiencePoints.ToString() + "experience points" + Environment.NewLine;

\_player.Gold += \_currentMonster.RewardGold;

rtbMessages.Text += "You recieve " + \_currentMonster.RewardGold.ToString() + " gold" + Environment.NewLine;

//random loot megadasa

List<InventoryItem> lootedItems = new List<InventoryItem>();

//hozzaadjuk a looteditme listahoz az itemeket, amiket megkapunk

foreach(LootItem lootItem in \_currentMonster.LootTable)

{

if(RandomNumberGenerator.NumberBetween(1, 100) <= lootItem.DropPercentage)

{

lootedItems.Add(new InventoryItem(lootItem.Details, 1));

}

}

//ha nem kaptunk itemet megadjuk a defaultot

if(lootedItems.Count == 0)

{

foreach(LootItem lootItem in \_currentMonster.LootTable)

{

if(lootItem.IsDefaultItem)

{

lootedItems.Add(new InventoryItem(lootItem.Details, 1));

}

}

}

//looteditems hozzaadasa az inventoryhoz

foreach(InventoryItem inventoryItem in lootedItems)

{

\_player.AddItemToInventory(inventoryItem.Details);

if(inventoryItem.Quantity == 1)

{

rtbMessages.Text += "You looted " + inventoryItem.Quantity.ToString() + " " + inventoryItem.Details.Name + Environment.NewLine;

}

else

{

rtbMessages.Text += "You looted " + inventoryItem.Quantity.ToString() + " " + inventoryItem.Details.NamePlural + Environment.NewLine;

}

}

//inventory es player statok frissitese

lblHitPoints.Text = \_player.CurrentHitPoints.ToString();

lblGold.Text = \_player.Gold.ToString();

lblExperience.Text = \_player.ExperiencePoints.ToString();

lblLevel.Text = \_player.Level.ToString();

UpdateInventoryListInUI();

UpdateWeaponListInUI();

UpdatePotionListInUI();

rtbMessages.Text += Environment.NewLine;

//visszavisszuk a jelenlegi helyre, hogy healeljuk, es uj szrnyet spannoljunk

MoveTo(\_player.CurrentLocation);

}

else

{

//a szorny meg el

//hany hp-t veszit a jatekos

int damageToPlayer = RandomNumberGenerator.NumberBetween(0, \_currentMonster.MaximumDamage);

rtbMessages.Text += "The " + \_currentMonster.Name + " did " + damageToPlayer.ToString() + " points of damage" + Environment.NewLine;

//hp levonasa

\_player.CurrentHitPoints -= damageToPlayer;

lblHitPoints.Text = \_player.CurrentHitPoints.ToString();

if(\_player.CurrentHitPoints <= 0)

{

rtbMessages.Text += "The " + \_currentMonster.Name + " killed you" + Environment.NewLine;

//hazavisszuk a jatekost

MoveTo(World.LocationByID(World.LOCATION\_ID\_HOME));

}

}

}

private void btnUsePotion\_Click(object sender, EventArgs e)

{

//a kivalasztott potiont haznaljuk

HealingPotion potion = (HealingPotion)cboPotions.SelectedItem;

//heal

\_player.CurrentHitPoints = (\_player.CurrentHitPoints + potion.AmountToHeal);

//nem lehet tobb mint max hp

if(\_player.CurrentHitPoints > \_player.MaximumHitPoints)

{

\_player.CurrentHitPoints = \_player.MaximumHitPoints;

}

//elvesszuk a potiont

foreach(InventoryItem ii in \_player.Inventory)

{

if(ii.Details.ID == potion.ID)

{

ii.Quantity--;

break;

}

}

rtbMessages.Text += "You drink a " + potion.Name + Environment.NewLine;

//szorny tamad

//hany hp-t veszit a jatekos

int damageToPlayer = RandomNumberGenerator.NumberBetween(0, \_currentMonster.MaximumDamage);

rtbMessages.Text += "The " + \_currentMonster.Name + " did " + damageToPlayer.ToString() + " points of damage" + Environment.NewLine;

//hp levonasa

\_player.CurrentHitPoints -= damageToPlayer;

if (\_player.CurrentHitPoints <= 0)

{

rtbMessages.Text += "The " + \_currentMonster.Name + " killed you" + Environment.NewLine;

//hazavisszuk a jatekost

MoveTo(World.LocationByID(World.LOCATION\_ID\_HOME));

}

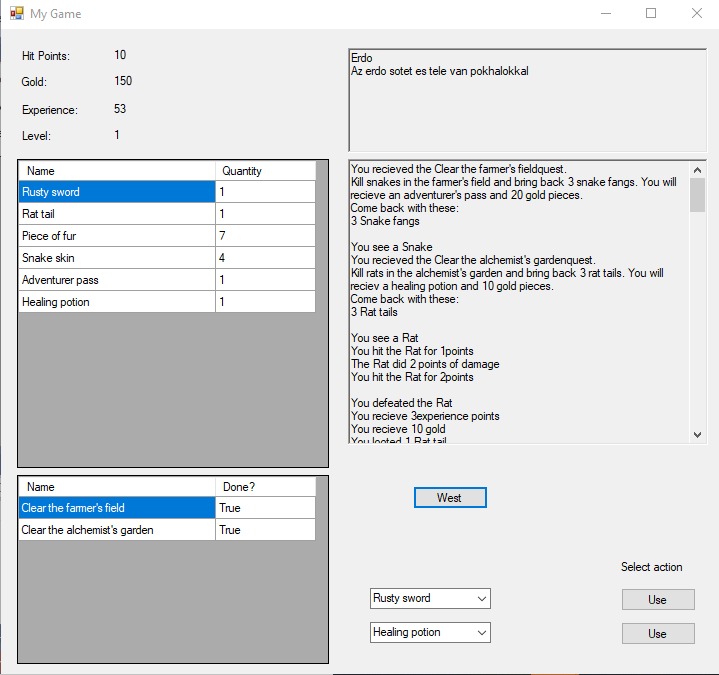
//UI frissites

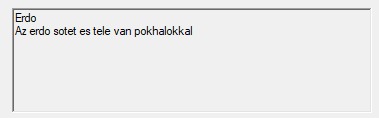
lblHitPoints.Text = \_player.CurrentHitPoints.ToString();

UpdateInventoryListInUI();

UpdatePotionListInUI();}}}

Használati útmutató

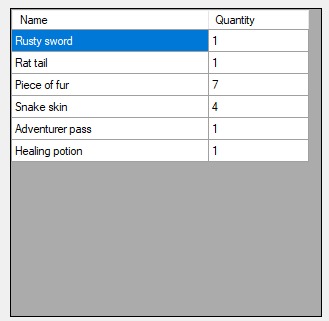
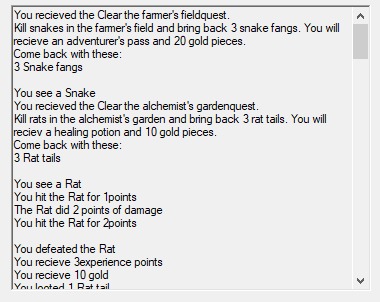
A játék a hős házában kezdődik. A játékos célja, a küldetések teljesítése, az “Erdő”-be való bejutás, valamint az óriás pók legyőzése. Utóbbihoz, a játékosnak le kell gyözzön számos szörnyet, és teljesítenie kell a két küldetést. A játékos szobáról szobár léphet, majd a csaták során küzdhet, bájitalt ihat, vagy elmenekülhet, az által, hogy elhagyja a helyszínt.

 (Ábra 2)

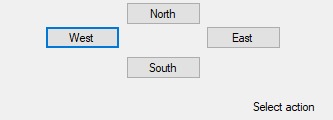
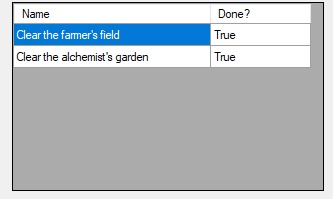
(Ábra 3)

(Ábra 1)

A bal felső sarokban (Ábra 2) a játékos jelenlegi életét, tulajdonában álló aranymennyiséget, a tapasztalat pontjait, valamint a szintjét olvashatjuk le. A jobb felső sarokban (Ábra 3) egy, a játékos jelenlegi helyszíne van kiírva, valamint, erről egy rövid leírás. A képernyő bal oldalán (Ábra 4) a játékos inventory-ja található, mely mint egy hátitáska működik, itt tekintheti meg a játékos a tulajdonában álló tárgyakat, valamint ezek mennyiségét. A képernyő jobb (Ábra 5) oldalán pedig az események vannak leírva, melyek végbe mentek. Ezen ablak segítségével ad visszajelzéseket a játék a játékosnak.

 (Ábra 5)

(Ábra 4)

A képernyő bal alsó sarkában (Ábra 6) a játékos által megszerzett küldetéseket találjuk, valamint, hogy ezek teljesítve vannak-e vagy sem. A jobb alsó sarokban 4 gombot (Ábra 7) találhatunk (North = észak, West = nyugat, East = kelet, South = dél) melyekre klikkelve a játékos az adott irányba mozog. A képernyő legalján (Ábra 8) pedig a küzdés során megjelenő 2 listát valamint gombot láthatjuk. Utóbbival a kiválasztott tárgyakat használjuk.

 (Ábra 7)

(Ábra 6) (Ábra 8)

Bibliográfia

1. <https://docs.microsoft.com/en-us/visualstudio/ide/creating-solutions-and-projects?view=vs-2019>
2. <https://support.microsoft.com/en-us/help/307368/how-to-create-classes-and-objects-in-visual-c>
3. <https://stackoverflow.com/questions/3554658/how-to-use-a-class-from-one-c-sharp-project-with-another-c-sharp-project>
4. <https://stackoverflow.com/questions/41196027/what-is-the-difference-between-void-and-static-void-function-in-c>
5. <https://docs.microsoft.com/en-us/visualstudio/ide/reference/generate-constructor?view=vs-2019>