2. Követelmény, projekt, funkcionalitás

2.1 Bevezetés

2.1.1 Cél

Dokumentumunk célja a projekt kimerítő specifikálása. Igyekszünk a feladatkiírásból ki nem derülő részeket pontosítani és ezeket lerögzíteni. Mindezek mellett próbálunk egy használható alapot létrehozni arra vonatkozólag, hogy a fejlesztők között elkerüljük az esetleges félreértéseket, vagyis megegyező szavak esetén ugyanarra gondoljon mindenki.

2.1.2 Szakterület

Ez a program egy játék, aminek célja a szórakoztatás, illetve a játékos egyes képességeinek fejlesztése. Ilyen képességek köze tartozik a gyors reakció, logisztikai képesség és problémamegoldás.

2.1.3 Definíciók, rövidítések

Definíció/rövídítés	**Jelentés**
IIT	Irányítástechnika és Informatika Tanszék
BME	Budapesti Műszaki és Gazdaságtudományi Egyetem
UML	Unified Modelling Language, szabványos modellezési nyelv, amivel a különböző diagramokat készítjük el
StarUML	Modellező eszköz
IntelliJ IDEA	Szoftverfejlesztő eszköz
Eclipse	Szoftverfejlesztő eszköz
Java	A feladathoz használt programozási nyelv
Git/GitHub	Verziókezelő eszköz
Slack	Kommunikációs eszköz, amely elősegíti a csapat együttműködését
Gimp/Paint	Képszerkesztő eszközök
Plug-in	Egy olyan szoftver komponens, amely egy létező programot, valamilyen plusz funkcióval lát el
Offtopic	A fő témától eltérő

Microsoft Word	Szövegszerkesztő eszköz
Google Docs	Szövegszerkesztő eszköz, a dokumentumok közös szerkesztését teszi lehetővé
BusyBot	Slacken belül a feladatmenedzsmentet megkönnyítő bot.
GitHub bot	A GitHub repositorynkban történő változásokat továbbítja a csapattagoknak Slack üzenet formájában.
RUP	Rational Unified Process, iteratív szoftverfejlesztési folyamat
GUI	Graphical User Interface, vagyis grafikus felhasználói felület, a program és a felhasználó közti kapcsolatot teremti meg képek, szövegek, rajzok formájában
JRE 1.8	Java Runtime Environment 1.8-as verziója
JDK 1.8	Java Development Kit, programozási eszközök széles skáláját tartalmazza
Repository	Tároló, itt vannak elhelyezve a verzió követő rendszerben tárolt fájlok
Iteratív, iteráció	Egy olyan folyamat egy lépése, amely a célját egy algoritmus többszöri, egymás utáni lefutásával éri el. Általában minden lépés után közelebb jutunk az eredményhez. Nem kell, hogy köze legyen a rekurzióhoz.
Absztrakció	Hétköznapi használatban elvonatkoztatásnak fordítják.
User	Más szóval a felhasználó.
Szkeleton	A projekt vázát képező elem.
Logisztika	Logisztika az energia, személyek, anyagok, alapanyagok, félkész- és késztermékek, Információk rendszeren belüli és rendszerek közötti áramlásának tervezésével, vezérlésével, szabályozásával, ellenőrzésével megvalósításával foglalkozó menedzsment-szemlélet, melynek célja a folyamathoz járuló összköltség és a vevőkiszolgálás színvonala közötti optimális elérése.
Problémamegoldás	Olyan képesség, amely elősegíti az adott személyt az előtte álló problémák ésszerű és hatékony kezelésében.

2.1.4 Hivatkozások

Feladatkiírás: https://www.iit.bme.hu/targyak/BMEVIIIAB02/feladat

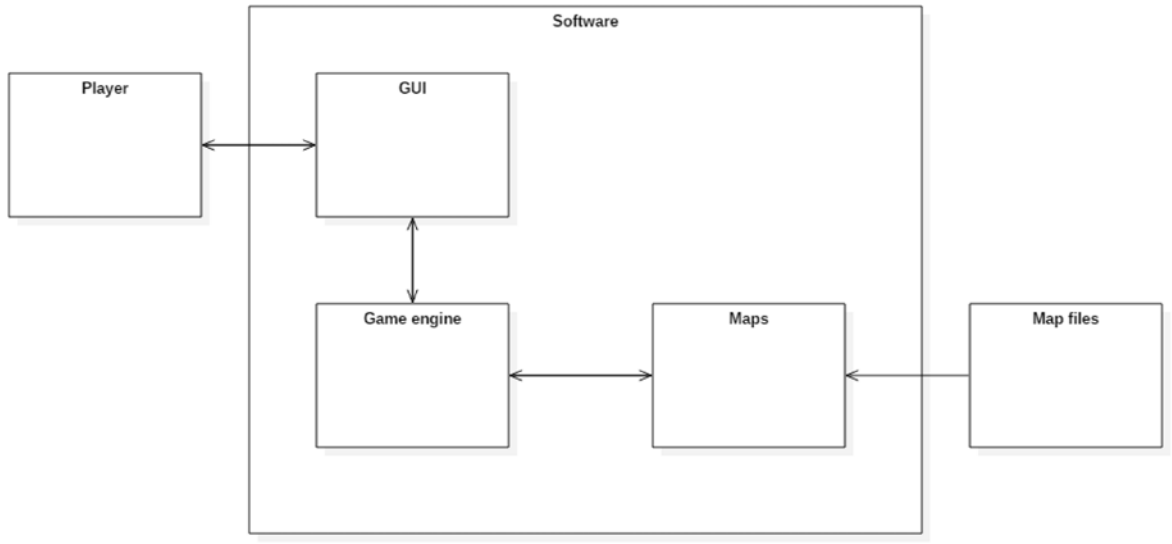
2.1.5 Összefoglalás

A dokumentum a következőket tartalmazza:

- Áttekintés, melynek célja egy magas absztrakciós szintről jellemezni a programot.
- Követelmények, ezeknek kell megfelelnie az elkészült programnak.
- Használati esetek, az összes olyan interakció ami előfordulhat a felhasználó és a program között.
- Egy szótár, mely segít az implementáció függő fogalmak körül a ködöt eloszlatni.
- A szoftver elkészítésére vonatkozó terveink.

2.2 Áttekintés

2.2.1 Általános áttekintés



Game engine (Játékmotor)
Ez az elem mondható a szoftver szívének, ez működteti a játékot, valamint ez felelős a játék funkcióinak lebonyolításáért. Kommunikál a Maps alrendszerrel, ha pálya betöltésére van szüksége, kérés-válasz formában. Továbbá kommunikál a GUI alrendszerrel is, amin keresztül kapja a felhasználó által bevitt parancsokat. majd az új, a játékos által kiadott parancsnak megfelelő állapotot visszatölti a GUI-ba, amit a GUI megjelenít.

Maps (Pályák)
Kiszolgálja a Game engine-t a pályák betöltésével.

GUI (Grafikus Felhasználói Felület)
A felhasználó és a szoftver közötti interakció megvalósításában nyújt segítséget.

Map files (Pálya fájlok)
Ezek olyan fájlok, amiben az alapértelmezett illetve az elmentett pályákat tárolja a program.

ha a játékos új játékot indít, vagy egy elmentettet tölt be, akkor a Maps alrendszer innen kapja az adatokat, amit továbbít a Game engine számára.

Player (Játékos)
A játékos grafikus felületen keresztül kommunikál a programmal, parancsokat ad annak úgyszintén a GUI-n keresztül, amiket a játékmotor kezel le és a változásokat megjeleníti a GUI-n.

2.2.2 Funkciók

A feladat egy, a felhasználó által vezérelt alkalmazás, egy logikai játékprogram elkészítése. Ebben a részben a játék felépítését, a játékmenetet tanulmányozzuk, anélkül, hogy a megvalósítás részleteit taglalnánk.

A játék alapegysége a pálya, ami mi esetünkben egy terepasztal. Ezen (illetve ennek alegységein) kívül más látható vagy vezérelhető terület nem létezik.

A terepasztalon különböző tárgyak helyezkedhetnek el, amik a játék lefolyását befolyásolják:

- Az első ilyen a **vonat** (szerelvény). A vonatot a felhasználó nem tudja sem létrehozni, sem irányítani. A pálya széleiről ezek véletlen időközönként érkeznek, és folyamatosan haladnak a pályán.
- A vonatok **kocsik**ból (vagonokból) állnak, az egyes kocsik különböző színűek lehetnek. Ha az összes utas leszáll egy kocsiról, akkor az elveszti színét. A játékban jelentőséggel bír a kocsik mozdonytól való távolsága is (lásd később).
- A szerelvények **sín**eken közlekednek. Ezek a pálya létrehozásakor generálódnak le, nem helyezhetőek át, nem módosíthatóak.
- A valóságnak megfelelően a vonatok **állomások** között haladnak. Az állomásoknak is van színe. Ehhez tartozik egy megkötés a játékban: csak olyan utasok szállhatnak le az egyes állomásokon, amelyek kocsijának a színe megegyezik az állomás színével.

Ezek voltak azok az elemek, amelyeket a felhasználó nem befolyásolhat. Most nézzük meg azokat, amelyekre van befolyása:

- Két sínpár találkozásakor el kell dönteni, hogy a vonat melyik irányba haladjon tovább. Erre szolgálnak a **váltók**. A user saját belátása szerint bármikor változtathatja ezek állását, ezzel befolyásolva a szerelvények útvonalát. A váltók kettős szereppel rendelkeznek: lehet használni őket az utasok lerakásához szükséges útvonal kiválasztására, illetve két vonat ütközésének elkerülésére. Lehetséges a váltót olyan állapotba is állítani, hogy egy vonat ne tudjon rajta áthaladni (pl.: hármas kereszteződésnél), ekkor az a vonat, aminek nincs biztosítva haladási irány, felrobban.
- A terepasztalon **alagut**at lehet építeni és megszüntetni. Az alagút két bejáratból áll, amit a játékos helyez el a pályán. Megkötés azonban, hogy a játékban egyszerre csak egy aktív alagút lehet. A bejáratok csak az arra kijelölt helyeken helyezhetőek el, nem lehet tetszőleges pontra rakni őket. Ha egy alagútban épp halad egy vonat, akkor a játékosnak nincs lehetősége az alagutat megszüntetni. Ha a vonat bemegy az egyik bejáraton, akkor a másikon fog kijönni. A vonat az alagútban is az addigi sebességével halad, illetve abba az irányba megy tovább az alagút elhagyása után, mint amilyen irányba ment a behajtás előtte. Az alagutak lehetnek rövidebbek is, mint a vonat, ekkor a játékos a szerelvénynek csak azt a részét látja, ami nincs benne az alagútban. Az alagút hasonló szerepet tölt be, mint a váltók. Ha menet közben a vonatnak van lehetősége behajtani egy alagútba, akkor azt minden esetben meg is teszi (az alagút a

preferált haladási irány).

Mint minden játékban, itt is van egy cél. A pályára beérkező utasokat el kell juttatni a nekik megfelelő állomásokhoz. Ha egy vonat áthalad egy állomáson, akkor az állomás színével megegyező színű vagonról leszállnak az utasok, így a vagon elveszti a színét, azzal már nem kell foglalkoznia a játékosnak. Az utasok leszállításában azonban van egy kikötés: mindig csak a mozdonyhoz legközelebbi, nem üres kocsiról szállhatnak le az emberek az egyes állomásokon. Cél még, hogy a vonatok ne ütközzenek egymásnak, illetve, hogy a váltók mindig úgy legyenek beállítva, hogy a szerelvények továbbhaladása biztosítva legyen. Hiszen ha ezek nincsenek biztosítva, az vonatok felrobbanását eredményezi, így a játék azonnal véget ér.

Az eddigiekben láthattuk a játék struktúráját, most nézzük meg, hogy hogyan is zajlik a játékmenet:

Először létrejön egy pálya, annak minden korábban felsorolt elemével (kivéve az alagutat, mivel annak létrehozása a játékos feladata). Ezután – az aktuális beállítástól függően – bizonyos időközönként vonatok érkeznek a terepasztalra, rajtuk különböző színű kocsikkal, benne utasokkal. A szerelvények folyamatosan mozognak, a játékos nem tudja őket megállítani. A váltók állításával és az alagutak kezelésével a felhasználó irányítja a terepasztalon folyó történéseket. Ha egy vonat felrobban, akkor a játéknak vége. Ha a pálya teljesítésre kerül, azaz minden utas épségben megérkezik a neki megfelelő állomásra, akkor az adott pálya teljesítettnek számít, a felhasználó továbblép a következő szintre, ahol ismételten meg kell küzdenie az elé táruló akadályokkal.

A játék előrehaladtával a felhasználónak egyre nagyobb kihívás sikeresen teljesíteni a pályákat, mivel a játék fokozatosan nehezedik. Ezt több eszköz is biztosítja:

- Egyre több vonat érkezik be a pálya területére, amiket ki kell üríteni.
- A szerelvények egyre több vagonból állnak, így több állomásra kell őket eljuttatni.
- A vonatok egyre gyorsabban mozognak a síneken, ebből kifolyólag nagyobb figyelmet kell fordítani az ütközések elkerülésére.

A játék egészét nézve a felhasználó célja az, hogy minél több pályát tudjon sikeresen teljesíteni.

2.2.3 Felhasználók

A játékosnak rendelkeznie kell alapvető logikával és fontos a gyors döntéshozatal. Fejlett logisztikai képességre is szükség van a játék céljának eléréséhez. A játékosnak nem kell mélyebb ismeretekkel rendelkezni a játék témájával kapcsolatban az élvezetes felhasználáshoz. Minden korosztálynak ajánlott a program használata.

2.2.4 Korlátozások

A standard Java függvénykönyvtár, a Szoftvertechnikák című tárgyban tanultak (többnyire RUP) és a Szoftver projekt laboratórium tárgyban megadott dokumentum sablonok felhasználhatók a fejlesztés folyamán. A programkódot és a pontos dokumentációt a turbosnakes csapatnak kell elkészítenie.

2.2.5 Feltételezések, kapcsolatok

https://www.iit.bme.hu/targyak/BMEVIIIAB02/feladat címen elérhető feladatkiírás.

2.3 Követelmények

2.3.1 Funkcionális követelmények

Az.	Leírás	Ellenőrzés	Prioritás	Forrás	Use-case	Komment
1	Pálya felépítése	Bemutatás	Alapvető	Feladat kiírás	Create	A játék elindításakor a pálya betöltődik, megjelenik a képernyőn.
2	Játékos tudjon váltókat állítani	Bemutatás	Alapvető	Feladat kiírás	Switch_direction	
3	Alagút építése	Bemutatás	Alapvető	Feladat kiírás	Set_tunnel	A játékos a pályán előre kitüntetett helyekre építhet egy alagutat. Egyszerre csak egy alagút lehet felépülve az egész pályán
4	Pálya mentése	Bemutatás	Fontos	Csapat	Save	A felhasználó egy fájlba mentheti a játék aktuális állását.
5	Pálya betöltése	Bemutatás	Fontos	Csapat	Load	A lementett játékállást vissza lehet tölteni.
6	A kocsi színével megegyező állomáson szállnak le az utasok	Bemutatás	Alapvető	Feladat kiírás	Empty_carriage	
7	Az utasok az első nem üres, megfelelő színű kocsiról szállnak le	Bemutatás	Alapvető	Feladat kiírás	Empty_carriage	
8	A szerelvény eltűnik a pályáról, ha az összes kocsija kiürül	Bemutatás	Fontos	Csapat	Delete_carriage	Különben nem tűnne el soha.
9	Ha a szerelvény alagútba megy, akkor az alagútban lévő része nem látszik	Bemutatás	Alapvető	Csapat	Check_tunnel	
10	Ha a szerelvény	Bemutatás	Fontos	feladatk	Check_tun	

	alagútban van, akkor az alagutat nem lehet megszüntetni			iírás	nel	
11	A szerelvény tud haladni	Bemutatás	Alapvető	Feladat kiírás	Step	
12	Ha a szerelvények összeütköznek, a játék véget ér	Bemutatás	Alapvető	Feladat kiírás	Check_crash	
13	A szerelvény az alagútban ugyanolyan módon halad, mint a síneken	Bemutatás	Alapvető	Csapat	Step	
14	Az alagutakat csak a kijelölt pontokra lehet lehelyezni				Set_tunnel	
15	A váltót annyi féle irányba lehet állítani, ahány lehetséges kimenet van.	Bemutatás	Fontos	Csapat	Switch_direction	
16	A játéknak vége, ha olyan váltóhoz érkezik, ami nem biztosít továbbhaladást.	Bemutatás	Fontos	Csapat	Step	
17	A vonatok alagutakban is ütközhetnek.	Bemutatás	Fontos	Csapat	Chech_crash	
18	A vonatok mindig az alagutat választják az egyéb lehetséges útvonalakkal szemben.	Bemutatás	Fontos	Csapat	Step	
19	A vonat abba az irányba halad a alagút elhagyása után, mint amilyen irányba haladt a bemenetelkor.	Bemutatás	Fontos	Csapat	Step	

2.3.2 Erőforrásokkal kapcsolatos követelmények

Azonosító	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
1	Microsoft Word / Google Docs	Nincs	Alapvető	Csapat	Szövegszerkesztés
2	StarUML	Nincs	Alapvető	Csapat	UML szerkesztés
3	IntelliJ IDEA	Nincs	Alapvető	Csapat	Szoftverfejlesztő környezet
4	Git/GitHub	Nincs	Alapvető	Csapat	Kollaboráció, verziókezelés
5	Slack	Nincs	Alapvető	Csapat	Csapatkoordináció, kommunikáció
6	Eclipse	Nincs	Alapvető	Csapat	Szoftverfejlesztő környezet
7	Gimp/Paint	Nincs	Opcionális	Csapat	Képszerkesztő
8	JRE 1.8	Nincs	Alapvető	Csapat	Java Runtime Environment
9	JDK 1.8	Nincs	Alapvető	Csapat	Java Development Kit

2.3.3 Átadással kapcsolatos követelmények

Azon.	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
1.	2017. március 22. - szkeleton leadás	bemutatás	alapvető	ütemterv	
2.	2017. április 19. - protó leadás	bemutatás	alapvető	ütemterv	
3.	2017. május 10. - grafikus leadás	bemutatás	alapvető	ütemterv	
4.	Az anyagot egyetlen zip fájlba kell csomagolni.	bemutatás	alapvető	tárgyhon lap	
5.	A programnak a laboratóriumban rendszeresített JDK alatt lefordíthatónak és futtathatónak kell lenni.	bemutatás	alapvető	tárgyhon lap	

2.3.4 Egyéb nem funkcionális követelmények

Azon.	Leírás	Ellenőrzés	Prioritás	Forrás	Komment
1	A játékos az egér használatával játszik.	Bemutatás	Alapvető	Csapat	
2	A játék kezelése intuitív az átlagfelhasználó számára.	Bemutatás	Alapvető	Csapat	Pl.: nincsenek a user számára ismeretlen, de elengedhetetlen billentyűkombinációk.
3	A program megbízhatóan működik a biztosított laborgépen.	Bemutatás	Alapvető	Csapat	
4	A játék hibák nélkül lefut.	Bemutatás	Alapvető	Csapat	Pl.: nem lép ki menet közben.
5	Az alkalmazás nem fut túl lassan.	Bemutatás	Fontos	Csapat	Egy-egy parancs végrehajtása (szinte) azonnal megtörténik.
6	A játék grafikus felülete könnyen értelmezhető.	Bemutatás	Fontos	Csapat	A grafikai elemek jól elkülönülnek, jól beazonosíthatóak.

2.4 Lényeges use-case-ek

2.4.1 Use-case leírások

Use-case neve	Create
Rövid leírás	A pálya legenerálása.
Aktorok	Timer
Forgatókönyv	A pálya felépítése.

Use-case neve	Step
Rövid leírás	A vonat haladása a pályán.
Aktorok	Timer
Forgatókönyv	A vonat egyenletes sebességgel halad a pályán, nem tud megállni. Ha egy alagúthoz érkezik, akkor mindig az alagutat választja más útvonallal szemben, és az alagút belsejében is ugyanolyan sebességgel, ugyanabban az irányban halad, mint az alagúton kívül.

Use-case neve	Check crash
Rövid leírás	Ütközés detektálása.
Aktorok	Timer
Forgatókönyv	A játéknak vége, ha a pálya bármely részén (akár alagútban is), bármely két vonat ütközik egymással.

Use-case neve	Check_tunnel
Rövid leírás	Az alagút vizsgálata.
Aktorok	Timer
Forgatókönyv	Ha egy vonat az alagútban van, akkor az alagútban lévő része nem látható és az alagutat nem lehet bezárni.

Use-case neve	Check_color_at_station
Rövid leírás	A szerelvények színének ellenőrzése.
Aktorok	Timer
Forgatókönyv	Az állomásoknál csak az állomás színével megegyező, legelöl lévő, nem üres kocsiról szállnak le az utasok.

Use-case neve	Empty_carriage
Rövid leírás	A kocsik kiürítése.
Aktorok	Timer
Forgatókönyv	Ha minden feltétel teljesül (első nem üres, állomás színével egyező kocsi), akkor a kocsi kiürül és elveszti a színét.

Use-case neve	Delete_carriage
Rövid leírás	A vonat törlése a pályáról.
Aktorok	Timer
Forgatókönyv	Ha a vonat összes kocsija kiürül, akkor eltűnik a pályáról.

Use-case neve	MouseClick
Rövid leírás	A játékos kattintását szemlélteti.
Aktorok	Player
Forgatókönyv	A játékos a játékot kattintással vezérelheti. Így tud új játékot kezdeni vagy egy régebbit betölteni, így állíthatja a pályán lévő elemeket (váltó állítása, alagút létrehozása/lerombolása), így tudja elmenteni az eddigi játékait és így tud kilépni a programból.

Use-case neve	Start
Rövid leírás	A játék indítása.
Aktorok	Player
Forgatókönyv	A játékos eldöntheti indításnál, hogy új játékot kezdjen vagy egy korábbit töltsön be.

Use-case neve	New_game
Rövid leírás	Új játék kezdése.
Aktorok	Player
Forgatókönyv	Egy új játék indul.

Use-case neve	Load
Rövid leírás	Elmentett pálya visszatöltése.
Aktorok	Player
Forgatókönyv	Betöltődik a korábban elmentett pálya.

Use-case neve	Save
Rövid leírás	Pálya mentése
Aktorok	Player
Forgatókönyv	A pálya kiíródik egy fájlba.

Use-case neve	Switch_direction
Rövid leírás	Váltó állítása
Aktorok	Player
Forgatókönyv	A váltó átállítódik kattintásra.

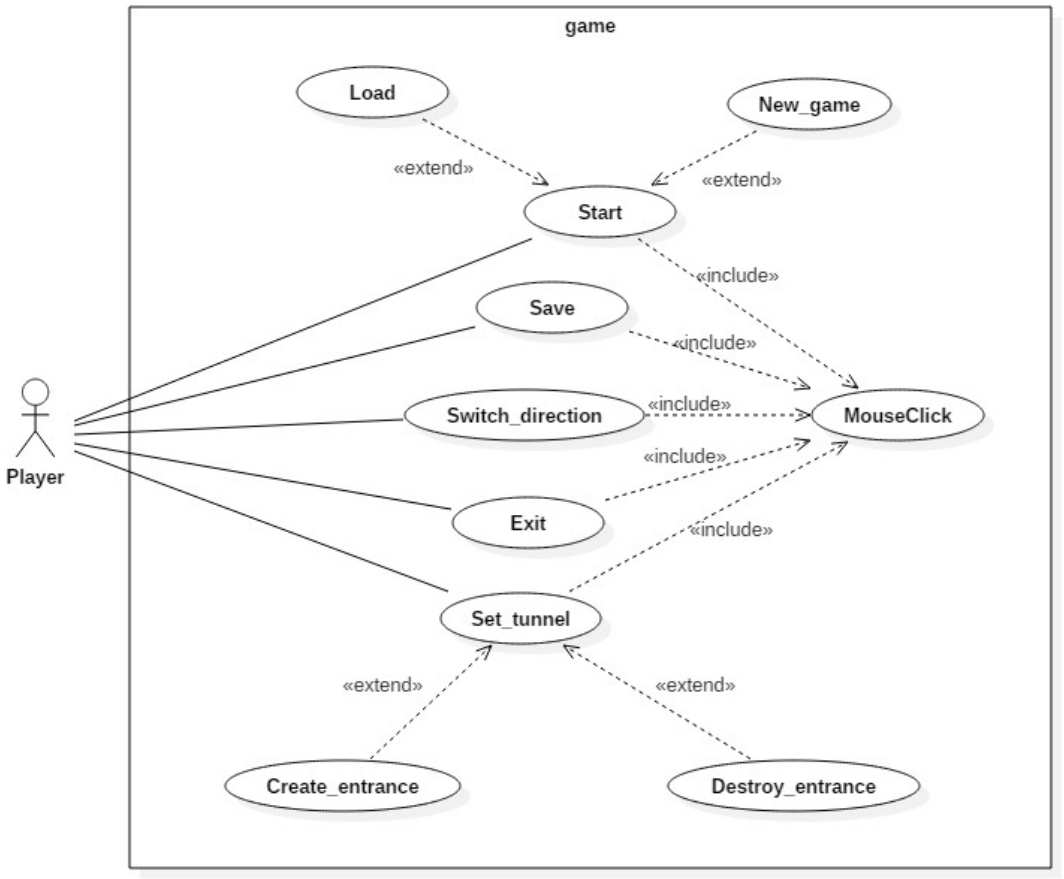
Use-case neve	Exit
Rövid leírás	A játék kilép.
Aktorok	Player
Forgatókönyv	A játékos bármikor képes megszakítani a játékot és kilépni a programból egy kattintással.

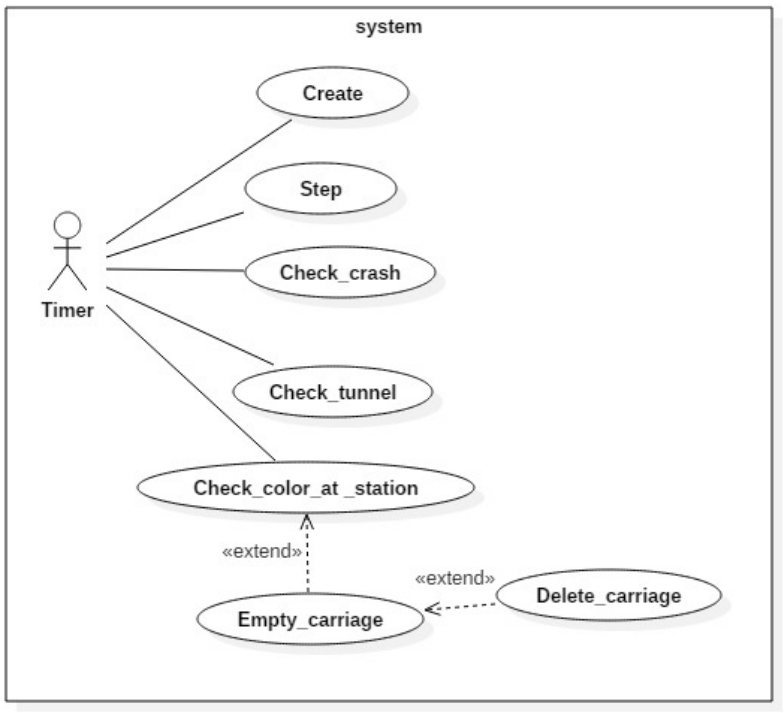
Use-case neve	Set_tunnel
Rövid leírás	Alagút építése/rombolása
Aktorok	Player
Forgatókönyv	A játékos képes a pályán kattintással, az arra megfelelő helyen alagutat létrehozni, vagy a már létrehozott alagutat lerombolni.

Use-case neve	Create_entrance
Rövid leírás	Alagút bejáratának építése.
Aktorok	Player
Forgatókönyv	A játékos kattintással létrehoz egy alagút bejáratot, akkor, ha a kattintás olyan elemén történik a pályának, ahol létre hozható alagút.

Use-case neve	Destroy_entrance
Rövid leírás	Egy már felépített alagút lerombolása.
Aktorok	Player
Forgatókönyv	A játékos képes kattintással egy felépített alagutat lerombolni.

2.4.2 Use-case diagram





2.5 Szótár

Kifejezés	**Jelentés**
Mozdony	A kocsikat húzó szerelvény, amely kizárólag a síneken közlekedhet
Kocsi	A mozdony után lévő szerelvények, különböző színnel ellátva, úgyszintén kizárólag a síneken közlekedhetnek
Vonat	Kizárólag egy mozdonyból és több kocsiból áll. Kocsik csak a mozdony után helyezkedhetnek el.
Szerelvény	Ebben a környezetben ugyan az mint a vonat.
Sín	Egy útvonal, amelyet a vonat követ. Vagyis ezen közlekedik.
Alagút	Olyan elem, amely alapvetően nem található a pályán, viszont a játékosnak lehetősége van ilyen elemeket elhelyezni az arra kijelölt helyeken, hogy segítse a vonatok zavartalan közlekedését.
Kijelölt hely	Olyan pontok a pályán, ahova a játékos alagutakat helyezhet el, ezek alapértelmezetten elemei a pályának, a pályán való elhelyezkedésük előre meghatározott.
Váltó	A sínekhez tartozó elem. Bizonyos pontokban a sínek több fele ágazhatnak, az ilyen pontokban a váltó határozza meg, hogy melyik irányba haladjon tovább a vonat. A játékosnak lehetősége van ezeknek a váltóknak a kezelésére, átváltására.
Megálló	A pálya azon elemei, amik kizárólag sínek mentén helyezkedhetnek el. Itt állnak meg a vonatok. A megállóknak is különböző színeik lehetnek.
Pálya	Sínekből, kocsikból, mozdonyokból, megállókból, váltókból, utasokból álló elem, amely alagutakat is tartalmazhat.
Szín	A megállókhoz illetve kocsikhoz rendelt megkülönböztető jelzés.
Időzítő	Szabályozza, hogy mennyi időnként indulnak a vonatok a pálya széléről.
Utas	Alapvetően a vonaton tartózkodnak, a megfelelő megállóban a megfelelő kocsiból leszállnak.
Összeütközés	Két vonat összeütközhet, ha nem megfelelően vannak irányítva, ilyenkor mindkét vonat felrobban.
Terepasztal	A pálya alapját képező elem.

2.6 Projekt terv

2.6.1 Feladatok és határidejük

- Követelmény, projekt, funkcionalitás - február 20.
- Analízis modell kidolgozása 1. - február 27.
- Analízis modell kidolgozása 2. - március 6.
- Szkeleton tervezése - március 13.
- Szkeleton elkészítése - március 20.
- Prototípus koncepciója - március 27.
- Részletes tervek - április 3.
- Prototípus készítése, tesztelése - április 10.
- Prototípus beadása - április 17.
- Grafikus felület specifikációja - április 24.
- Grafikus változat elkészítése - május 1.
- Grafikus változat beadása - május 8.
- Átadás - május 10.

2.6.2 Beosztás és felelősségek

A program elkészítésének részfolyamataiban minden csapattag részt vesz, hiszen fontos számunkra, hogy a feladatok súlya egyenletes eloszlást képviseljen az egyes csapattagok között, valamint a felmerülő problémákat ezáltal könnyebb elhárítani az értekezleteken több szem többet lát alapon. A fentiekben leírtak teljeskörű betartása mellett fontosnak tartjuk, egyéb kitüntettett felelősségek kiosztását:

Vízi Előd	Játék kinézetének (grafikus felület), pályák felépítésének tervezése és ezek kódolása, dokumentációk írása, diagramok készítése.
Salamon Krisztián	Dokumentáció formai egyeztetése, valamint pull requestek felülvizsgálata. Feladatom továbbá a pályák tervezése, valamint az azokon található grafikai elemek megrajzolása.
Fenes Áron	Tesztelés a projekt során, dokumentáció tartalmi felülvizsgálása, játék kinézetének tervezése, kódolása a projektnek.
Dobó Ádám	Dokumentáció írása, pálya/pályák megtervezése, kódolás, diagramok készítése, valamint annak vizsgálata, hogy a termék egy előállított iterációja megfelel-e az addigi követelményeknek.
Papp Attila	Projekt vezető, gyűlések szervezése és csapattagok rugdosása (előrehaladás

	tekintetében). Feladatom még a kódolás és a kommunikációs csatornák összeegyeztetése, valamint a github repoban a develop ágból a master ágba történő pull requestek felügyelése, hogy azok a megbeszélt konvenció szerint történjenek.

2.6.3 Választott eszközök

Csapatunk a **GitHub**ot fogja használni verziókezelő rendszernek, melyen belül egy privát repóhoz van mindenkinek hozzáférése. A GitHubot mindenki saját megítélése szerint fogja elérni egy asztali alkalmazáson vagy fejlesztőeszközön esetlegesen a webes felületen keresztül.
A programkódok megírása az **Intellij IDEA 2016.3.3**-as verziójával fog történni az alapértelmezett pluginkészlet felhasználásával.

UML diagramok elkészítésére a StarUML-t alkalmazzuk.

A webes kommunikációnkat a **Slack** könnyíti meg, melyen belül külön csatornáink vannak általános, projekt hirdetmény illetve offtopic beszélgetések lebonyolítására. Két bot segíti munkánkat, az egyik a BusyBot, melynek segítségével könnyebb az egyes kiosztott feladatokat menedzselni, illetve a másik egy GitHub bot, amely megmutatja a repónkba érkező változásokat.

Természetesen a dokumentumok elkészítésénél is nagy hangsúlyt fektetünk a közös terhelésre, így a beadandókat is közösen szerkesztjük a **Google Docs** segítségével.

2.7 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017. 01. 31. 14:00	2 óra	Fenes Dobó Salamon Vízi Papp	Github repó és Slack beállítások létrehozása és tesztelése.
2017. 02. 13. 19:00	3 óra	Fenes Dobó Salamon Vízi Papp	Értekezlet, feladat vázlatos megértése és esetleges kihívások megbecslése. Szerepek kiosztása a dokumentáció kitöltésében. Döntés: ● pálya elemei: vonat, kocsi, sín, alagút, állomás, váltó ● a játékot le lehet majd menteni ● Kiosztott szerepek: ○ Papp: ■ 2.1.1 ■ 2.1.5 ■ 2.3.1 ■ 2.3.4 ■ 2.4.1 ■ 2.6

			○ Salamon: ■ 2.2.2 ■ 2.3.1 ■ 2.3.2 ■ 2.3.4 ○ Fenes: ■ 2.1.2 ■ 2.2.3 ■ 2.2.5 ■ 2.3.1 ■ 2.4.2 ○ Dobó: ■ 2.1.3 ■ 2.1.4 ■ 2.2.1 ■ 2.3.3 ■ 2.4.1 ○ Vizi ■ 2.2.1 ■ 2.2.4 ■ 2.3.1 ■ 2.3.2 ■ 2.5 ● dokumentáció vázlatos kitöltését mindenki befejezi a konzultációig ● a végleges dokumentációt elegendő vasárnap délutánig elkészíteni
2017. 02. 13. 23:15	1,5 óra	Papp	Ezen dokumentum 2.1.1, 2.1.5 és 2.3.1 alpontjainak elkészítése
2017. 02. 14. 9:00	1 óra	Salamon	Funkciók (2.2.2) leírása, az azzal kapcsolatos kérdések megfogalmazása.
2017. 02. 14. 16:00	1 óra	Fenes	Szakterület (2.1.2), felhasználók (2.2.3), feltételezések, kapcsolatok (2.2.5) szövegének átgondolása, megfogalmazása, dokumentumba írása.
2017. 02. 14. 20:00	2 óra	Fenes Dobó Salamon Vízi Papp	Értekezlet a kódolási stílusunk megalkotására vonatkozóan, illetve egyéb a verziókezeléssel kapcsolatos konvenciók felállítása. A funkciók leírása során felmerült kérdések megtárgyalása. Döntés: ● IntelliJ-t használunk ● az általános Java konvenciókat betartjuk ● kódolás közben dokumentálunk, Javadoc szabványt betartva ● bevezettük a hegy, mint akadály ötletét ● pontozás ötletének elvetése
2017.02.14. 16:00	1,5 óra	Dobó	Definiciók, rövidítések (2.1.3) megírása, use-case leírások (2.4.1) készítése, átadással kapcsolatos követelmények (2.3.3) megfogalmazása, 2.1.4

			megírása.
2017.02.14. 19:00	1,5 óra	Fenes	Use-case diagram megtervezése (2.4.2), StarUML programban kivitelezése, dokumentációba beillesztése.
2017.02.14. 20:15	2 óra	Papp	Ezen dokumentum 2.3.4, 2.4.1 és 2.6 pontjainak megírása
2017. 02. 14. 21:00	1 óra	Salamon	A funkciók leírásának (2.2.2) befejezése, finomítása. Néhány újonnan felmerülő funkcionális követelmény (2.3.1) beírása.
2017. 02. 14. 19:00	2 óra	Vízi	Szótár (2.5) nagy részének megírása, korlátozások (2.2.4) kifejtése, erőforrásokkal kapcsolatos követelemények egy részének megírása (2.3.2).
2017. 02. 15. 13:00	0,5 óra	Fenes Dobó Salamon Vízi Papp	Értekezlet. A konzultáción felmerülő kérdések rövid átfutása, elképzelések cseréje. Döntés: ● hegy ötletének elvetése (konzulens javaslatára) ● alagút: kijelölt helyek? utána merre menjen tovább? váltó az alagút előtt, után? (következő értekezlet előkészítése)
2017. 02. 15. 19:30	2 óra	Fenes Dobó Salamon Vízi Papp	Értekezlet. A 13:00-kor megkezdett, de hamar befejezett értekezlet folytatása. Döntés: ● alagutak kijelölt helyeken (lásd Funkciók rész) ● az alagút két bejárata egyszerre helyezhető el ● alagútból kilépés után a haladási irány nem változik (nem kell váltó)
2017. 02. 15 21:00.	1 óra	Salamon	Funkciók (2.2.2) átírása a konzultáció és az azutáni értekezletek alapján. Néhány erőforrásokkal kapcsolatos követelmény megfogalmazása (2.3.2)
2017.02.15 21:00	2 óra	Dobó	Általános áttekintés diagramjának elkészítése (2.2.1), definíciók kiegészítése (2.1.3).
2017. 02. 16. 13:30	2,5 óra	Vízi	Általános áttekintésnél található diagram elemeinek magyarázata (2.2.1), a teljes dokumentáció átolvasása, elírások javítása, konzisztencia felülvizsgálata, majd ezek alapján a szótár (2.5) illetve a definíciók (2.1.3) bővítése. Néhány hiányzó funkcionális követelmény leírása (2.3.1).
2017. 02. 16. 20:00	2 óra	Fenes	Use-case diagram (2.4.2) kibővítése, átírása a megbeszélés alapján, funkcionális követelmények (2.3.1) use-casehez rendelése.
2017. 02. 16. 18:30	1 óra	Papp	Dokumentum formázásának átnézése, tördelések rendbeszedése.
2017. 02. 18. 18:50	1,5 óra	Salamon	Nem funkcionális követelmények (2.3.4) megírása, Funkciók (2.2.2) kiegészítése, napló (2.7) javítása.
2017.02.19. 9:45	0,5 óra	Dobó	Use-case leírások (2.4.1) bővítése a kibővített use-case diagram alapján.