*Beágyazott rendszerek szoftvertechnológiája házi feladat*

**Tetris**

# Csapattagok

* Kovács-Nagy Máté *UDNHPI*
* Virágh Eszter *CVMO4Y*
* Nárai Ádám *RPGLH6*

# Specifikáció

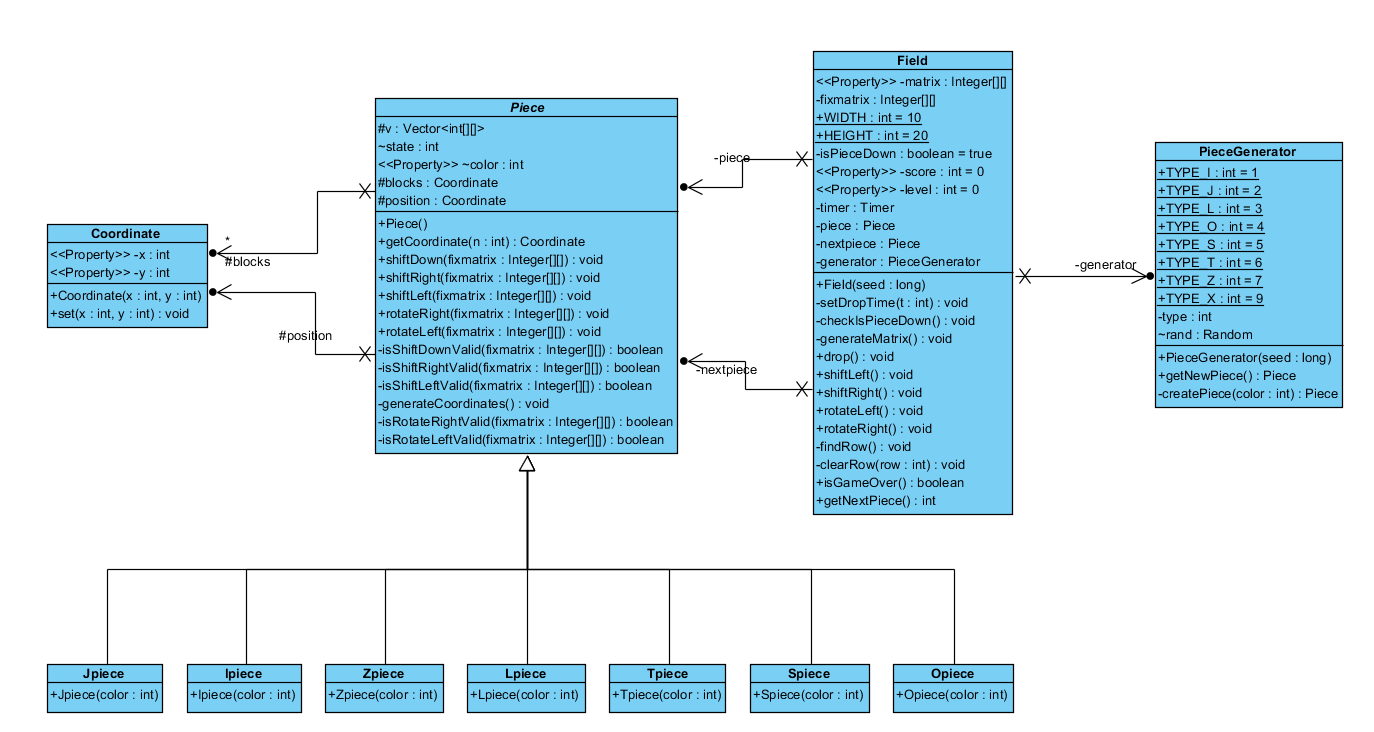
A program indulásakor a játékos választhat a szerver és kliens módok között. Szerver módban várakozik egy másik játékos csatlakozására, kliens módban pedig csatlakozni tud egy szerverhez az IP címének megadásával. Ha a játék elindult mindkét játékos a saját játékterületét játszik, de látja közben a másik játékos állását is. A két játokos sorban ugyanazokat a véletlenszerűen generált négyblokkos elemeket kapja. Ahogy egy elem megjelenik a képernyő tetején, ezt a játékos a jobbra/balra nyilakkal tudja oldalra mozgatni, a lefelé nyíllal vagy Space-el rögtön ledobni, valamint a felfelé nyíllal vagy X-el jobbra és a Ctrl-al vagy Y-al pedig balra forgatni. Az adott elem magától is folyamatosan „esik lefelé”, ennek sebessége a játék előrehaladtával az elért pontszám alapján fokozatosan nő. Ha egy elem leért a játéktér aljára, egy új jelenik meg a képernyő tetején és ha egy sor teljesen telített, akkor az eltűnik. A játéknak akkor van vége, ha a játéktér „betelt”, azaz a legfelső sorába is kerül elem. A játékosok az elhelyezett elemek és a törölt sorok száma alapján kapnak folyamatosan pontokat. A játékot az nyeri, aki nagyobb pontszámot tud elérni.

# Tesztesetek

* A forgatás és mozgatás működése olyan esetekben, amikor nem mindegyik kivitelezhető. (például ha egy elem a pálya szeleinél vagy már lerakott elemek közelében van)
* Egyszerre több sor telik be.
* Hálózati kapcsolat megszakad.

# Fejlesztői dokumentáció

## Játékmotor (Nárai Ádám)



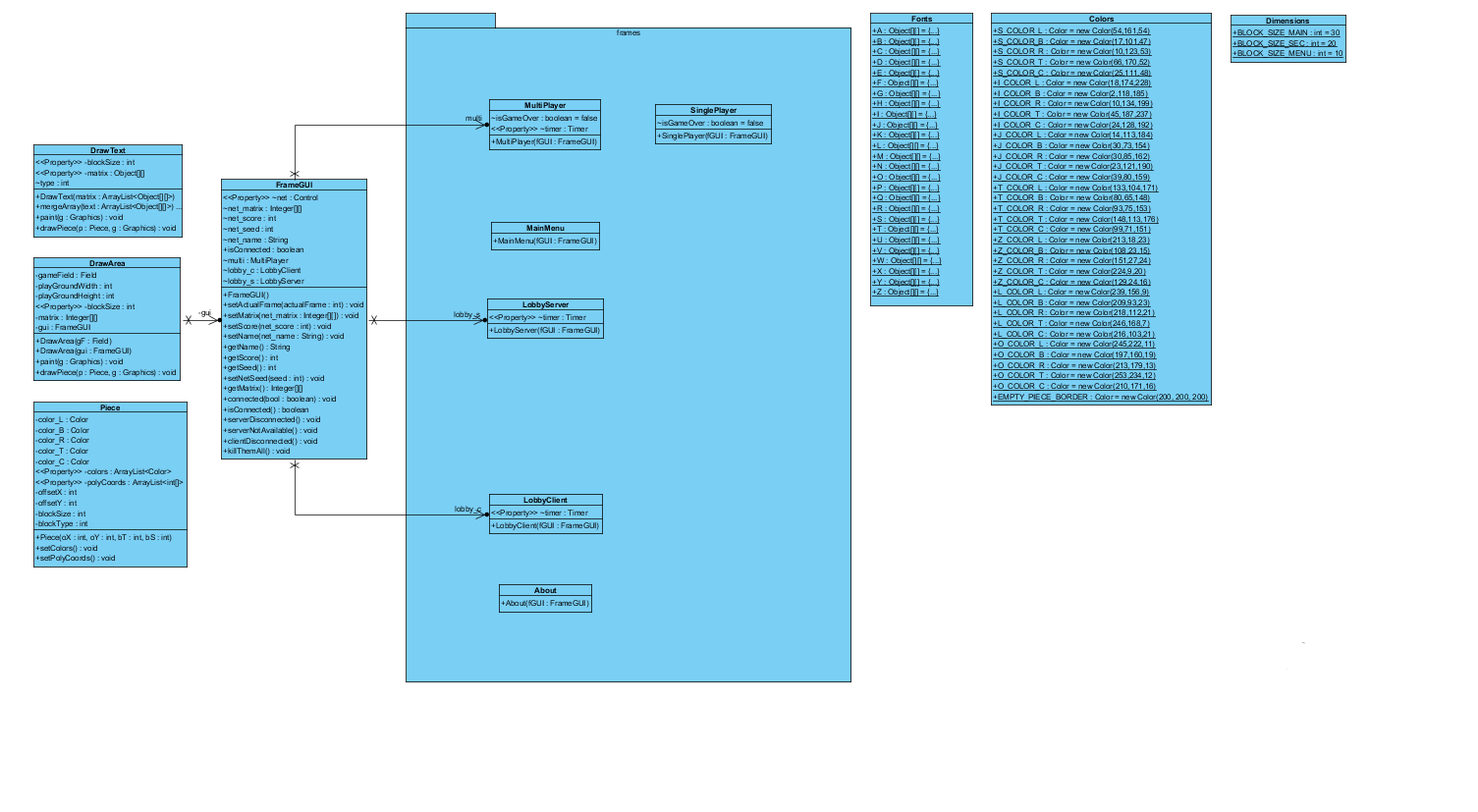
Az engine pacakage-ben találhatóak a játékmotort megvalósító osztályok. A Field osztályt példányosítva lehet játékteret létrehozni, majd annak publikus tagfüggvényeit kell hívni a játékmechanika megfelelő vezérléséhez. A példányosításkor a konstruktor argomentumában meg kell adni az elemeket generáló randomgenerátor integer seed értékét. Erre azért van szükség, hogy mind a szerver mind a kliens azonos elemeket generáljon, de az elemek mégis véletlenszerűek legyenek. A Field.WIDTH és Field.HEIGHT tartalmazzák a játéktér méretét.

package.Field publikus tagfüggvényei:

* setDropTime(int t): Az elemek automatikus esésének periódusideje adható meg vele ms-ban.
* boolean isGameOver(): true értékkel tér vissza, ha a játéknak vége
* drop(): Az aktív elem leejtése a játéktér aljára.
* shiftLeft(): Aktív elem balra mozgatása.
* shiftRight(): Aktív elem jobbra mozgatása.
* rotateLeft(): Aktív elem balra forgatása.
* rotateRight(): Aktív elem jobbra forgatása.
* Integer [][] getMatrix(): Visszatér a játéktér pillanatnyi állapotával egy WIDTHxHEIGHT méretű Integer mátrixként. Az Integer értékek a PieceGenerator osztályban defininiált elemtípusoknak felelnek meg. (pl.: PieceGenerator.TYPE\_I = 1)

A Coordinate osztály egy egyszerű két int koordinátát tartalmazó tároló. A Pice osztály az absztrakt elem osztály, ilyen típusú referenciával kezelhetőek az egyes elemeket megvalósító leszármazott osztályai (Ipice, Jpiece, stb.) Továbbá ez az osztály valósítja meg az elemek mozgatását, forgatását és annak ellenőrzését, hogy az adott esetben ezek a műveletek érvényesek-e. A PieceGenerator osztály felel a véletlenszerű új elem generálásáért.

## GUI (Kovács-Nagy Máté)



A graphics package-ben lett implementálva a játékmotorban legenerált játékteret tartalmazó mátrix megjelenítése, a panelek inicializálása, és az azok közötti váltások. Ezen felül a játékos egérrel és billentyűzettel való beavatkozása is ezen osztályokban lett lekezelve. 6 különböző „activity” lett definiálva:

* **MainMenu**: A kezdőképernyő, tartalmazza a többi, innen elérhető képernyő elérését, egy-egy nyomógomb formájában.
* **SinglePlayer**: Az egyjátékos módot tartalmazó képernyő, a játéktérből, valamint az aktuális pontszámokból áll, tartalmazza a főmenübe való visszalépéshez szükséges gombot.
* **LobbyServer**: Többjátékos mód esetén szerver módban itt értesülhetünk az IP címünkről, a Host gombra nyova pedig várakozhatunk a kliens csatlakozására.
* **LobbyClient**: Többjátékos mód esetén kliens módban itt adhatjuk meg a szerver IP címét, és csatlakozhazunk hozzá.
* **Multiplayer**: A szerver és a kliens egyaránt erre a képernyőre kerül sikeres csatlakozás esetén, itt mindkét játékos láthatja saját, és ellenfele játékterét, és pontjait.
* **About**: A projektfeladat ismertetése.

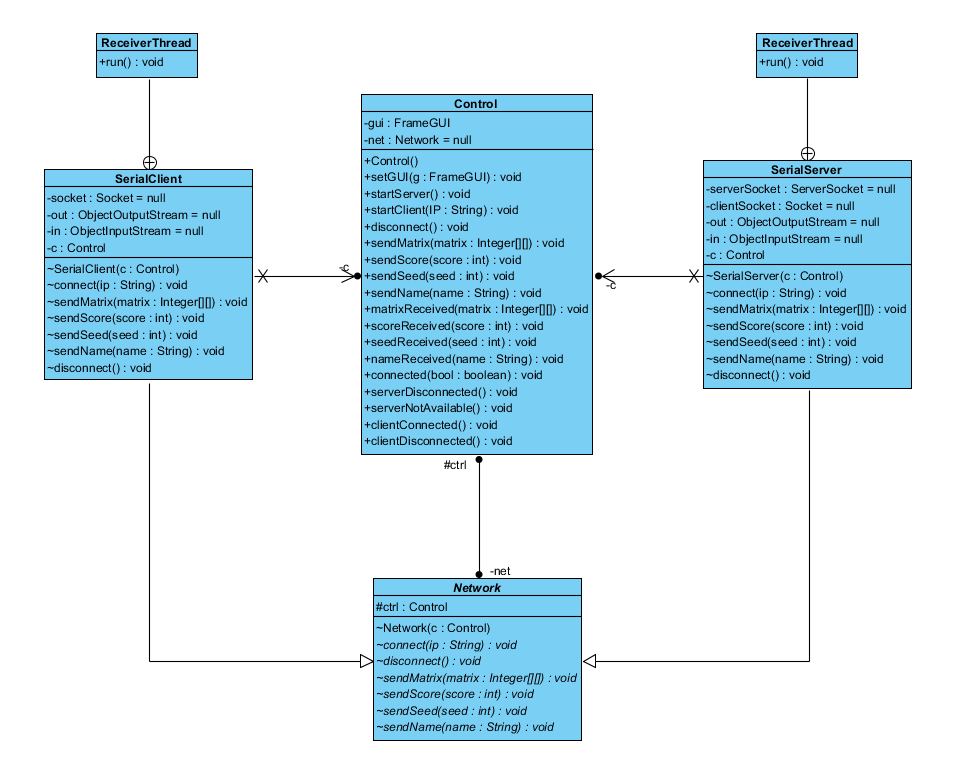
A **FrameGUI** osztály végzi el a képernyők közötti váltogatást a setActualFrame() függvénnyel, a get és set függvényekkel pedig lekérhetjük az ellenfél, illetve frissíthetjük a saját információinkat.

A **Colors**, **Dimensions** és **Fonts** osztályokban konstansok vannak definiálva, melyeket a grafikai megjelenítéskor felhasználunk.

A **DrawArea** és a **DrawText** a játéktér kirajzolásáért, valamint a főmenüben található „gombok” kirajzolásáért felelősek, az előbb említett konstansokat ezen osztályok hívják meg.

Mindkét előbbi osztály a **Piece** osztályt példányosítja, amikor egy blokkot rajzol a rajztérre. Minden egyes blokk 5 poligonból épül fel, mind az 5 poligon különböző színű, és összesen 7 fajta blokk létezik.

## Hálózat (Virágh Eszter)



A hálózatot megvalósító osztályok a network package-ben találhatóak. A hálózat kezelése a Control osztály publikus tagfüggvényein keresztül valósítható meg. Az osztály példányosítása után a setGUI(FrameGUI) tagfüggvényével meg kell adni egy FrameGUI megjelenítést vezérlő osztályt, mely tartalmazza a szükséges hálózat kezelő tagfüggvényeket.

A Control osztály publikus tagfüggvényei:

* startServer(): Szerver indítása.
* startClient(String): Csatlakozás a megadott IP címen található szerverhez.
* disconnect(): Kapcsolat bontása.
* sendMatrix(Integer[][]): Mátrix küldése hálózaton.
* sendScore(int): Score küldése hálózaton.
* sendSeed(int): Randomgenerátor seed érték küldése hálózaton.
* sendName(String): Játékos nevének küldése a hálózaton.

A következő publikus tagfüggvényeket a hálózatkezelő osztályok egy adott esemény bekövetkezésekor hívják, ezek feladata csak annyi, hogy tovább hívják a GUI eseménykezezelő függvényeket:

* matrixReceived(Integer[][])
* scoreReceived(int)
* seedReceived(int)
* nameReceived(String)
* connected(boolean)
* serverDisconnected()
* serverNotAvailable()
* clientDisconnected()
* clientConnected()

A Network osztály a SerialClient és SerialServer osztályok ősosztálya, ezek rendre a kliens és a szerver működését valósítják meg. Ezekben az osztályokban felül vannak írva a Network tagfüggvényei az elvárt működésnek megfelelően. Az egyes eseményeknél (Sikertelen csatlakozás, Szerver nem elérhető, Sikeres csatlakozás, stb.) a Control osztály megfelelő eseménykezelő tagfüggvényei kerülnek meghívásra, valamint bizonyos esetekben a konzlra is kiírásra kerül a hibaüzenet. A szerver és a kliens mindig az 10007 portot használják. A négy különböző típusú elem küldése és fogadása úgy valósul meg, hogy küldés előtt egy jelzőbyte-ot küldünk ami az adat típusát jelöli, majd ezt fogadás előtt kiolvassuk és ez alapján eldönthető, hogy éppen milyen adat érkezett. A küldött byte mátrix esetén 1, score esetén 2, seed esetén 3, név esetén pedig 4.

# Felhasználói dokumentáció

A felhasználó az alkalmazás elindításakor a főmenüből indul, innen 5 lehetőség közül választhat. A **Singleplayer** gombra kattintva egyjátékos módban játszhat, ekkor megjelenik a **Singleplayer** ablak. Itt véletlenszerűen történik a blokkok sorsolása, a játékos a jobbra és balra billentyűkkel mozgathatja a blokkot, a felfel gombbal forgathat, a lefele gombbal pedig ledobhatja azt. Jobb oldalt látható, hogy hanyadik szinten van, és hány pontja van, ha egy vízszintes sor minden helye tartalmaz elemet, úgy az a sor eltűnik. A játék célja minél több pontot összegyűjteni. az **Exit** gombra kattintva, vagy az Esc gombot megnyomva bármikor visszatérhet a főmenübe. Ha a játéktér betelik blokkokkal, a játéknak vége, a felhasználó szembesül az elért pontszámmal.

A Server illetve kliens gombra kattintva a felhasznál a **Server** és a **Kliens** lobbyba kerül. A **Server lobbyban** a felhasználó beírhatja a nevét, és megtudhatja az IP címét, erre lesz majd szüksége a kliensnek, hogy csatlakozni tudjon a játékhoz. A host gombot megnyomva a **Server** játékos várakozik a **Kliens** csatlakozására.

A **Kliens** szintén beírhatja a nevét, valamint az IP Address részhez a Szerver IP címét, és a Connect game gombra kattintva (amennyiben a **Server** hostolt, és létrejön a kapcsolat) mindkét játékos a **Multiplayer** ablakba kerül.

A **Multiplayer** ablakban ugyanaz a mechanizmus érvényes, mint a **Singleplayer** ablakban, annyi különbséggel, hogy az ellenfelünk játékterét és pontszámát is láthatjuk. A játéknak akkor van vége, ha mindkét játékos játéktere betelt, ekkor az nyer, aki több pontot gyűjtött öszze. Ha játék közben bármelyik játékos kilép, úgy a kapcsolat megszakad, és a másik játékos átkerül a főmenübe.

Az **About** gombot megnyomva a játék készítőit láthatjuk, és az adott feladataikat, az **Exit** gombbal pedig kiléphetünk az alkalmazásból.