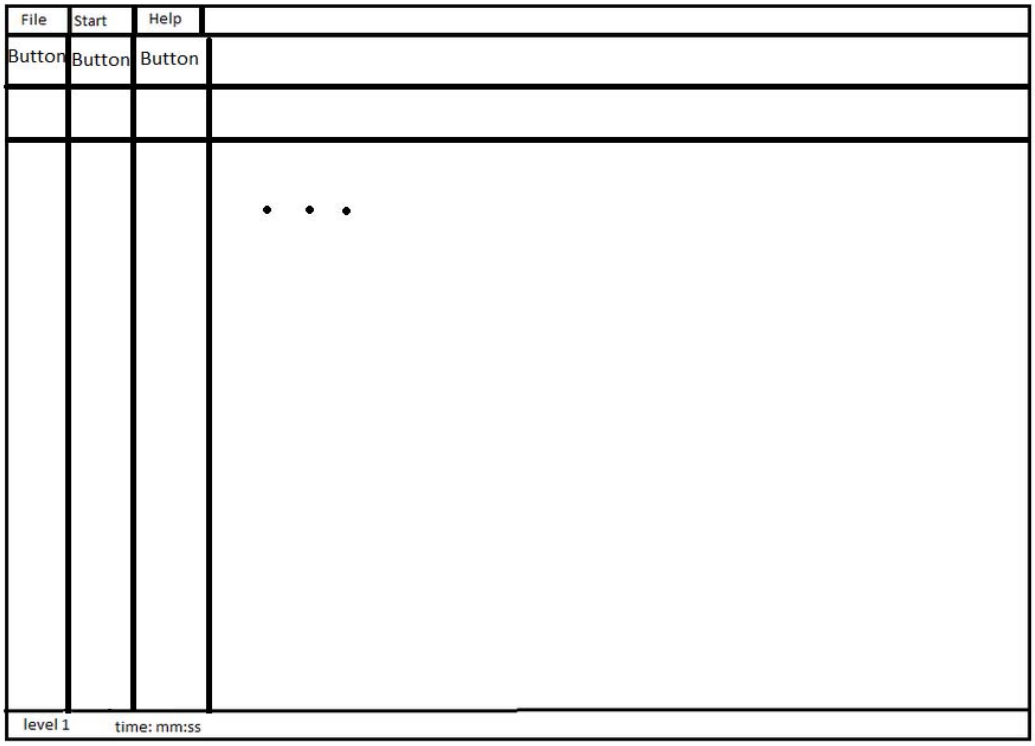
11.Grafikus felület specifikációja

11.1 A grafikus interfész

A szoftver elindítása után egy olyan felület jön elő amit az alábbi rajzos ábra szemléltet. Legelőször a fent látható menü jelenik meg a különböző funkciókkal.



- File menüben lehet majd pályákat betölteni, valamint kilépni a játékból
- Start-al indítható a játék, illetve itt lehet újrakezdeni.
- Help menüpontban lehet a játék logikájáról valamint annak szabályairól olvasni.

A pályát úgy építjük fel, hogy egy mátrix szerű szerkezet minden indexű eleme egy Button lesz. Ezekre fogjuk a képeket beállítani ami az adott elemet jelenti majd a játék logikájához.
A gombok nem lesznek kattinthatóak, ha arra nincs szükség a játékvezérlés szempontjától. Ilyen elemek pl.: sínek, állomás. A váltó onnan lesz megismerhető a játékos számára, hogy a váltó képét tartalmazó button kerete más színű lesz. A váltó mindig olyan sínt ábrázoló képet fog megjeleníteni ami megegyezik a beállított állásnak megfelelővel. Az aktív alagút szemléltetése is hasonló módon történik a váltóéhoz. Azt, hogy melyik alagút aktív az szemléltetni, hogy az alagút egész hosszában (összes tunnelrail és tunnelentrence tartalmazó button) zöld kerettel rendelkezik. A többi, értelem szerűen inaktív alagút keretének színe piros lesz. Az alagút aktiválható, ha bármelyik részére (tunnelentranceok és runnelrailek) kattint a játékos. A játékpálya alatt lesz egy sáv, ahol a felhasználó láthatja az aktuális szint nevét és lesz egy óra, ami méri, hogy azon a szinten mennyi ideje játszik a felhasználó.

Ahogy az fentebb is említésre került a programunkban található MapElementeket a nekik megfelelő képekkel fogjuk ábrázolni a grafikus felületen. Ezeket a képeket az alábbi felsorolásban szeretnénk bemutatni:

- Sín
  - egyenes

  - kanyarodó

- Állomás
  - üres

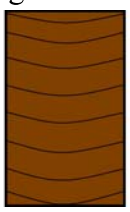
  - nem üres

- Vonat
  - mozdony

  - kocsi
    - ❑ utas szállító (kék, zöld, sárga, piros)

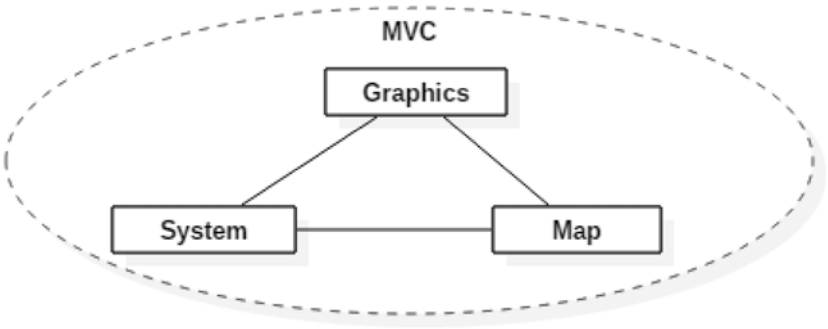
- ❑ szeneskocsi

- Alagút



Megjegyzés: A fenti felsorolásba az állomásokból illetve az utasszállító kocsiból csak a piros színűt raktuk be szemléltetésképpen, de a játékunkban ezenkívül még van zöld, sárga és kék színű kocsi illetve állomás is. Továbbá a szeneskocsi csak annyiban különbözik a többi kocsitól, hogy tiszta fekete a színe, és fekete színű állomás nincsen.

11.2 A grafikus rendszer architektúrája



11.2.1 A felület működési elve

Már a tervezés korai fázisaiban (Analízis modell elkészítése) is igyekeztünk az MVC alapelveit szem előtt tartva megalkotni a szoftverünk struktúráját, így nagy átszervezést nem igényel az egyes MVC komponensek bevezetése illetve azonosítása.

- System osztályunk fogja a Controller szerepét felvenni
- Map osztályunk (amit a System tartalmaz) szolgál Modelként
- A View szerepét a Graphics látja el.

Alapvetően Push alapú MVC-ben gondolkodtunk, amiben a modell illetve a (controller) értesíti a felületet, hogy változott. Természetesen a felületnek is lehet üzenete a Contoller irányába (pl. kattintás), ez a már elkészített függvények meghívását fogja kiváltani.

Vizsgáljuk meg konkrét példán keresztül:

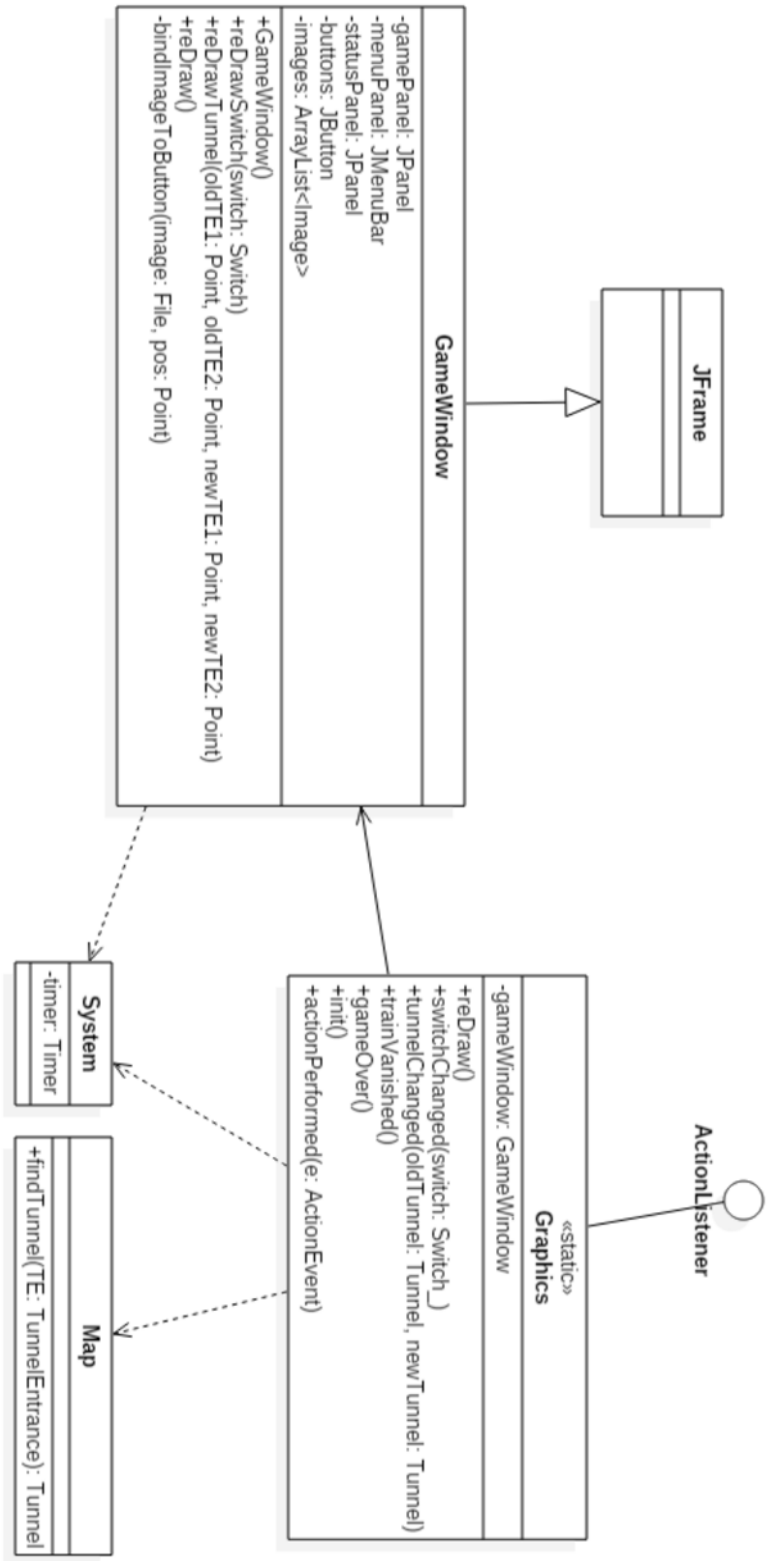
- Tfh, a játék futása közben úgy alakult, hogy két vonat összeütközött. Ekkor (a logikának köszönhetően) meghívódik a System egy ezt kezelő függvénye, ami értesíteni fogja a View-t, ezáltal frissítve a felületet.
- Tfh, a játékos egy switchre kattintott a felületen. Ekkor alapvetően a View irányából szeretnénk információt vinni a modellhez. Rendszerünk erre is fel van készítve, hiszen a System osztály képes parancsokat értelmezni és a megfelelő parancs meghívásán keresztül tudja állítani a modellben a switchet (természetesen a játék szabályait figyelemben tartva)

Látható, hogy az információáramlás mindkét irányban zökkenőmentesen biztosított, hiszen az előbb említett példák alapvetően generikusak, azaz egy switch állítás ugyan azt a mechanizmust váltja ki, mint egy tunnel építés, csak éppen más függvények hívódnak.

A megjelenítésért felelős osztályok a következőképpen épülnek be: System kommunikál a Graphics megfelelő függvényeivel ezáltal elérve az üzenet alapú kommunikációt. A Graphics actionPerformed metódusa pedig üzenetet intéz a Systemhez.

Természetesen szem előtt tartottuk az erőforrásokkal való hatékony bánásmódot, így a switch állítása vagy tunnel építése esetén nem rajzolódik újra az egész pálya, csak a megfelelő elemek.

11.2.2 A felület osztály-struktúrája



11.3 A grafikus objektumok felsorolása

11.3.1 GameWindow

- **Felelősség**

Feladat a grafikus felület összefogása, valamint a komponensei által fogadott és küldött üzenetek kezelése.

- **Ősosztályok**

JFrame

- **Interfészek**

Nincsenek.

- **Attribútumok**
  - **- JPanel gamePanel:** a játékhoz tartozó vezérlőket összefogó panel.
  - **- JPanel menuPanel:** a menüt tartalmazó panel
  - **- JPanel statusPanel:** a státuszsávot összefogó panel
  - **- JButton buttons:** az egyes pályaelemek illetve vonatok egy gomb formájában vannak elhelyezve a pályán
  - **- ArrayList<Image> images:** a játékbeli alakzatok kép reprezentációja

- **Metódusok**
  - **+ GameWindow():** A játék ablakának konstruktora. Itt példányosítjuk az egyes elemeket valamint itt történik azok elrendezésének beállítása.
  - **+ void reDrawSwitch(Switch: switch):** egy switch állásának újrarajzolása, ha változás történt.
  - **+ void reDrawTunnel(oldTE1: Point, oldTE2: Point, newTE1: Point, newTE2: Point):** egy tunnel felépítése (gombra festése). Meg kell adni a régi tunnelt, hogy vissza tudjuk állítani az eredeti kinézetét, valamint az újat.
  - **+ void reDraw():** az egész pálya újrarajzolása
  - **- void bindImageToButton(image: File, pos: Point):** kép hozzárendelése a gombhoz.

11.3.2 System

- **Felelősség**

A játék központi logikáját valósítja meg. Most el lett látva egy timer-el, hogy lehessen ütemet adni a játéknak.

- **Ősosztályok**

Nincsenek.

- **Interfészek**

Nincsenek.

- **Attribútumok**
  - **- Timer timer:** egy időzítő, ami a játék ütemét adja
- **Metódusok**
  - **nem történt változtatás**

11.3.3 Map

- **Felelősség**

A játékbeli elemeket összefogó osztály. Kibővült egy új függvénnyel.

- **Ősosztályok**

Nincsenek.

- **Interfészek**

Nincsenek.

- **Attribútumok**
  - **nem módosult**
- **Metódusok**
  - **+ Tunnel findTunnel(TunnelEntrance TE):** megkeresi az adott TE-hez tartozó Tunnelt és visszatér vele.

11.3.4 Graphics

- **Felelősség**

A játék grafikáját kezelő osztály.

- **Ősosztályok**

Nincsenek.

- **Interfészek**

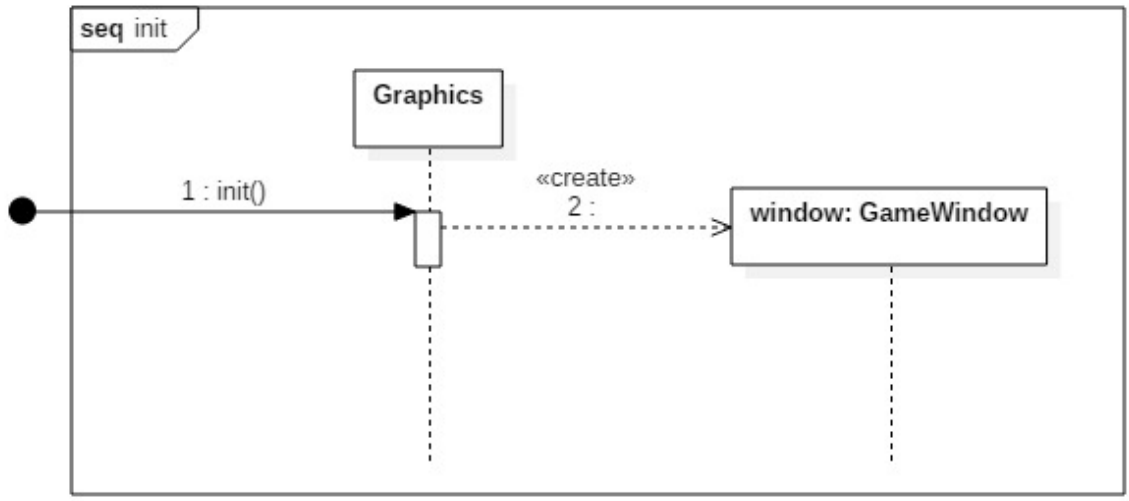
ActionListener

- **Attribútumok**
  - **- GameWindow gameWindow:** az előbbiekben tárgyalt játékablak.
- **Metódusok**
  - **+ void reDraw():** pálya újrarajzolása
  - **+ void switchChanged(Switch_ switch):** ha megváltozott egy switch állása, akkor a megfelelő grafikai művelet elvégzése
  - **+ void tunnelChanged(Tunnel oldTunnel, Tunnel newTunnel):** ha egy tunnel épült akkor a megfelelő grafikai függvény meghívása a kinézet frissítése gyanánt.
  - **+ void trainVanished():** ha egy vonat eltűnik, akkor a kinézetről is tűnjön el.

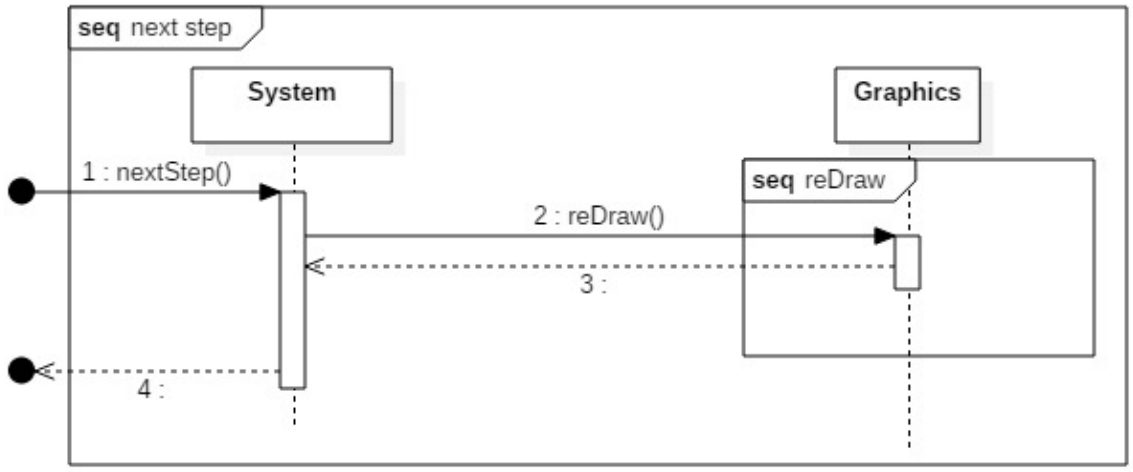
- **+ void gameOver():** értesítsük a felhasználót, ha a játéknak vége.
- **+ void init():** megfelelő konstruktorok meghívása és a kinézet felállítása
- **+actionPerformed(ActionEvent e):** esemény történt a GUI-n, annak továbbítása

11.4 Kapcsolat az alkalmazói rendszerrel

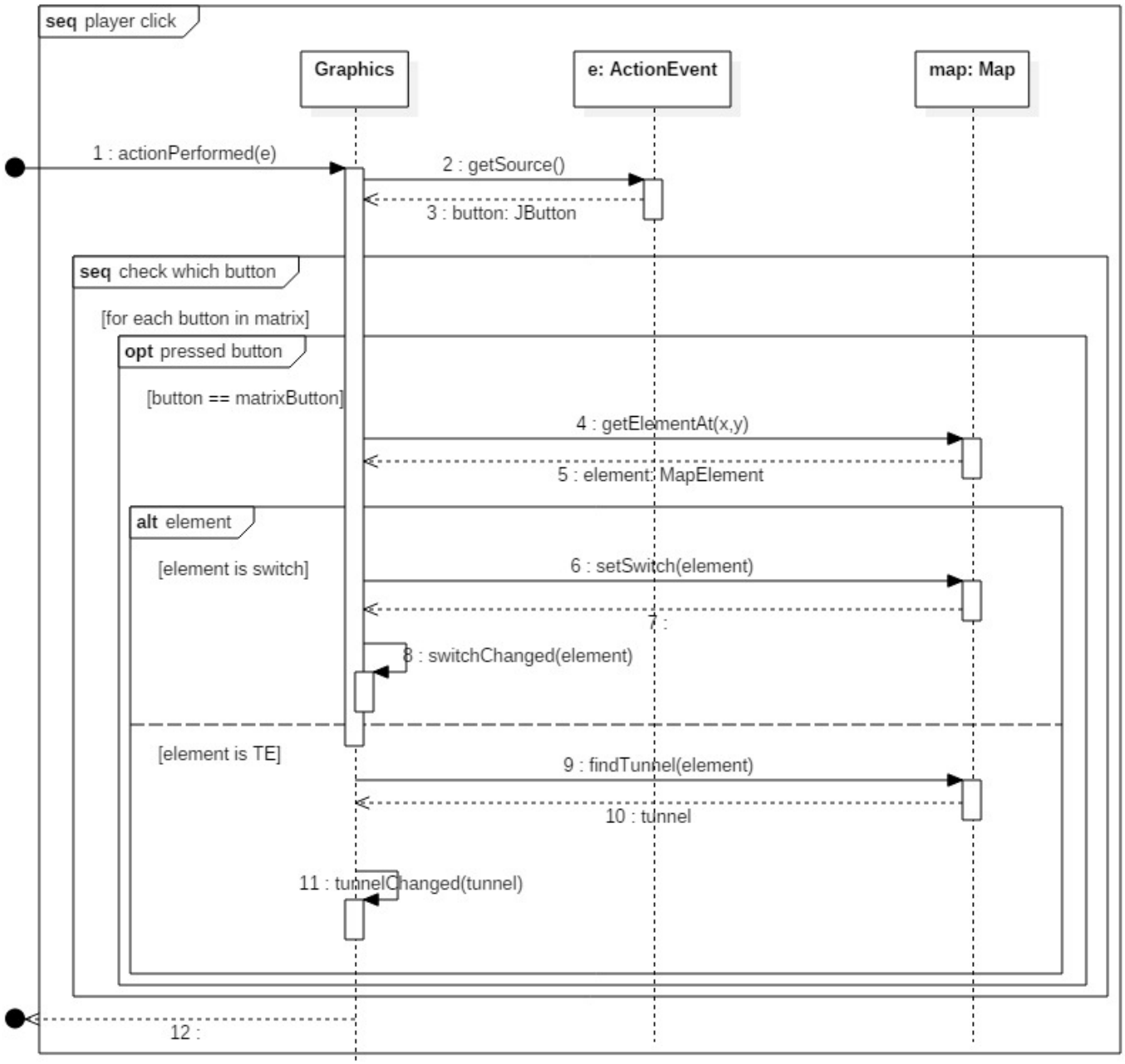
11.4.1 init



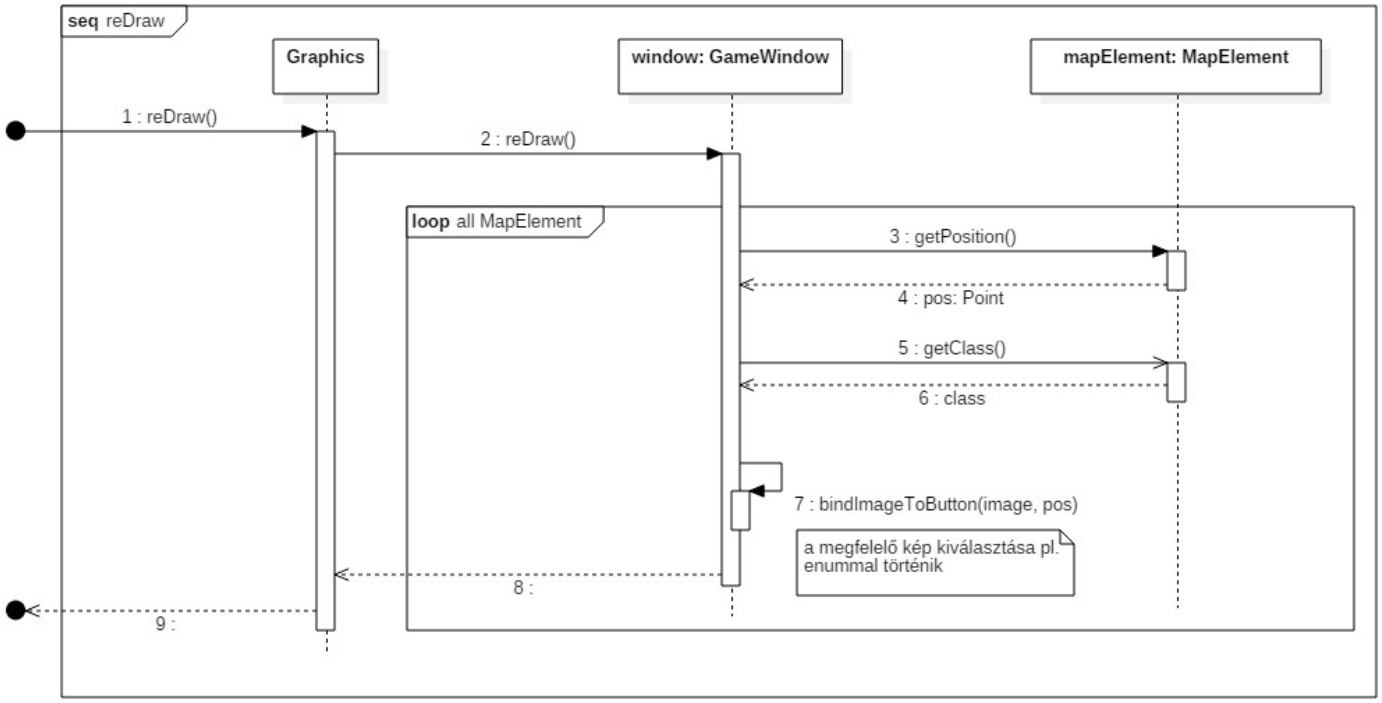
11.4.2 next step



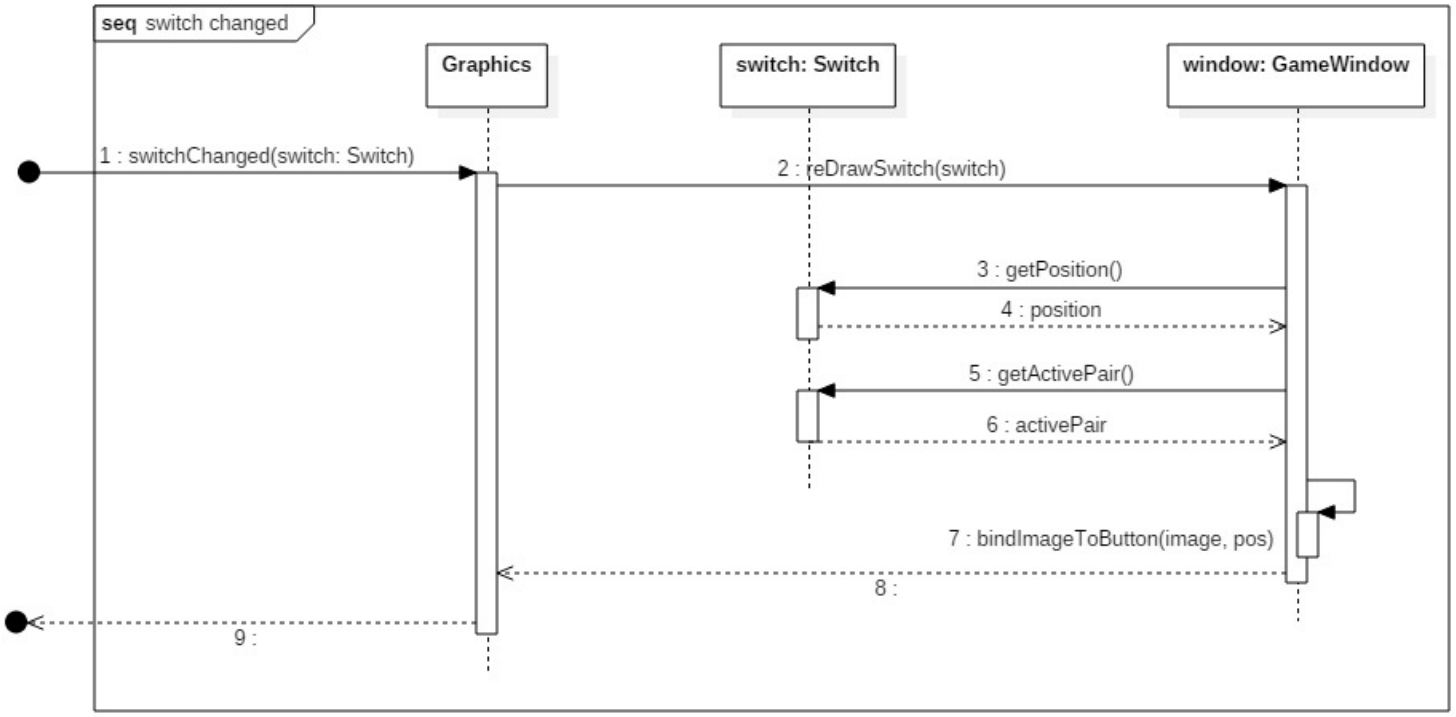
11.4.3 player click



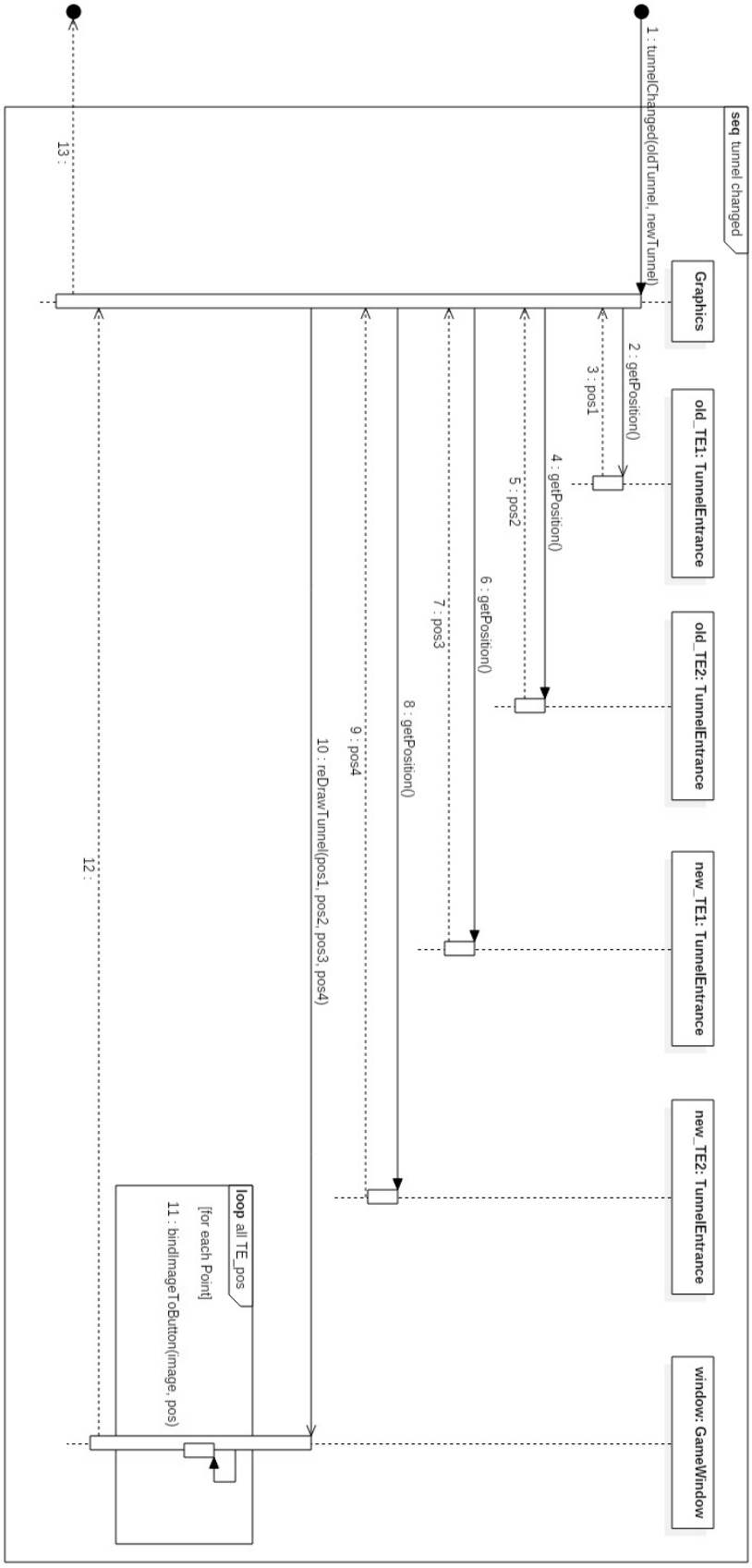
11.4.4 reDraw



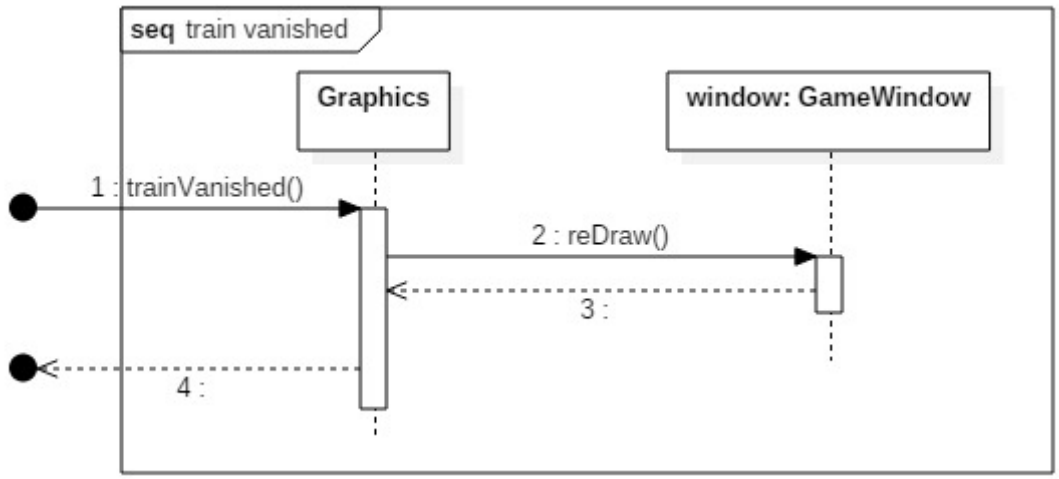
11.4.5 switch changed



11.4.6 tunnel changed



11.4.7 train vanished



11.5 Napló

Kezdet	Időtartam	Résztvevők	Leírás
2017. 04. 22. 10:00	2 óra	Dobó Fenes Papp Salamon Vizi	MVC alapelveinek értelmezése és a szoftverünkben történő implementálásának végiggondolása. Feladatok kiosztása: ● Dobó: ○ init szekvencia ○ next step szekvencia ○ reDraw szekvencia ○ tunnel changed szekvencia ● Fenes: ○ osztálydiagram elkészítése a megbeszéltek alapján ○ grafikus interfész megtervezése ○ grafikus elemek tervezése ● Papp: ○ a felület működési elvének kitalálása (MVC modell szem előtt tartása) ○ osztályleírások elkészítése ● Salamon: ○ player click szekvencia ○ switch changed szekvencia ○ train vanished szekvencia ● Vizi: ○ grafikus interfész megtervezése ○ grafikus elemek tervezése ○ grafikus elemek megvalósítása

			Határidő mindenkinek: 2017. 04. 22. 20:00
2017. 04. 22. 10:30	3 óra	Dobó	init, next step, reDraw, tunnel changed szekvencidiagramok elkészítése.
2017. 04. 22. 11:00	1 óra	Fenes Vizi	Grafikus interfész és a grafikus elemek megtervezése.
2017. 04. 22. 14:00	1 óra	Fenes	A grafikus interfész leírásának elkészítése.
2017. 04. 22. 14:00	3 óra	Papp	Ezen dokumetum ● 11.2.1 ● 11.3 pontjának elkészítése.
2017. 04. 22. 15:00	2 óra	Vizi	Grafikus elemek elkészítése képszerkesztő alkalmazás segítségével.
2017. 04. 22. 16:00	1 óra	Fenes	Osztálydiagram elkészítése.
2017. 04. 22. 16:30	3 óra	Salamon	A felvállalt szekvenciadiagramok elkészítése: player click, switch changed, train vanished.