# Grafikus felület specifikációja

## A grafikus interfész

[A menürendszer, a kezelői felület grafikus képe. A grafikus felület megjelenését, a használt ikonokat, stb screenshot-szerű képekkel kell bemutatni. Az építészetben ez a homlokzati terv.]

## A grafikus rendszer architektúrája

[A felület működésének elve, a grafikus rendszer architektúrája (struktúra diagramok). A struktúra diagramokon a prototípus azon és csak azon osztályainak is szerepelnie kell, amelyekhez a grafikus felületet létrehozó osztályok kapcsolódnak.]

A modell elemei nem tudnak semmit a view elemeiről, de a view osztályai képesek olvasni a hozzájuk tartozó modell osztály view-nak átadott interfészét. Minden view osztály implementálja az updateable interfészt, aminek van egy update tagfüggvénye. Ez újrarajzolja az adott view osztályt, a hozzácsatolt modell osztály adatai alapján. A modell és a view kapcsolatát a „ViewRepository” osztály tartja fenn. Ez az osztály tartalmaz magában egy statikus Hash táblát, amelynek a kulcsa egy objektum és az értéke a hozzá tartozó view osztály. Amikor egy view osztály frissítését szeretnénk kérni, akkor azt a modell osztály kezdeményezi. Ezt a „ViewRpository” updateObject tagfüggvényével teheti meg, ahol saját magát adja át, mint argumentum és utána a ViewRepositry meghívja a hozzá tartozó view update metódusát. A modellt viewhoz kötni, a ViewRepository bind eljárásával lehet megtenni. Azért, hogy a játék során létrejövő objektumok is össze legyenek kötve egy saját view-al, ezért a modellben léteznek „AbtactFactory” osztályok, amely létrehoz a modell osztálynak megfelelő view osztályt, és összeköti a két osztályt.

* szövegesen részletezni a grafikus elemek működését (mi a működés elve, pl. minden modell-elemhez van egy view, amelyik értesül a változásokról, ki kinek szól, a movell ad infót a viewnak, vagy a view kér infót a modelltól, stb). Cél az, hogy a modellhez minél kevésbé kelljen hozzányúlni. Ha holnapután VR sisakkal is akarjuk játszani a játékot, a modellt ne kelljen átírni. – Zsombor, én megnézem
* ha új pályaformátum kell a grafikus infók miatt – Zsombor, én megnézem

### A felület működési elve

[Le kell írni, hogy a grafikai megjelenésért felelős osztályok, objektumok hogyan kapcsolódnak a meglevő rendszerhez, a megjelenítés során mi volt az alapelv. Törekedni kell az MVC megvalósításra. Alapelvek lehetnek: **push** alapú: a modell értesíti a felületet, hogy változott; **pull** alapú: a felület kérdezi le a modellt, hogy változott-e; **kevert**: a kettő kombinációja.]

* push alapú – vki megfogalmazza, aki érti - Gergő

### A felület osztály-struktúrája

[Osztálydiagram. Minden új osztály, és azon régiek, akik az újakhoz közvetlenül kapcsolódnak.]

## A grafikus objektumok felsorolása

[Az új osztályok felsorolása. Az régi osztályok közül azoknak a felsorolása, ahol változás volt. Ezek esetén csak a változásokat kell leírni.]

* szövegesen is le kell írni, hogy mi változott – az abstractfactoryk kerültek a modellbe – lásd o.diagram - ÉN
* objektumok felsorolása – Geri
* osztályok elemzése – Geri és én

### Osztály1

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]

#### Ősosztályok

[Mely osztályokból származik (öröklési hierarchia)]

Legősebb osztály → Ősosztály2 → Ősosztály3...

#### Interfészek

[Mely interfészeket valósítja meg.]

#### Attribútumok

[Milyen attribútumai vannak]

* **attribútum1**: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
* **attribútum2**: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa

#### Metódusok

[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása, láthatósága (UML jelöléssel)
* **int bar(Osztály5 o1)**: metódus leírása, láthatósága (UML jelöléssel)

### Osztály2

#### Felelősség

[Mi az osztály felelőssége. Kb 1 bekezdés. Ha szükséges, akkor state-chart is.]

#### Ősosztályok

[Mely osztályokból származik (öröklési hierarchia)]

Legősebb osztály → Ősosztály2 → Ősosztály3...

#### Interfészek

[Mely interfészeket valósítja meg.]

#### Attribútumok

[Milyen attribútumai vannak]

* **attribútum1**: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa
* **attribútum2**: attribútum jellemzése: mire való, láthatósága (UML jelöléssel), típusa

#### Metódusok

[Milyen publikus, protected és privát metódusokkal rendelkezik. Metódusonként precíz leírás, ha szükséges, activity diagram is a metódusban megvalósítandó algoritmusról.]

* **int foo(Osztály3 o1, Osztály4 o2)**: metódus leírása, láthatósága (UML jelöléssel)
* **int bar(Osztály5 o1)**: metódus leírása, láthatósága (UML jelöléssel)

## Kapcsolat az alkalmazói rendszerrel

[Szekvencia-diagramokon ábrázolni kell a grafikus rendszer működését. Konzisztens kell legyen az előző alfejezetekkel. Minden metódus, ami ott szerepel, fel kell tűnjön valamelyik szekvenciában. Minden metódusnak, ami szekvenciában szerepel, szerepelnie kell a valamelyik osztálydiagramon.]

* mindenki a sajátját

## Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2025.04.30. 17:00 | 1 óra 30 perc | Kohár | Tevékenység:  Felhasználói felület alapterveinek kidolgozása |
| 2025.05.01. 11:00 | 1 óra | Kohár | Tevékenység:  Alapvető panelek leírása az osztálydiagrammban |
| 2025.05.02. 14:00 | 1 óra | Kohár | Tevékenység:  Szekvencia diagrammok leírása az panelekhez. |
| 2025.05.03. 15:00 | 1 óra 30 perc | Kohár | Tevékenység:  Szekvencia és osztály diagrammok javítása |
| 2025.05.04. 14:00 | 1 óra 30 perc | Kohár | Tevékenység:  Szekvencia és osztály diagrammok javítása |