**Hálózati tenisz játék − Tervezési dokumentáció**

Bevezetés

Jelen dokumentum célja ismertetni a Villamosmérnöki MSc képzésen indított Beágyazott rendszerek szoftvertechnológiája című tárgyra készült házi feladat (Hálózati tenisz játék) tervezési és megvalósítási fázisait, illetve a feladatok felosztását a 2016/17 II. félévben.

*Házi feladat kiírás: Tenisz*

A jól ismert játék egy egyszerű 2 dimenziós hálózatos megvalósítása. Kétszemélyes játék egyszerűsített fizikai törvények betartásával (lehessen a labdát "csavarni"). Számolja a pontokat, figyelje a játék végét. A játékállás menthető és visszatölthető legyen.

*A projektben résztvevő személyek:*

*Hallgatók:*

* Ágoston Dávid – U1DHH4
* Juhancsik Tamás – X1SLN4
* Kovács Adrienn – IR9TVR

*Konzulens:*

* Ferencz Bálint - Méréstechnika és Információs Rendszerek Tanszék

A feladatok felosztása

A program elkészítését három különböző részre osztottuk, így a csapat mindhárom tagjának egy-egy terület jutott:

* Hálózati kommunikáció – Ágoston Dávid
* Játéklogika – Juhancsik Tamás
* Felhasználói felület – Kovács Adrienn

Tervezés

A tervezés során első lépésként elkészült a program specifikációja, melyben leírásra kerül, hogy milyen követelményeket támasztunk a megvalósítandó játékkal szemben, valamint egy elképzelt működése a programnak. Az elkészült specifikációt az 1. melléklet tartalmazza.

A következő lépésben elkészítettük a program működését és felépítését tartalmazó ábrákat, amelyek az alábbiak:

* Aktivitás diagram (2. melléklet)
* Szekvencia diagramok (3. melléklet)
* Osztály diagram (4. melléklet)

*Aktivitás diagram*

Az aktivitás diagram egy folyamatábrában ismerteti a program alapvető működését/folyamatát.

*Szekvencia diagramok*

A szekvencia diagramok segítségével meghatározható a program belső viselkedése, az osztályok közötti kapcsolatok, valamint a program és felhasználó(k) közötti interakció.

*Az alábbi szekvencia diagramok készültek el a tervezés során:*

Beállítások módosítása

Megadja, hogy a felhasználó hogyan és milyen tulajdonságait tudja változtatni és eközben milyen belső működés zajlik le a programban.

Játék szerverként

Megadja, hogy a felhasználó hogyan tud szervert indítani, szerverként játszani, illetve a program belső működése is megfigyelhető.

Játék kliensként

Megadja, hogy a felhasználó hogyan tud klienst indítani, szerverhez csatlakozni, kliensként játszani, illetve a program belső működése is megfigyelhető.

*Osztály diagram*

Az osztálydiagramon megfigyelhető a program struktúrája, a megvalósított osztályok, illetve azok kapcsolata, viszonya. A fontosabb osztályok, illetve a példányosított objektumok rövid leírása a „*Fontosabb osztályok, objektumok leírása”* című fejezetben található.

*Használati utasítás*

Az implementáció után készült a program alapvető működését, illetve annak irányítását leíró Használati utasítás a felhasználók számára, annak érdekében, hogy program kezelését a felhasználók a legegyszerűbben elsajátíthassák. A használati utasítást az 5. melléklet tartalmazza.

Fontosabb osztályok, objektumok leírása

*Main*

Ez az osztály tartalmazza a main() függvényt és példányosítja a Control osztályt.

*Control*

Ez az osztály valósítja meg a játék logikát/vezérlést, a Network osztály példányán keresztül pedig képes a hálózaton keresztül kommunikálni egy másik Control osztály példánnyal. A következőkben a Control osztály főbb tagfüggvényeinek rövid leírása olvasható:

* **selectOptions(): void**

A függvény megnyitja a GUI beállítások ablakát.

* **setName(String newName): void**

Ez a függvény elmenti a beállításokban beállított nevet.

* **setScore(int newScore): void**

Ez a függvény elmenti a beállításokban beállított elérendő pontszámot.

* **showMenu() :void**

A függvény megnyitja a GUI menü ablakát.

* **startServer(): void**

Ez a függvény egy szerver socket elindítását kezdeményezi.

* **startClient(): void**

Ez a függvény egy kliens socket elindítását kezdeményezi.

* **joinSuccesfull(): void**

A szerver és kliens sikeres csatlakozását veszi észre.

* **newGame(): void**

Ez a függvény új játékot indít.

* **loadGame(): void**

Ez a függvény a külső mySQL adatbázisból tölt vissza egy korábban elmentett játékot (feltéve, ha létezik).

* **saveGame(): void**

A játék során a Save gombra kattintva az aktuális eredmények mentésre kerülnek egy mySQL adatbázisba.

* **startGame(): void**

Ez a függvény elindítja a játékot.

* **startSet(): void**

Ez a függvény vezérli az aktuális labdamenetet, a következőkben ismertetésre kerülő függvények segítségével.

* **ballPos(int currentX, int currentY): void**

Ez a függvény számítja a labda aktuális pozícióját.

* **racketPos(int currentY): void**

Ez a függvény számítja az ütő aktuális pozícióját.

* **updateScore(): void**

Gól esetén ez a függvény állítja be az új eredményeket.

*Network*

A network, illetve annak leszármazottjai (Server, Client osztályok) valósítják meg a hálózati kommunikációt. A kommunikáció TCP/IP alapú websocket technológia alapján valósul meg. A következőkben a Network osztály főbb tagfüggvényeinek rövid leírása olvasható:

*GUI*

A GUI biztosítja a felhasználói interfészt a játékosok számára, ezen osztály példánya keresztül tudják befolyásolni a program belső viselkedését, működését. A következőkben a GUI osztály főbb tagfüggvényeinek rövid leírása olvasható: