Torpedó játék tervezői kézikönyv

Erdélyi Janka Anna

Olasz-Szabó Sára

Vajay Levente

A torpedó játék program felépítése

A torpedó játék program alapvetően három részegységre bontható: a játék implementálása, a GUI és a hálózati kapcsolatok létesítése. A továbbiakban ennek a három program résznek a részletes kifejtése található, ezek osztályai, az osztályok attribútumai, függvényei.

A torpedó játék belső működésének implementálása

ShipSegment osztály

A ShipSegment osztály a hajó egy celláját jelképezi.

Attribútumai:

`private int xKoord=-1`: A hajó szegmens vízszintes koordinátája [1,10] intervallumba eső pozitív egész szám

`private int yKoord=-1:` A hajó szegmens függőleges koordinátája [1,10] intervallumba eső pozitív egész szám. Értéke a függőleges tengely mentén haladva fentről levele nő.

`private boolean hit`: Az értéke `true`, ha a szegmenst már eltalálták egyébként `false`

Metódusai

`public ShipSegment(int x ,int y):` A hajó szegmens konstruktora. Létrehozásakor megadandó a hajószegmens x és y koordinátája

`public int getxKoord():` XKoord attribútum getter függvénye

`public int getyKoord():` yKoord attribútum getter függvénye

`public boolean isHit():` a hit változó getter függvénye

`public boolean isequalSegment(ShipSegment Segment):` megvizsgálja, hogy az adott hajó szegmens a függvényhívásakor kapott hajószegmenssel azonos helyen helyezkedik e el a pályán (x és y koordináták páronkénti egyezőségét ellenőrzi).

`public void HitSegment():` az adott hajószegmens hit változóját true-ra állítja, „kilövi” a hajószegmenst.

Ship osztály

A ship osztály a hajókat jelképezi.

Attribútumai:

`private int ShipSegmentNumber:` egy hajó szegmenseinek a száma

`private ArrayList<ShipSegment> newSegmentArray:` a hajó szegmenseinek láncoltlistája

`private boolean sinks=false`: true, ha egy hajó összes szegmensének hit változója true értékű

`private int ShipID:` a hajó azonosító száma

Metódusai

`public Ship(int ShipSegmentNumber,int ShipID):` a hajó osztály konstruktora. Létrehozáskor megadandó a hajószegmensek száma, és a hajó azonosítója.

`public int getShipID():` A hajó azonosítójának getter függvénye

`public void addSegment(ShipSegment newShipSegment):` Hozzáad egy hajószegmenst az adott hajó láncoltlistjához.

`public boolean fireShip(ShipSegment firedPlace ):` a paraméterként kapott hajószegmensről megvizsgálja, hogy a hajó tartalmazza e, és ha igen, akkor „kilövi”, meghívja a HitSegment() függvényt, ekkor a függvény visszatérési értéke is true. Ellenkezőesetben false értékkel tér vissza.

`public boolean SinkShip():` megvizsgálja, hogy a hajó összes szegmensét „kilőtték” e már, és ha igen akkor a sink attribútumot true értékre állítja és a függvény is tru értékkel tér vissza.

`public boolean inShipSegment(ShipSegment ActShipSegment):` megvizsgálja, hogy a paraméterként kapott hajószegmens egyenlő e bármely a hajó láncolt listájában tárolt hajó szegmensével, és ha igen, akkor true, ellenkező esetben false értékkel tér vissza.

A GameBoard osztály

A GameBoard osztály egy adott hajóelrendezést tárol a játéktáblán.

Attribútumai:

`private int ShipNumber = 10:` a hajók számáz jelöli, egy táblán mindig 10db van.

`private Ship[] newShipArray:` A hajókat tároló tömb

`private boolean endgame = false:` értéke igaz, ha a táblán az összes hajószegmenst eltalálták, az összes hajó elsüllyedt, a játék ekkor végetér.

`private int [][] ShipMatrix= new int[10][10]:` a táblát jelképezi 2D int tömb formájában, ahol a koordináták szerint el helyezkedik hajó, ott az azonosítója kerül beírásra, egyébként 0 értékű.

Az elrendezések tárolása:

A ShipMatrix-szal azonos formában kerültek eltárolásra az adott elrendezések is txt. formátumban.

```
0  0  0  0 32 32 32  0 23  0
0 11  0  0  0  0  0  0 23  0
0  0  0  0 14  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0
13 0  0  0 12  0 22  0  0 21
0  0  0  0  0  0 22  0  0 21
0 41  0  0  0  0  0  0  0  0
0 41  0  0  0  0  0  0  0  0
0 41  0  0  0 31 31 31  0  0
0 41  0  0  0  0  0  0  0  0
```

Itt a számjegyek első karaktere azt mutatja meg, hogy milyen hosszú a hajó, a második karaktere pedig egyfajta sorszám. Ezeket a számokat használom az egyes hajók azonosítójának ShipID is. Ilyen elrendezésekből hat fajtát tárolok, és a későbbiekben leírt módon ezeket forgatom tükrözöm.

Metódusai:

`public GameBoard():`a konstruktor paraméter nélkül hívodik meg és a MakeBoard () függvényt hívja meg.

`public void MakeBoard()`: Elkészít egy elrendezést a ReadShipMatrix és a forgató-tükröző függvények segítségével. Random számokat felhasználva a fájl kiválasztáshoz és a forgató és tükröző függvények végrehajtási számának meghatározásához.

`public int [][] getShipMatrix()`: Visszaadja a hajók mátrixát (ShipMatrix) olyan formává alakítva, ahol a hajókat 1-es jelöli a többi helyen pedig 0 szerepel.

`private void rotate90Clockwise()`: elforgatja a ShipMátrix-ot mátrixot 90°-kal.

`private void flipHorisontal()`: tükrözi a mátrixot horizontálisan

`private void flipVertical()`:tükrözi a mátrixot vertikálisan

`private Ship [] makefreeShipArray()`:Létrehoz egy üres hajó tömböt, amibe a hajókat ID-jükkel és hosszukkal hozzák létre.

`public Ship[] ShipMatrixtoShipArray()`: a hajók mátrixát a hajók tömbbéjé alakítja felhasználva a makefreeShipArray függvényt

`private int getRandomNumberUsingInts(int min, int max)`: vissza ad egy egész számot min és max között (min még lehet, de max már nem)

`public Ship [] getShip()`: a newShipArray változó getter függvénye

`public boolean EndGame()`: ha az összes hajó „elsüllyedt” akkor vége a játéknak ekkor a visszatérési értéke true, és az endgame változót is átállítja true értékűre, különben false.

`public boolean fireGameBoard(ShipSegment firedPlace)`: megvizsgálja, hogy a paraméterként adott hajó szegmens tagja e egy hajónak és ha igen akkor „kilövi”, tehát meghívja a Ship.fireShip(firedPlace) függvényt. A visszatérési értéke true ha talált hajót és false ha nem.

`private void ReadShipMatrix(String filename)`: beolvassa a txt fájlból a ShipMátrixot.

A GUI

GameWindow osztály

A GameWindow osztály egy JFrame osztály, mely ComponentListener-t implementál. A játék indulásánál ezt az osztályt példányosítjuk, megnyílik a játékablak.

Attribútumai:

`GamePanel gamepanel`: A GamePanel osztály egy példánya, ami az adott játékmenethez tartozik.

`StartPanel startpanel`: A StartPanel osztály egy példánya, a hálózati kapcsolatot ez alapján építjük ki.

`static Font PixelFont50`: Egy importált font típust tárol.

`static Font PixelFont20`: Egy importált font típust tárol.

`String status`: A játék aktuális státuszát tárolja.

Metódusai

`public GameWindow()`: A GameWindow konstruktora. Létrehozza az ablakot a megfelelő paraméterekkel és meghívja az openStartPanel() metódust.

`public void openStartPanel()`: Beállítja a játék státuszát „STARTPANEL”-re, majd példányosítja a StartPanel osztályt. Ezzel a JPanel típusú StartPanel megjelenik az ablakban és a hálózati kapcsolathoz szükséges beállításokat el tudjuk végezni. Hozzáadunk a startpanel-hez egy ComponentListener-t.

`public void openGamePanel()`: Beállítja a játék státuszát „GAMEPANEL”-re, majd példányosítja a GamePanel osztályt. Elkezdődik a játék. Hozzáadunk a gamepanel-hez egy ComponentListener-t.

Override metódus:

`public void componentHidden(ComponentEvent e)`: A ComponentListener override metódusa, mely akkor hívódik meg, ha a valamely megfigyelt osztály láthatósága megváltozik. A startpanel és gamepanel láthatósága akkor változik meg, ha azokat be akarjuk zárni, így a forrás alapján az openStartPanel() vagy openGamePanel() metódusokat hívjuk meg.

StartPanel osztály

A StartPanel osztály egy JPanel típusú osztály, mely ActionListener-t implementál. Ebben a JPanel-ben adhatjuk meg a hálózati kapcsolat beállításait a játék indítása előtt.

Attribútumai:

`JButton clientbutton`: Erre kattintva a kliens módot választhatjuk ki.

`JButton serverbutton`: Erre kattintva a szerver módot választhatjuk ki.

`String mode = „SERVER”`: A kiválasztott módot tárolja.

`JTextField portInputTextField`: A kívánt portcímet itt lehet megadni.

`int port`: A portcímet tárolja.

`JButton submitbutton`: Erre kattintva a beállításokat leadjuk.

A kinézethez tartozó változók:

```
int buttonpadding = 350;
int buttonsize = 140;
int bordersizechosen = 7;
int bordersizenotchosen = 2;
```

Metódusai

`public StartPanel()`: A StartPanel konstruktora. Megjeleníti a StartPanel-t a megfelelő beállításokkal, hozzáadja a vizuális elemeket.

`public int getPort()`: Visszaadja a beállított portszámot.

`public int getMode()`: Visszaadja a beállított módot.

Override metódusok:

`public void ActionPerformed():` Érzékeli a kattintásokat a panel gombjain. Ha a kliens vagy szerver gombokat nyomja meg a felhasználó, akkor a mód attribútumot állítja, ha a submit gombot, akkor beolvassa a beírt portszámot és láthatatlanná teszi a StartPanel-t. Ezt a GameWindow ComponentListener-je érzékeli, és bezárja a StartPanelt, elindítja a játékot.

A vizuális elemeket hozzáadó metódusok:

```
public void addClientButton()
public void addServerButton()
public void addSubmitButton()
public void addPortInputTextField()
public void addPortQuieryLabel()
public void addModeQuieryLabel()
```

GamePanel osztály

A GamePanel osztály egy JPanel típusú osztály, mely ActionListener-t és MouseListener-t implementál. Ebben a JPanel-ben maga a játék jelenik meg.

Attribútumai:

`GameBoard board`: Az aktuális játékmenethez tartozó board.

`Network comInterface`: A hálózati kapcsolatért felelő osztály.

`int [][] shipmatrix`: A saját hajóink helyzetét tárolja.

`int [][] lastHitCoordiante`: A legutóbb meglőtt pont koordinátái.

`String mode:` A módot tárolja.

`String status`: A játék státuszát tárolja, ez alapján tudjuk, hogy melyik játékos következik.

`JPanel [][] myshippanels = new JPanel[10][10]:` A sajt hajók kijelzésére szolgáló JPanel-ek.

`JPanel [][] enemyshippanels = new JPanel[10][10]`: Az ellenség hajóinak megjelenítésére szolgáló JPanelek.

`JButton restartbutton`: Erre kattintva a újraindítható a játák.

`private boolean restart = false`: Ha a restartbutton-ra kattintunk, a restart változó igazba vált.

`JButton reconnectbutton`: Erre kattintva úrjracsatlakozhatunk, visszatérünk a StartPanel oldalra.

`private boolean reconnect = false`: Ha a reconnectbutton-ra kattintunk, ez a változó igazba vált

`JLabel turnlabel:` Kijelzi, hogy ki következik és a játék végén az eredményt közli.

A kinézethez tartozó változók:

```
int targetsize = 35;
int seabordersize = 50;
int sidepadding = 90;
```

```
int toppadding = 130;
int labelpadding = 80;
int width = 1000;
int height = 700;
```

Metódusai

`public GamePanel():` A GamePanel konstruktora. Megjeleníti a GamePanel-t a megfelelő beállításokkal, hozzáadja a vizuális elemeket. Inicializálja a hálózati kacsolatot a StartPanel-en beállítottak alapján. Ha kapcsolódott minidkét játékos, a server oldali kezdi a játékot, a kliens esetén meghívja a waitForShot metódust.

`public void waitForShot():` Várja az ellenség leadott lövését. Megjeleníti, hogy talált-e a lövés, az információt továbbítja az ellenségnek is, majd beállítja, hogy ki következik. Ha az ellenfél, akkor maradunk ebben a metódusban, ha mi, akkor kilépünk és a MouseListener és ActionListener override metódusait várjuk.

`public boolean getRestart():` Visszaadja a restart értékét.

`public booelan getReconnect():` Visszaadja a reconnect értékét.

`public int [] getCoordinates(Point point):` A kattintott pont alapján kiszámítja, hogy mely panelre céloztunk a lövéskor.

`public boolean hasBeenShotAt(int i, int j):` Igazzal tér vissza, ha a megadott koordinátájú panelre céloztunk már.

`public void responseDorShot(int [] response):` A lövésünkre kapott választ kezeli le.

`public void restartFunc():` A játék újraindításához szükséges feladatokat végzi el az újraindulás előtt. Elküldi az újraindulás command-ját az ellenfélnek és bezáratja a GamePanel-t.

`public void reconnectFunc():` Az újracsatlakozáshoz szükséges feladatokat végzi el az újracsatlakozás előtt. Elküldi az újracsatlakozás command-ját az ellenfélnek és bezáratja a GamePanel-t.

Override metódusok:

`public void ActionPerformed(ActionEvent e):` Érzékeli a kattintásokat a panel gombjain. Ha a Restart gombot nyomják meg, akkor a játék újraindul, ha a Reconnect gombot akkor a StartPanel-re ugrunk vissza.

`public void mouseClicked(MouseEvent e):` A kattintás helye alapján beazonosítja, hogy mely koordinátájú hajót lőttük meg. Ha mi jövünk, leadja a lövést az ellenségnek, majd a válasz alapján kijelzi a találatot.

`public void mouseEntered(MouseEvent e),`

`public void mouseExited(MouseEvent e):` Ha mi jövünk, az ellenség hajóit célozva jelzik nekünk, hogy mely paneleket célozhatjuk a szín változtatásával.

A vizuális elemeket hozzáadó metódusok:

`public void addSea():` Kirajzolja a hátteret.

`public void addRestartButton():` Hozzáadja a Restart gombot a Panekhez.

`public void addMyNavy():` A shipmatrix-ban lévő hajók helye alapján kirajrolja a saját hajóinkat a táblára.

`public void addEnemyNavy():` Az ellenség hajóinak célozható helyeit rajzolja ki.

`public void addReconnectButton():` Hozzáadja a Reconnect gombot a Panelhez.

`public void addTurnLabel():` Hozzáadja a Labelt, mely kijelzi, hogy ki következik.

`public void shootAtMyShipsGUI(int i, int j, String hitmiss):` Az ellenség lövéseit jeleníti meg, megkapja a koordinátáit a lövésnek és hogy talált-e.

`public void shootAtEnemyShipsGUI(int i, int j, String hitmiss):` Az ellenség hajóira leadott lövéseinket jelzi ki.

`public void enemysTurnGUI():` Átállítja a turnlabel-t és a statust arra, hogy az ellenség jön.

`public void myTurnGUI():` Átállítja a turnlabel-t és a statust arra, hogy mi jövünk.

`public void computingGUI():` Átállítja a turnlabel-t és a statust arra, hogy a játékmenet várakozik.

`public void endofGameGUI():` Átállítja a turnlabel-t és a statust arra, hogy ki nyerte meg a játékot.

Hálózati kommunikáció

A játék kommunikációja két fél közt lokális hálózaton történik TCP/IP kapcsolatot felépítve. A hálózati rész három osztályból épül fel. Egy ősosztályból, ami deklarálja a közös metódusokat, valamint egy kliens és egy szerver osztályból, amiket a játék kezdetén kiválasztott mód alapján használunk fel. A kommunikáció bájt stream alapú. A kommunikáció felépítése a kliens vagy szerver osztályok példányosításával indul. Szerver oldal esetén létrehozunk egy serverSocketet amivel majd fogadjuk a beérkező kapcsolati kéréseket. A kliens oldalon a kiválasztott portra próbálunk meg csatlakozni a helyi hálózaton.

Az üzenetek fogadására a szerver és a kliens osztályokban, hogy a GUI-t ne kelljen várakoztatni, egy külön szálat hoztunk létre. Ezen a külön szálon folyamatosan figyeljük a beérkező bájtokat. Egy üzenet meghatározott alakú koordináta hármast foglal magába ([1,2,0]), amelynek harmadik tagjának három értelmezett értéke lehet (0,1,2), egyéb esetben hibát jelzünk. Az üzenetek végét a záró szögletes zárójel jelzi. Amikor példányosítjuk a kommunikációs interfészt, akkor átadjuk a gamePanel létrehozott példányát is. A beérkező üzeneteknek létrehozott szálból, a 3. koordináta alapján hívjuk meg a gamePanel szükséges függvényét. Eldöntjük, hogy lövés érkezett be, lövésre adott válasz, vagy egy specifikus üzenet, ami a játék újraindításra vagy újracsatlakozására vonatkozik. A restart és reconnect üzenetek a játék menetben felhasznált koordinátáktól teljesen eltérőek, azaz nem lehetséges értékek.

Network/client/server osztály

A változók és függvények a 3 osztályban megegyeznek. A network osztálytól örökli a kliens és szerver osztály a változóit és metódusait, viszont az egyes függvény törzse osztályonként eltérő a dorestart() és doReconnect() függvényeken kívül. A változók nevei egyértelműen árulkodnak a funkcionalitásukról.

Attribútumai

```
protected Socket socket
protected OutputStream outputStream
protected OutputStreamWriter outputStreamWriter
protected InputStream inputStream
protected InputStreamReader inputStreamReader
protected StringBuffer stringBuffer: A beérkező üzenetek feldolgozását segítő buffer
protected String request
protected int[] resetSign={-100,-100}
protected int[]  reConnectSign={-200,-200}
protected GamePanel gamePanel
```

Metódusai

`public void SendData(){}:A paraméterként megkapott tömböt küldi el`

`public int GetPortNUm(){}: Lekérhető a beállított portszám`

`public void CloseConnection(){}: Lezárja a hálózati kommunikáció`

`public void doRestart(){}: Elküldi a másik félnek, hogy restart gombot nyomtak`

`public void doReconnect() {}: Elküldi a másik félnek, hogy reconnect gombot nyomtak`