

Specifikáció

Gitanos

Modern szoftverfejlesztési eszközök, 2022/23 I.

Készítette: Babos Kristóf, Göntér Mátyás, Fazekas Richárd, Lendvai Áron

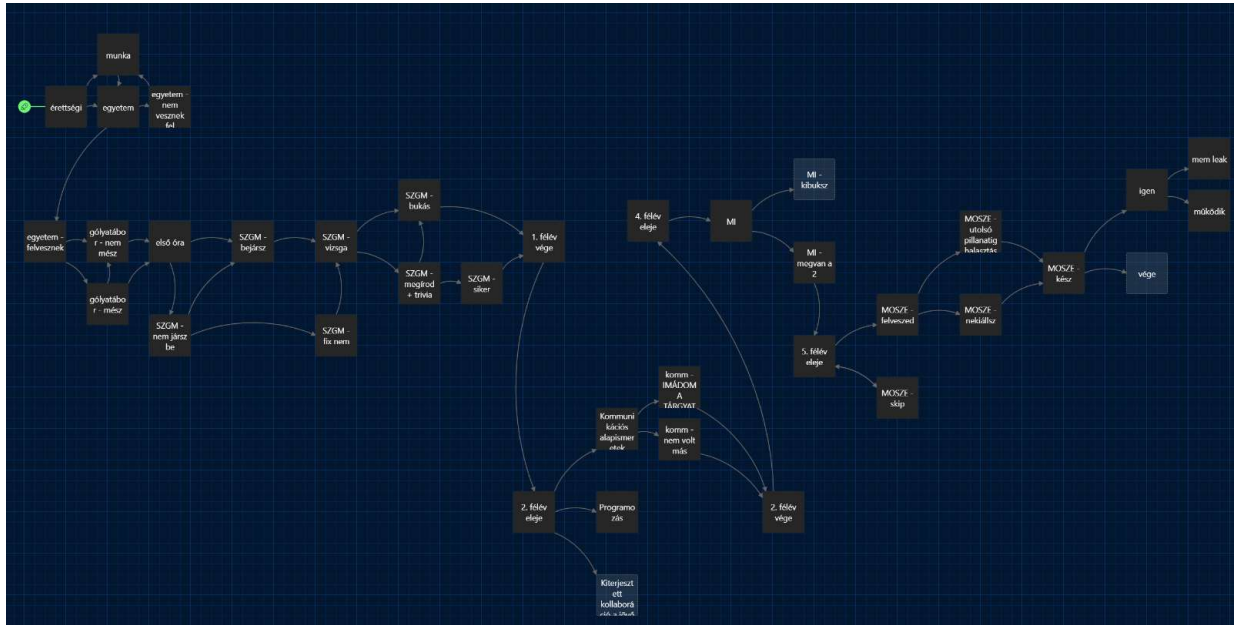


**SZÉCHENYI
EGYETEM**
UNIVERSITY OF GYŐR —

Tartalomjegyzék

Történet	3
Használati esetek (use case)	4
Azonosított követelmények	6
Struktúra (UML class diagram)	7
Branch használat	8
Game Engine V 1.0	8
Game Engine V 2.0	8
Specifikáció	8
Program	10
A program működése	10
A program osztályai	11
A program segédfüggvényei	11
A main() függvény	12
Megvalósítandó feature-k	12
ASCII képek megjelenítése	12
Játékos specifikus tulajdonságok	13
A történet tárolása	13
Fájlformátum	13
Generátor segédprogram	14

Történet



- A játék a győri Széchenyi István Egyetemen belül játszódik, azon belül arra koncentrálódik, hogy hogyan is játszódik le egy átlagos mérnökinformatikus hallgató élete és milyen megpróbáltatásokon kell, hogy keresztül menjen ahhoz, hogy végül a jól megérdemelt diplomáját megszerezhesse
- A játék magában foglalja, hogy a játékos megszerezte az érettségit, majd ezt követően döntenie kell arról, hogy elmegy-e dolgozni, vagy pedig inkább továbbmegy egyetemre tanulni
- Vannak bizonyos szálak amelyek következtében a játék hossza megrövidülhet, vagy meghosszabbodhat
- A cél természetesen az, hogy minél kevesebb idő és lépés alatt megszerezze a játékos a diplomáját
- Évről évre az új tárgyaknak és kihívásoknak köszönhetően, olyan új döntéseket kell meghoznia amelyek eltérő mértékben befolyásolják azokat a metrikákat, melyek kihatnak azokra a funkciókra amelyeket a szerencse befolyásol
- Ha felveszik az egyetemre, akkor dönt arról hogy részt vesz-e a gólyatáborban vagy sem

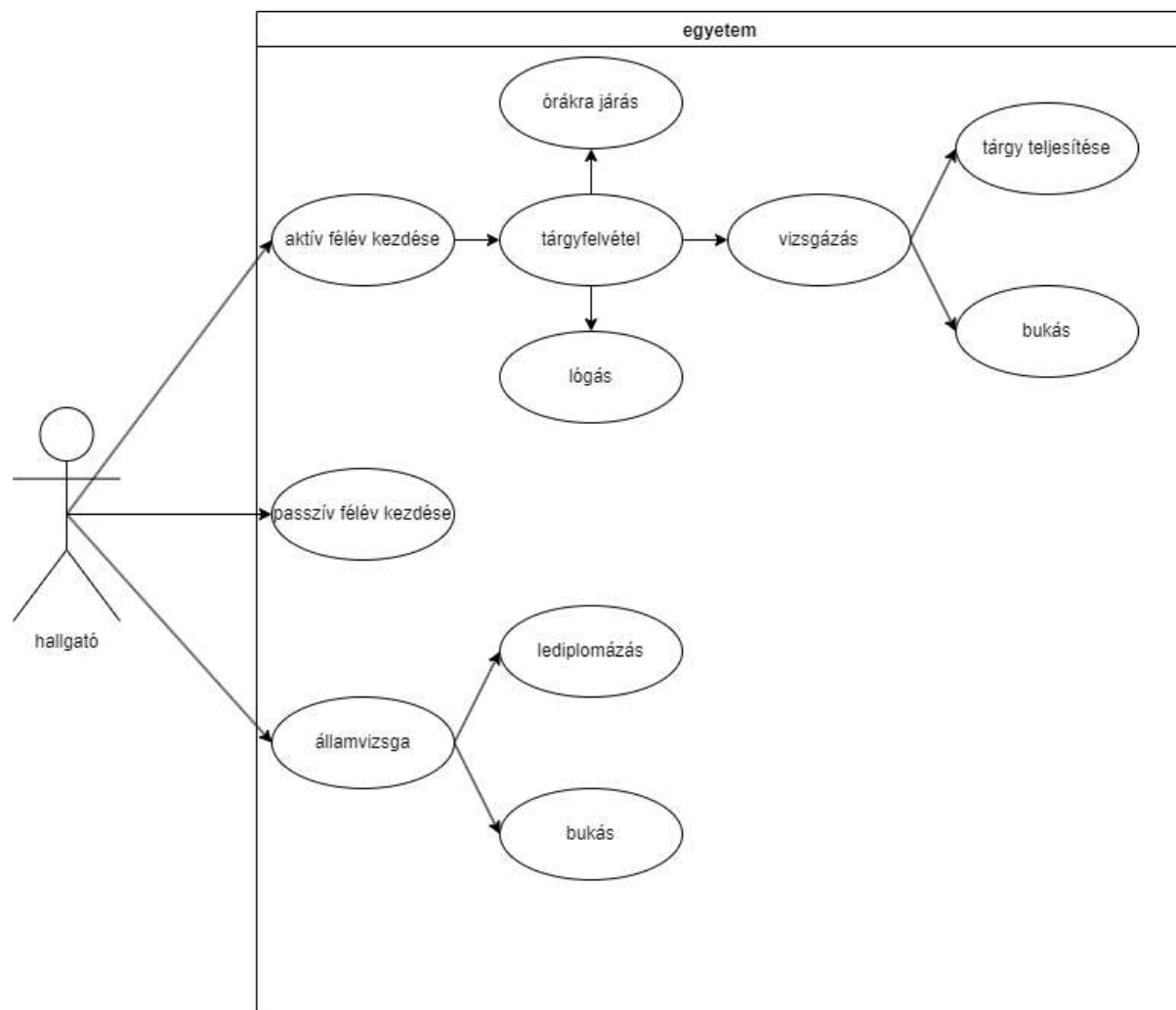
- A gólyatáborban lehetősége van a játékosnak nagy előnyt szerezni azzal, hogy ismeretségeket épít ki. Ez az előny pedig kihat majd az ezt követő tanulmányai során abban a formában, hogy nagyobb eséllyel fog majd átmenni a vizsgákon mint az, aki ezt az ismertségeket nem szerzi meg
- Ezt követően el elérkezünk a játékos megismerkedik és saját tapasztalatot szerez az egyetemi tanítással kapcsolatban és bemegy a legelső órájára
- Az egyetemi pályafutás során vannak specifikus tárgyak, amelyekhez kötődő döntések súlya jelentőséggel bírhat arra, hogy hogy fog végbemenni a folyamat, és hogy milyen hosszú időt fog igénybe venni a diploma megszerzése
 - Ilyen tárgy például az SZGM (Számítógépek működése)
 - Programozás
 - Kommunikációs alapismeretek
 - Modern szoftverfejlesztési eszközök
- Van egy szerencse mechanika is a játékban, amelyre a játékos által hozott döntések direkt hatással bírhatnak.
- Az előbb említett mechanikára sok különböző dolog is kihathat nevezetesen:
 - Szórakozás
 - Az órákra való bejárás vagy nem bejárás (bejárás/lógás)
- A szórakozás kivitelezésének lehet pozitív és negatív hatása is, attól függően hogy a többi változóval milyen kapcsolatban és harmóniában van. Pozitív hatása lehet abból a szempontból hogy az adott játékos ismertséget/hírnevet(popularity) szerezhet, amely ismertséget később felhasználhatja a tanulmányai során. Ezzel szemben lehet negatív hatása is, akkor hogyha a szórakozás limitek nélkül megy végbe és ezzel kevesebb idő és energia marad az órákra való bejárásra és a tanulásra (alacsonyabb lesz a knowledge változó értéke). Tehát összességében a szórakozás mint koncepció hatást gyakorolhat a knowledge és a popularity változókra.
- Természetesen a különböző tárgyak eltérő nehézségűek és eltérő hogy mennyi kreditet kaphatunk azok teljesítéséért.

- Minél több órát hagy ki az egyén aki játszik, annál kisebb lesz az esélye annak hogy átfog menni a vizsgán és ez végül akár bukással is járhat (és sok ugyanazon tárgyból való bukás után újra kell kezdeni a játékot). Erre ki is találtunk egy változót amelynek a neve knowledge.
- Az előbb említett döntési lehetőségek és választások mint például a (szórakozás vagy lógás az órákról) eltérő súlyozást kapnak kontextustól függően. Itt a kontextust tulajdonképpen a tárgy kreditszáma vagy a nehézségi változója adhatja. Tehát például ha a játékos kihagy 2 Számítógépek működése órát, akkor annak súlya nagyobb lesz , mintha ugyanúgy 2 órát lógott volna el Kiterjesztett kollaboráció a jövő Internetén tárgyról.
- Ezen kívül még van egy hangulat változó is, amely megmutatja hogy a hallgató (a játékos) milyen mentális állapotban van egy adott időpontban. Ez befolyással lehet arra, hogy a játékosnak sikerül-e átmennie a vizsgákon, valamint, hogy sikerül-e lediplomáznia és hogy összességében képes lesz-e végigmenni az egyetem megpróbáltatásain.
- Egy másik jelentőséggel bíró játék mechanika pedig nem más, mint a chance. Ennek a számát vagy eredményét 2 másik változó szorzata által kapjuk meg, nevezetesen a popularity és a knowledge szorzatából.
- A játékot akkor tudjuk sikeresen “megnyerni” vagy elvégezni, hogyha Modern szoftverfejlesztési eszközök nevű tárgyat sikeresen teljesítjük, majd pedig elmegyünk államvizsgázni

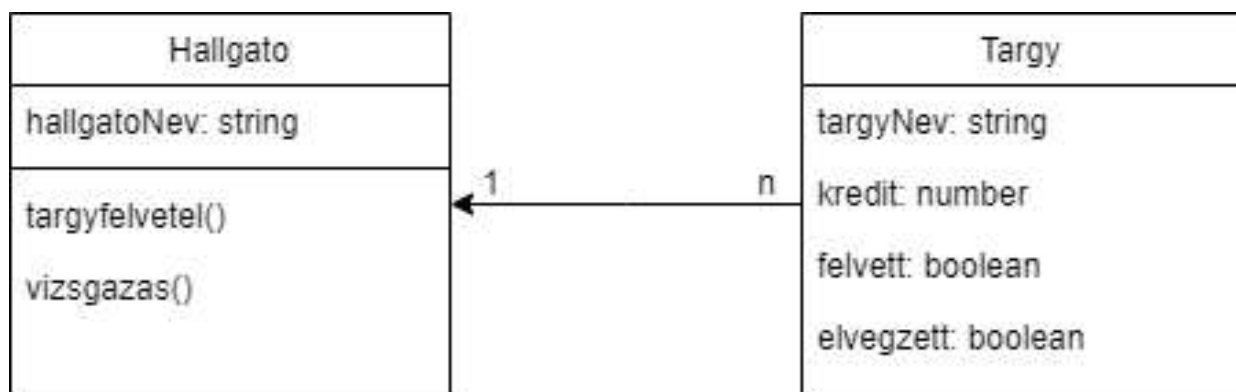
Használati esetek (use case)

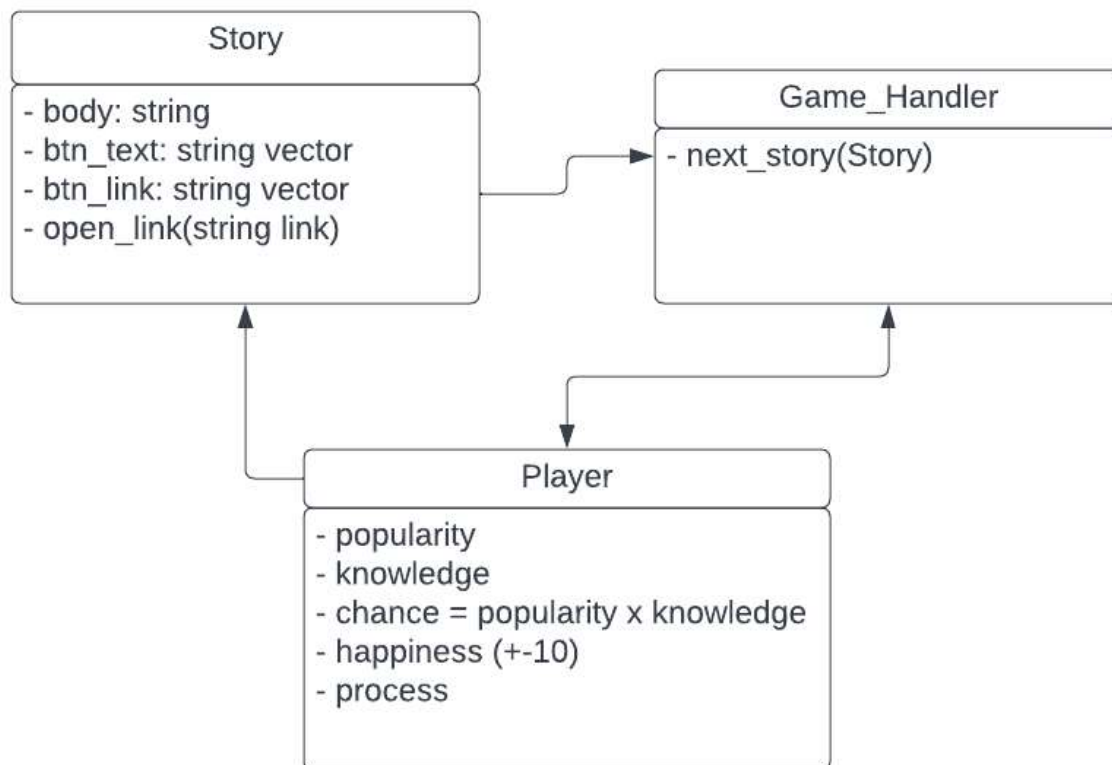
Megnevezés	Prioritás	Leírás
tárgyfelvétel	high	több tárgyból lehet választani, különböző tárgyak különböző nehézségűek
vizsgázás	high	próbálkozás adott tárgy elvégzésével <ul style="list-style-type: none">• kreditszám és előző lépések befolyásolják a sikeresség esélyét• ha sikeres akkor adott tárgy abszolválva• túl sok bukás a játék végét jelentheti
szórakozás	medium	növeli az ismertséget, de lehet a tudás kárára <ul style="list-style-type: none">• rossz döntések akár a játék végét is jelenthetik
órákra járás	low	növeli adott tárgy elvégzésének esélyét
lógás	low	csökkenti adott tárgy elvégzésének esélyét, ismertséget viszont növeli

Use case diagram



Struktúra (UML class diagram)

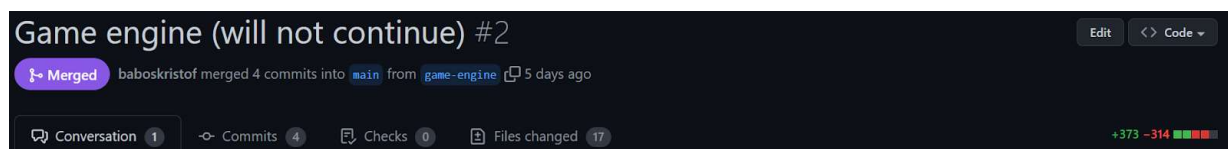




Branch használat

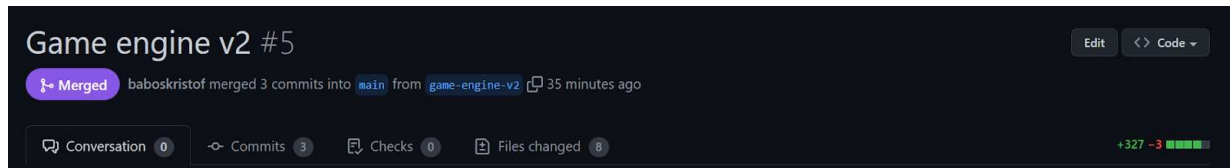
Game Engine V 1.0

Az első koncepció kivitelezéséhez a Windows Forms-al került megvalósításra C++ nyelven. Ez a megjelenítést sokkal elősegítette, viszont nem tudtunk vele dinamikusan fejlődni. Új ötletek ütötték fel a fejüket, ezért ezen ág megszüntetése mellett döntöttünk 3-4 napos munkát átvittünk egy másik projektbe.



Game Engine V 2.0

Az első engine bukása után ebben az ágban kezdtünk el fejleszteni. Lecseréltük a Windows Forms-ot C++ konzol alkalmazásra, melyekbe így beletudjuk integrálni az ASCII karakter megjelenítést. Illetve a felhasználó konzolos formában játszhatja végig a játékunkat. Terveztünk bele egy kisebb Easter Egg-et is, mivel véleményünk szerint minden játékban szükség van arra, amiért kiemelkedik a sorból a többi közül.

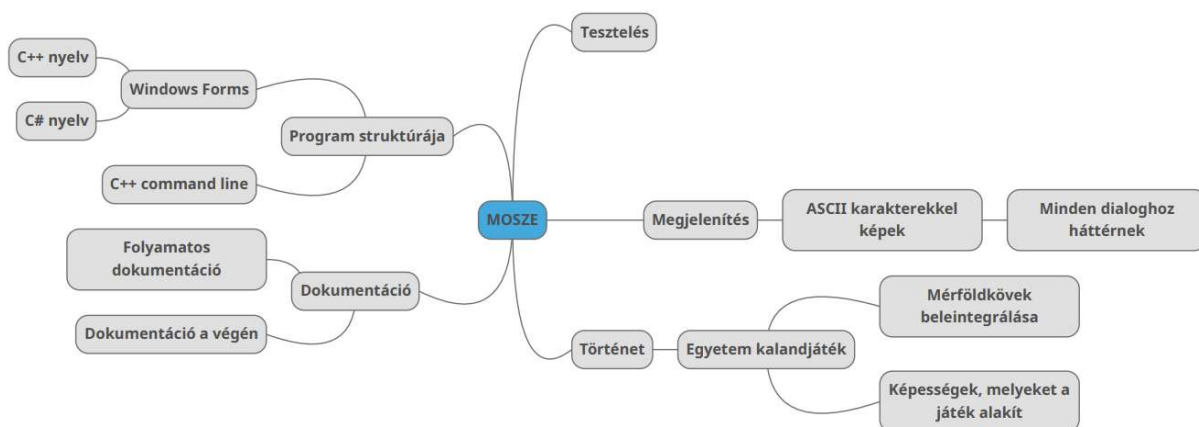


Specifikáció

Ebben az ágban a programhoz készített ábrák előrehaladását tartottuk, melyeket folyamatosan frissítünk. Mindmap-ek, ennek a specifikációnak az adott verziója stb.

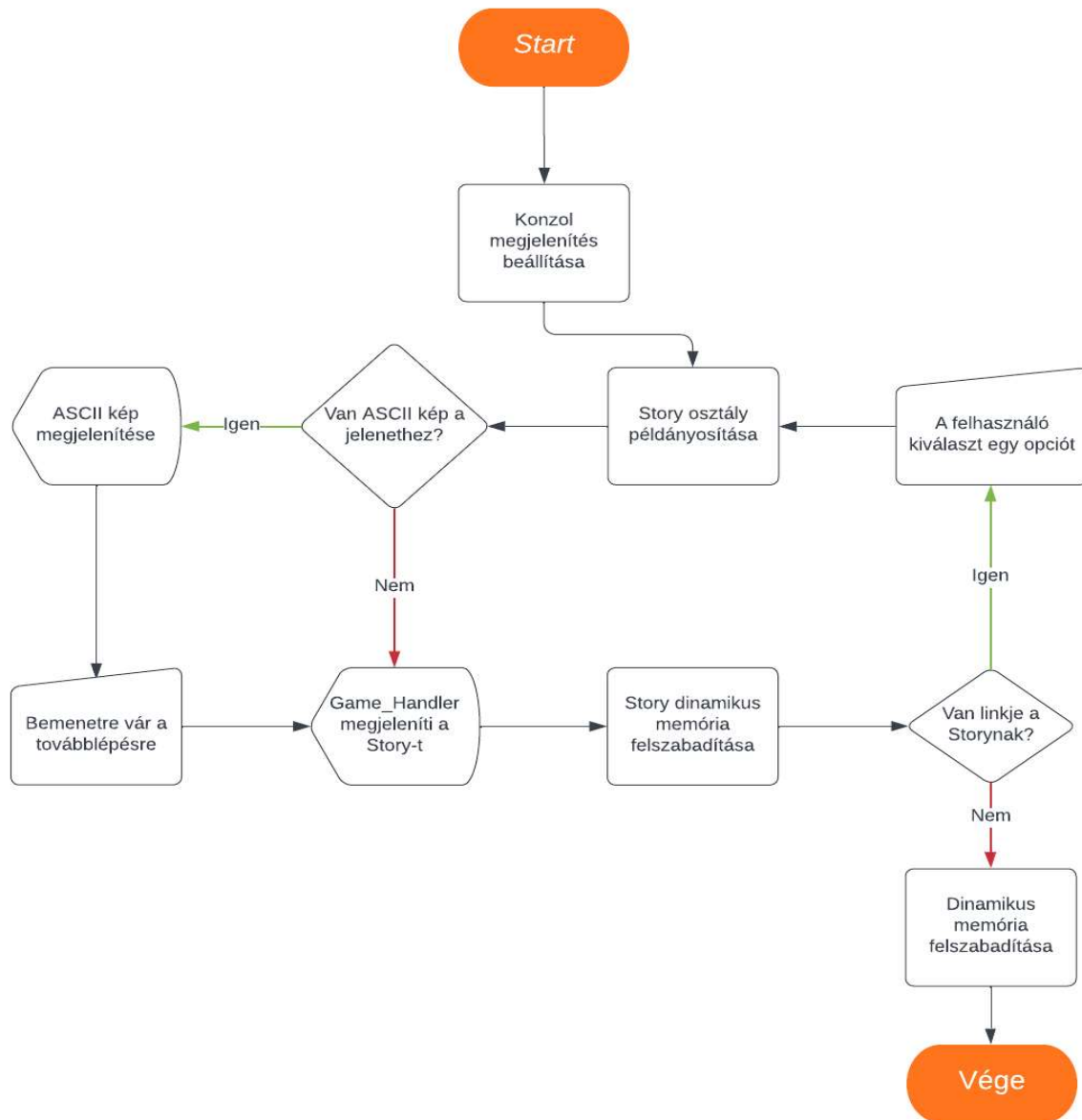
**A MÉRFÖLDKŐ, VAGYIS A SPECIFIKÁCIÓ LEADÁSA ELŐTT EZEK A BRANCHEK
MERGELÉSRE KERÜLTEK!**

Mindmap - ötletek



Program

A program működése



1. A felhasználó elindítja a programot.
2. Megjelenik a kezdő story.
3. A megjelenő opciók közül választ egyet a felhasználó.
4. A kiválasztott opciót továbbítja a kontrollernek.

5. Ismétlődik a 2-4. sorozat a játék végéig.
6. Befejeződik a program.

A program osztályai

`class Story`

- Konstruktora egy link, ami a megnyitandó „game” fájlt jelöli
- Attribútumai:
 - `string` body: a megjelenítendő szövegtörzs
 - `vector<string>` btn_texts: a megjelenítendő „gombok”, opciók
 - `vector<string>` btn_links: az opciókhoz tartozó linkek (a következő story)
- Metódusai:
 - `void` open_link(`string` btn_link): a megadott link alapján megnyitja az adott fájlt – mindenképpen a `../story/[link].game` helyen keresi; beállítja a `btn_texts` és `btn_links` változókat

`class Game_Handler`

- Konstruktora egy `Story`, amit megjelenít
- Metódusai:
 - `void` next_story(`Story` next_story): megjeleníti a `story.body` és a `story.btn_texts` szövegeket

A program segédfüggvényei

`namespace functions:`

`vector<string>` split(`string` s, `string` delimiter): a bejövő s stringet felosztja a delimiter alapján, és a substringeket egy vektorban adja vissza

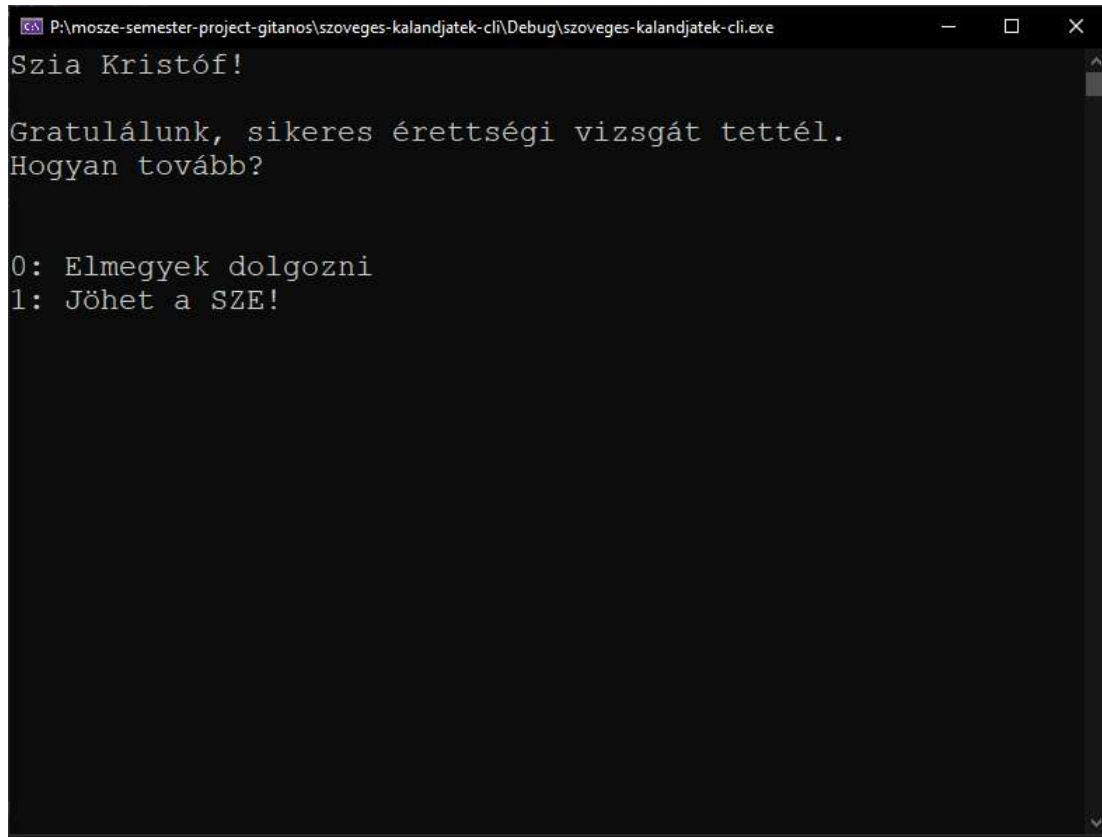
`void` set_console_appearance(): egy előre meghatározott módon beállítja a megjelenítő konzolt

- x helyzet: 100 px
- y helyzet: 100 px
- magasság: 800 px
- szélesség: 600 px
- betűtípus: `FF_DONTCARE`
- betűmagasság: 24 px

A main() függvény

- Inicializálja a lokális változókat: `option` és `link`
- Deklarálja a `link`-et a kezdőstoryra (`_start.game`)
- Meghívja a `set_console_appearance()` függvényt
- Létrehoz dinamikusan egy `s` storyt `link` alapján és egy `g` game handlert `s` alapján
- Bekéri a felhasználótól az `option`-t
- Átírja `link`-et a kiválasztott linkre: `btn_links[option]`
- Felszabadítja az `s` memóriaterületen lévő storyt, így megfékezve a memory leaket
- Törli a konzol képernyőjét: `system("CLS");`
- Felülírja `s`-t egy dinamikusan létrehozott story-val a `link` alapján
- Meghívja a `g` game handler `next_story` metódusát, így kiírja a következő story szövegét és opcióit
- Mindezeket megismétli egészen addig, amíg tart a játék, vagyis van következő link
- Felszabadítja a `g` memóriaterületet

Teszt program



```
P:\mosze-semester-project-gitanos\szoveges-kalandjatek-cli\Debug\szoveges-kalandjatek-cli.exe
Szia Kristóf!

Gratulálunk, sikeres érettségi vizsgát tettél.
Hogyan tovább?

0: Elmegyek dolgozni
1: Jöhet a SZE!
```

A program kinézetének terve látható az ábrán. Megjelenik a szövegtörzs, majd két sortörés, utána pedig az összes kiválasztható opció.

A játékos a 0 vagy 1 karakterek beírásával tud választani.

Megvalósítandó feature-k

User input validation

A felhasználói bemeneteket ellenőrzi egy függvény, ezzel biztosítva, hogy a program akadálymentesen és hibamentesen futhasson. Vagy egy hibaüzenetet ír ki, vagy pedig egyszerűen nem történik semmi, amíg nem kapunk helyes bemenetet.

ANSI/UTF-8 kódolás

A magyar betűk miatt mindenképp támogatni kell az ékezetes karaktereket, így lehetőség szerint az alapértelmezett ISO-8859-1 kódolás helyett ANSI vagy UTF-8 kódolást szükséges használni a megjelenítésnél.

ASCII képek megjelenítése

Implementálandó feladat az ASCII képek feldolgozására és megjelenítésére szolgáló osztály. Lehetséges módja ennek, hogy a Game_Handler osztályt módosítsuk, refaktoráljuk. Másik lehetséges módja egy teljesen különálló interfész, amit meghív a Game_Handler adott esetekben. Még eldöntendő, hogy a .visual fájlok meghívását mikor és hogyan oldjuk meg, ez nagymértékben függ az elkészítendő kód megvalósításától, mivel ezt egy külön, extra funkcióként szeretnénk hozzáadni a programhoz.

Felmerül az is, hogy a felhasználó kiválaszthatja, hogy szeretne-e egyáltalán “képeket” megjeleníteni. Ezt a rendelkezést elmenthetnénk egy felhasználó specifikus fájlba.

Játékos specifikus tulajdonságok

A fent említett fájlban továbbá szerepelhetne, hogy a felhasználó hol tart a játékban (progress), milyen opciókat választott ki, milyen karakterisztikával rendelkezik - például mekkora a szerencse faktora, az ismertsége. Tehát szeretnénk, ha a játékot nem csak a döntések, hanem a véletlenszerűség is befolyásolná mérsékelt módon; a döntések alapján.

Példaként: ha a játékos nem megy el gólyatáborba, akkor kisebb lesz az ismertsége -> vizsgálhoz kevesebb anyagot tud szerezni -> nem lesz akkora esélye jó jegyet szerezni. Vagy ha a játékos egyáltalán nem jár be órákra -> sokkal kisebb esélye lesz átmenni egy tárgyon, még ha a trivia kérdést jól is válaszolja meg.

A történet tárolása

Fájlformátum

Minden story elemet külön fájlban tárolunk a ./story mappában. A fájlok neve a következő:

[story_name].game

Az elemek a következőképp épülnek fel:

[Többsoros szöveg,

“Body Text”]

[[opció1]>[link1]

[[opció2]>[link2]

([[opció3]>[link3])

A többsoros szöveg, amit szabadon írunk. Ez lesz a body text később a programban, ezzel találkozunk a felhasználók először.

Az opciók a választási lehetőségek, amiket a felhasználó választhat. Ha az input 0, akkor az első opciót választja, ha 1, akkor a másodikat, ha 2, akkor a harmadikat.

Egy link vezethet egy .game vagy egy .visual fájlhoz. Ha az előbbi teljesül, akkor egy újabb story elemet kapunk, ha az utóbbi, akkor egy ASCII kép jelenik meg, amit billentyű megnyomásával tud továbbléptetni a felhasználó.

A .visual fájlnak nincs különleges felépítése, csak a megjelenítendő karakterláncot tartalmazza, amit egy külön controller osztály fog feldolgozni és megjeleníteni.

Generátor segédprogram

Segítségként írtunk egy story generátor programot, ami user inputból automatikusan elkészíti a megadott ".game" fájlokat. Beadjuk a fájl nevét, a story szövegét, az opciók szövegét és linkjét, majd ha üres stringet talál, előről kezdődik.

Hátránya, hogy a szövegekbe nem tudunk sortörést írni (egyelőre, megoldás későbbre: a \n karaktorsor helyettesítése whitespace-el), illetve hogy csak a program megszakításával tudunk kilépni, mivel az egyszerűség kedvéért végtelen while ciklusban fut.

```
PS P:\mosze-semester-project-gitano> python story\generator.py
szöveg: teszt szöveg egy!
fájlnév: test1
opció 1 text: opció1
opció 1 link: op1_link
opció 2 text: opció2
opció 2 link: op2_link
opció 3 text:
szöveg: teszt szöveg kettő?!
fájlnév: test2
opció 1 text: 
```