Beágyazott rendszerek szoftvertechnológiája

Házi feladat

**Tervdokumentáció**

Lövöldözős Játék: Szóbalövő

Készítők:

Hencz András (IWQKZW)

Rácz Benedek (AQAEFR)

~~Bitay Olivér (R6R1I5)~~

Konzulens: Györke Péter

# Bevezetés

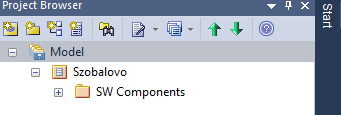
A házi feladatként beadott játékunk célja hiányos szavakba belelőni a megfelelő betűket. A feladatban meg kell valósítani a játék motorját, egy hálózati részt és a GUI-kat.

A játék játszható egy- illetve kétszemélyes módban is, ezért szükség van hálózati kapcsolat kialakítására.

Az UML szerű dokumentációhoz az Enterprise Architect nevű szoftvert használjuk a Sparx Systems-től. Ebben egyszerűen ábrázolhatók az egyes UML komponensek. A program használata nem ingyenes, de letölthető 30 napos próbaverzió. A program hátterében egy ORACLE adatbázis található mely tartalmazza a létrehozott komponenseket és a közöttük lévő kapcsolatokat is! Az Enterprise Architect segítségével modellezhető mind a szoftver statikus mind a dinamikus viselkedése.

# Alapvető struktúrák, osztályszervezés

Mint említettük a Szóbalövő játszható kétszemélyes módban is, ezért szükség van valamiféle hálózati kapcsolatra. A program motorja egy szerveren fut, amelyhez egy kliens segítségével lehet csatlakozni. A kliens feladata létrehozni a menüket és a GUI-t, kapcsolódni a szerverhez majd a játék során a fenntartania a folyamatos kommunikációt.

Első lépésként létrehoztunk a programnak egy modell-t:

A SW Component mappába kerülnek az egyes komponensekhez tartozó class-ok.

A program indulásához szükséges main függvény a MainClass osztály tartalmazza. Ez példányosítja az adott játékhoz tartozó szervert és a klienst.

## Beállítások

Fontos még itt megemlíteni a beállításokat, mert ezeknek is nagy szerepük van az osztályok kialakításában.

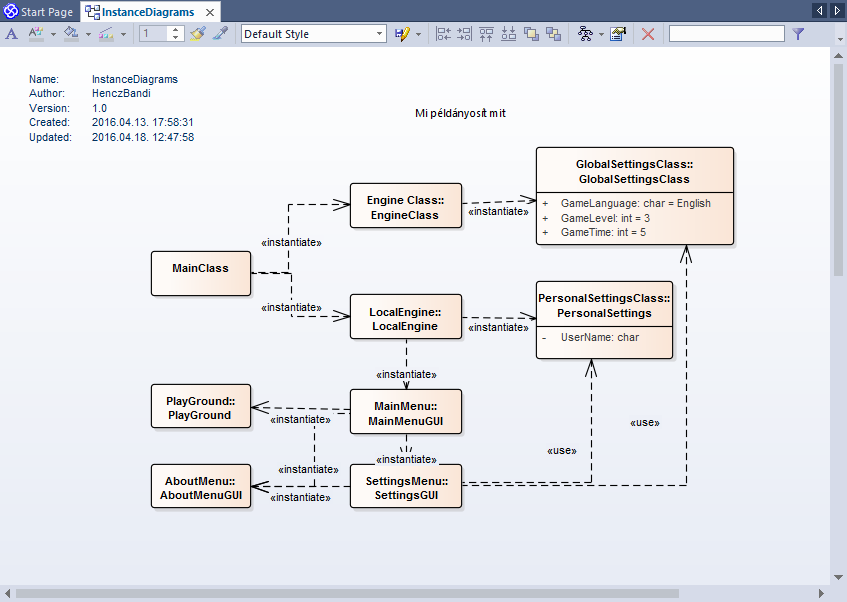
A specifikációban meghatározott beállításokat két különálló részre bonthatjuk szét. Vannak olyanok, amik a játékkal kapcsolatosak, illetve olyan, amely a játékossal. Ez utóbbi a játékos neve, az előbbihez tartozik a játék nehézségi szintje, a szavak nyelve valamint a játékidő.

A játékos nevét természetesen a lokális motornak (tehát a kliensnek) kell ismerni, a többi beállítást pedig a motornak, ami a szerveren fut.

Amikor elindítunk egy új játékot akkor a kliens csatlakozik a megfelelő szerverhez. Egy játékos mód esetén ez a saját maga által elindított szerver a localhoston, multiplayer esetén pedig meg kell határozni, hogy melyik szerverhez csatlakozzon. Természetesen annak a játékosnak a beállításai fognak érvényre jutni kétjátékos mód esetén, amelyiknek a szerveréhez csatlakozik a másik játékos is.

Látható tehát, hogy többféle interfész is szükséges a beállítások elmentésére. Ezeknek létrehoztuk a GameSettings és a GamerSettings osztályokat.

Az alábbi ábrán az látható, hogy melyik osztályt mi példányosít:



Természetesen ez a diagram is megtalálható a csatolt eap fájlban.

# GUI szervezés

Alapvetően a játék eléggé statikus jellegű, a felhasználó beavatkozásai alapján kell feladatokat elvégezni. Ez alapján mondhatjuk, hogy a GUI szerepe a játék vezérlésében kiemelkedő.

Alapvetően kétféle felhasználói felületet kell elkülöníteni. Az egyik a statikus menük, a másik a (játékos beavatkozására) dinamikusan változó játéktér.

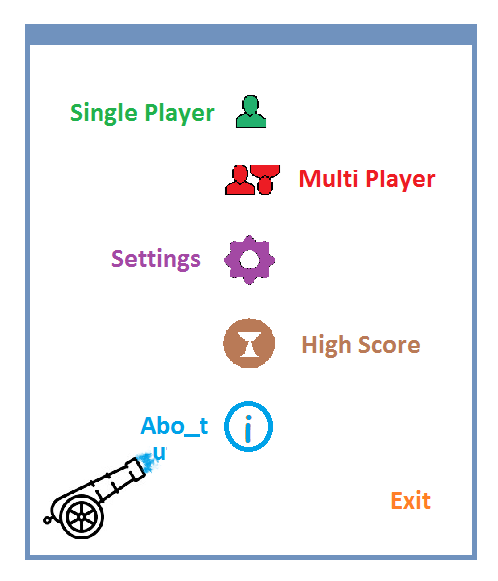
## Menük

Alapvetően háromféle menüt kell megkülönböztetnünk. Az egyik a főmenü (main menu), amely tartalmazza a játékindításhoz szükséges gombokat, a másik a beállítás menü (settings menu) amelyben a beállításokat menthetjük el, illetve a harmadik az about menu.

Ezeket JFrame segítségével tervezzük megvalósítani.

A menüket a kliens példányosítja.

A főmenü terve:



A beállítások menü terve már szerepel ebben a dokumentumban . Az about menü tartalmazza a készítők nevét és elérhetőségét.

## Játéktér

A játéktér a játék egyik legfontosabb és legbonyolultabb eleme, mivel ennek a szerepe a konkrét játék megjelenítése, a játék irányítása, és az eredmények kiírása.

A játéktéren találhatóak a szavak, az ágyú és a belövendő betűk is.

Az ágyút a jobb és bal nyilakkal lehet mozgatni, a kilövendő betűk között a „space” billentyű segítségével lehet váltani, a lövedéket kilőni pedig a felfelé mutató nyíllal vagy az enterrel lehet. Egyszerre maximum 1 lövedéket lőhet ki a program.

Ütközésvizsgálat:

A kilőtt lövedéket meg kell vizsgálni, hová csapódik be a szavak közé. Ezt egy mátrix segítségével lehet optimálisan megvalósítani. A játékteret lebontjuk nagyobb egységekre. Egy-egy ilyen egységet a mátrix megfelelő eleme reprezentál. A játékoshoz képest a játéktér bal felső sarka a mátrix első eleme és a jobb alsó sarka az utolsó. A mátrixba számokat tárolunk. A számok azt jelentik, hogy a mátrix adott eleme melyik szónak melyik karakteréhez tartozik.

Amikor kilövünk egy lövedéket akkor az Y koordinátát vesszük az ütközésvizsgálatnál alapul. Amint a lövedék eléri az első „szintet” ahol már lehetnek szavak, akkor a program megvizsgálja, hogy a mátrix melyik eleméhez csapódik be a lövedék. Ha 0, akkor ott nincsen semmi és mehet tovább a lövedék, ha negatív, akkor ott befagyasztott szó van, az megállítja a lövedéket, ha pozitív szám, akkor meg kell vizsgálni, hogy az adott szó értelmes-e az adott helyre belőtt karakterrel.

Ha igen akkor megkapja a játékos a pontot, ha nem akkor befagy a szó.

A játéktéren szerepel továbbá a játékos neve és a még a játékból hátralévő idő is. Ehhez egy timer-t használunk, aminek az értékét a beállításokban eltároljuk. Ha vége a programnak, akkor felugrik egy ablak az eredményekkel, és a GUI elküldi ezeket a motornak, hogy eltárolja.

# Kommunikáció a motor és a GUI között

A játék motorja felelő azért, hogy a megfelelő számú, méretű és nyelvű szavakat kapja meg a GUI, illetve azért, hogy meghatározza, hogy egy adott szó helyes-e vagy sem.

A motor kezeli a szótárakat, illetve a szerveroldali kapcsolatért is felelős.

# Szerver, kliens létrehozása

A hálózati réteg létesítésére hozzuk létre a Client és a Server osztályokat. Ezek implementálják az IClient és IServer interface-eket, valamint TCP/IP kapcsolatot létesítenek a másik eszközön futó programmal.  
  
Ha szerver módot állítunk be a játék kezdésekor, a program létrehoz egy szerver socketet, és várja, hogy csatlakozzanak hozzá. Sikertelen csatlakozás, time-out esetén hibát jelez, és újra megpróbálhatjuk a multiplayer mód beállítását.

Ha kliensként indítjuk, akkor megpróbál csatlakozni a szerverhez előre beállított port alapján. Csatlakozás sikertelensége esetén hibát jelez.

# Hálózati kommunikáció

A játék során a másik játékos számára releváns történéseket objektumok tartalmazzák, ezeknek az példányoknak az átvitelét kell megvalósítani. A hálózati kommunikáció során a küldés ObjectOutputStream-en történik serializable objektumok küldésével, és ObjectInputStream-en keresztül történik azok fogadása.